The LaTeX 2_ε Sources

Johannes Braams
David Carlisle
Alan Jeffrey
Leslie Lamport
Frank Mittelbach
Chris Rowley
Rainer Schöpf

2024-11-01

This file is maintained by the LaTeX Project team.
Bug reports can be opened (category `latex`) at
https://latex-project.org/bugs.html.

Contents

01 ltdirchk.dtx **1**

1 LaTeX System Dependent Initializations **1**

2 Initialization **2**
2.1 INITEX . . . 3
2.2 Some bits of 2e . . . 4

3 texsys.cfg **5**
3.1 texsys.cfg . . . 5
3.2 UNIX (web2c) . . . 6
3.3 UNIX (other) . . . 7
3.4 MSDOS (emtex) . . . 7
3.5 MSDOS (other) . . . 7
3.6 VMS (DECUS TeX, PD VMS 3.6) . . . 7
3.7 VMS (???) . . . 7
3.8 MACINTOSH (OzTeX 1.6) . . . 8
3.9 MACINTOSH (other) . . . 8
3.10 FAKE EXAMPLE . . . 8

4 Setting `\@currdir` **9**

5 Setting `\input@path` **10**

6 Filename Parsing **11**

7 TEX Versions **13**

8 ltxcheck.tex **13**

02 ltplain.dtx **14**

1 Plain TEX **14**

03 ltvers.dtx **37**

1 Version Identification **37**
1.1 Declaring an all-new module 40

04 ltluatex.dtx **42**

1 Overview **42**

2 Core TEX functionality **42**

3 Plain TEX interface **43**

4 Lua functionality **43**
4.1 Allocators in Lua 43
4.2 Lua access to TEX register numbers 44
4.3 Module utilities 45
4.4 Callback management 45

5 Implementation **46**
5.1 Minimum LuaTEX version 46
5.2 Older LATEX/Plain TEX setup 47
5.2.1 Fixes to `etex.src/etex.sty` 47
5.2.2 luatex specific settings 48
5.3 Attributes 49
5.4 Category code tables 49
5.5 Named Lua functions 51
5.6 Custom whatsits 51
5.7 Lua bytecode registers 52
5.8 Lua chunk registers 52
5.9 Lua loader 52
5.10 Lua module preliminaries 54
5.11 Lua module utilities 54
5.11.1 Module tracking 54
5.11.2 Module messages 55
5.12 Accessing register numbers from Lua 56
5.13 Attribute allocation 57
5.14 Custom whatsit allocation 58

5.15 Bytecode register allocation . . . 58
5.16 Lua chunk name allocation . . . 58
5.17 Lua function allocation . . . 59
5.18 Lua callback management . . . 59
5.18.1 Housekeeping . . . 59
5.18.2 Handlers . . . 64
5.18.3 Public functions for callback management . . . 67

05 ltexpl.dtx 73

1 expl3-dependent code 73
1.1 Loader . . . 73
1.2 Using expl3 code . . . 76

2 Document-level command names for expl3 functions 77

06 ltdefns.dtx 80

1 Definitions 80
1.1 Initex initializations . . . 80
1.2 Saved versions of TEX primitives . . . 80
1.3 Command definitions . . . 81
1.4 Robust commands and protect . . . 90
1.5 Acting on robust commands . . . 96
1.5.1 Copying robust commands . . . 98
1.5.2 Showing robust commands . . . 100
1.5.3 Commands defined with `\DeclareRobustCommand` . . . 101
1.5.4 Commands defined with `\newcommand` (with optional argument) 103
1.5.5 Showing environments . . . 104
1.6 Internal defining commands . . . 106

2 Discretionary Hyphenation 110

07 ltcmd.dtx 113

1 Creating document commands 113
1.1 Variables and constants . . . 113
1.2 Declaring commands and environments . . . 117
1.3 Structure of xparse commands . . . 122
1.4 Normalizing the argument specifications . . . 127
1.5 Preparing the signature: general mechanism . . . 135
1.6 Setting up a standard signature . . . 136
1.7 Setting up expandable types . . . 141
1.7.1 Copying a command and its internal structure . . . 144
1.7.2 Showing the definition of a command . . . 150
1.8 Grabbing arguments . . . 156
1.9 Grabbing arguments expandably . . . 168
1.10 Argument processors . . . 174

1.11 Conversion to key–value form . . . 176
1.12 Utilities . . . 179
1.13 Messages . . . 183
1.14 User functions . . . 188

08 lthooks.dtx 194

1 Introduction 194

2 Package writer interface 194
2.1 LaTeX 2_ε interfaces . . . 194
2.1.1 Declaring hooks . . . 194
2.1.2 Special declarations for generic hooks . . . 195
2.1.3 Using hooks in code . . . 196
2.1.4 Updating code for hooks . . . 197
2.1.5 Hook names and default labels . . . 199
2.1.6 The `top-level` label . . . 202
2.1.7 Defining relations between hook code . . . 202
2.1.8 Querying hooks . . . 204
2.1.9 Displaying hook code . . . 205
2.1.10 Debugging hook code . . . 206
2.2 L3 programming layer (`expl3`) interfaces . . . 206
2.3 On the order of hook code execution . . . 209
2.4 The use of "reversed" hooks . . . 211
2.5 Difference between "normal" and "one-time" hooks . . . 212
2.6 Generic hooks provided by packages . . . 212
2.7 Hooks with arguments . . . 213
2.8 Private LaTeX kernel hooks . . . 215
2.9 Legacy LaTeX 2_ε interfaces . . . 215

3 LaTeX 2_ε commands and environments augmented by hooks 216
3.1 Generic hooks . . . 216
3.1.1 Generic hooks for all environments . . . 217
3.1.2 Generic hooks for commands . . . 218
3.1.3 Generic hooks provided by file loading operations . . . 218
3.2 Hooks provided by `\begin{document}` . . . 219
3.3 Hooks provided by `\end{document}` . . . 219
3.4 Hooks provided by `\shipout` operations . . . 221
3.5 Hooks provided for paragraphs . . . 221
3.6 Hooks provided in NFSS commands . . . 221
3.7 Hook provided by the mark mechanism . . . 222

4 The Implementation **222**
4.1 Debugging . . . 222
4.2 Borrowing from internals of other kernel modules . . . 223
4.3 Declarations . . . 223
4.4 Providing new hooks . . . 225
4.4.1 The data structures of a hook . . . 225
4.4.2 On the existence of hooks . . . 226
4.4.3 Setting hooks up . . . 227
4.4.4 Disabling and providing hooks . . . 233
4.5 Parsing a label . . . 235
4.6 Adding or removing hook code . . . 239
4.7 Setting rules for hooks code . . . 260
4.8 Specifying code for next invocation . . . 281
4.9 Using the hook . . . 283
4.10 Querying a hook . . . 289
4.11 Messages . . . 292
4.12 LaTeX 2_ε package interface commands . . . 295
4.13 Deprecated that needs cleanup at some point . . . 299
4.14 Internal commands needed elsewhere . . . 301

09 ltcmdhooks.dtx **303**

1 Introduction **303**

2 Restrictions and Operational details **304**
2.1 Patching . . . 305
2.1.1 Timing . . . 305
2.2 Commands that look ahead . . . 305

3 Package Author Interface **306**
3.1 Arguments and redefining commands . . . 307

4 The Implementation **307**
4.1 Execution plan . . . 307
4.2 Variables . . . 308
4.3 Variants . . . 309
4.4 Patching or delaying . . . 309
4.5 Patching commands . . . 311
4.5.1 Patching by expansion and redefinition . . . 312
4.5.2 Patching by retokenization . . . 320
4.6 Messages . . . 326

10 ltsockets.dtx **328**

1 Introduction **328**

2 Configuration of the transformation process **328**
2.1 The template mechanism 328
2.2 The hook mechanism 329
2.3 The socket mechanism 329
2.3.1 Examples 331
2.3.2 Details and semantics 332
2.3.3 Command syntax 334
2.3.4 Rationale for error handling 335

3 The Implementation **336**
3.1 Debugging the socket structures 336
3.2 The L3 layer commands 337
3.3 Error messages 340
3.4 The LaTeX 2_ε interface commands 340

11 lttemplates.dtx **342**

1 Introduction **342**

2 What is a document? **342**

3 Types, templates, and instances **343**

4 Template types **343**

5 Templates **343**

6 Multiple choices **346**

7 Instances **347**

8 Document interface **348**

9 Changing existing definitions **348**

10 *Ad hoc* adjustment of templates **349**

11 Getting information about templates and instances **349**

12 The implementation **350**
12.1 Variables and constants 351
12.2 Testing existence and validity 353
12.3 Saving and recovering property lists 354
12.4 Creating new template types 356
12.5 Design part of template declaration 357
12.5.1 Storing values 360
12.6 Implementation part of template declaration 361
12.7 Editing template defaults 366
12.8 Creating instances of templates 368
12.9 Using templates directly 370
12.10 Assigning values to variables 371

12.11 Using instances . . . 374
12.12 Assignment manipulation . . . 374
12.13 Showing templates and instances . . . 375
12.14 Messages . . . 376
12.15 User functions . . . 380

12 ltalloc.dtx 382

1 Counters 382

13 ltcntrl.dtx 384

1 Program control structure 384

14 lterror.dtx 388

1 Error handling and tracing 388
1.1 General commands . . . 388
1.2 Specific errors . . . 394
1.3 Tracing . . . 398

15 ltpar.dtx 399

1 Paragraphs 399
1.1 Implementation . . . 399

16 ltpara.dtx 401

1 Introduction 401
1.1 The default processing done by the engine . . . 401

2 The new mechanism implemented for LaTeX 403
2.1 The provided hooks . . . 404
2.2 Altered and newly provided commands . . . 405
2.3 Examples . . . 406
2.3.1 Testing the mechanism . . . 406
2.3.2 Mark the first paragraph of each `itemize` . . . 408
2.4 Some technical notes . . . 408
2.4.1 Glue items between paragraphs (found with fancypar) . . . 408

3 The Implementation 409
3.1 Providing hooks for paragraphs . . . 409
3.2 The error messages . . . 416

17 ltmeta.dtx 418

1 Introduction 418
1.1 \DocumentMetadata . . . 418

2 The Implementation 418

18 ltspace.dtx 421

1 Spacing 421
1.1 User Commands . . . 421
1.2 Chris' comments . . . 421
1.3 Some immediate actions . . . 423
1.4 The code . . . 424
1.5 Vertical spacing . . . 431
1.6 Horizontal space (and breaks) . . . 437

19 ltlogos.dtx 441

1 Logos 441

20 ltfiles.dtx 442

1 File Handling 442
1.1 Safe Input Macros . . . 455
1.2 Listing files . . . 463

21 ltoutenc.dtx 466

1 Font encodings 466
1.1 Removing encoding-specific commands . . . 468
1.2 The order of declarations . . . 469
1.3 Docstrip modules . . . 469
1.4 Definitions for the kernel . . . 470
1.4.1 Declaration commands . . . 470
1.4.2 Hyphenation . . . 478
1.4.3 Miscellania . . . 478
1.4.4 Default encodings . . . 478
1.4.5 Math material . . . 481
1.5 Definitions for the OT1 encoding . . . 482
1.6 Definitions for the T1 encoding . . . 484
1.7 Definitions for the OMS encoding . . . 490
1.8 Definitions for the OML encoding . . . 491
1.9 Definitions for the OT4 encoding . . . 491
1.10 Definitions for the TS1 encoding . . . 493
1.11 Definitions for the TU encoding . . . 498

2 Package files 509
2.1 The fontenc package . . . 509

22 ltcounts.dtx 512

1 Counters and Lengths 512
1.1 Environment Counter Macros 512

23 ltlength.dtx 521

1 Lengths 521

24 ltfssbas.dtx 523

1 Preliminary macros 523

2 Macros for setting up the tables 524

3 Selecting a new font 531
3.1 Macros for the user 531
3.2 Macros for loading fonts 536

4 Assigning math fonts to *versions* 544

25 ltfssaxes.dtx 551

1 Changing the font series 551
1.1 The series lookup table 551
1.2 Mapping rules for series changes 552
1.3 Changing to a new series 560

2 Changing the shape 565
2.1 Mapping rules for shape combinations 566
2.2 Changing to a new shape 568

3 Make sure we win . . . 570

26 ltfsstrc.dtx 572

1 Introduction 572

2 A driver for this document 572

3 The Implementation 573

4 Handling Options 573

5 Macros common to `fam.tex` and `tracefnt.sty` 575
5.1 General font loading 575
5.2 Math fonts setup 581
5.2.1 Outline of algorithm for math font sizes 581
5.2.2 Code for math font size setting 582
5.2.3 Other code for math 583

6 Scaled font extraction 585
6.1 Sizefunctions 593

27 ltfsscmp.dtx 597

28 ltfssdcl.dtx 602

1 Interface Commands 602

29 ltfssini.dtx 634

1 NFSS Initialization 634
1.1 Providing math *versions* 634

2 Custom series settings for main document families 635

3 Supporting nested emphasis 652
3.1 Legacy 656
3.2 Miscellaneous 656

30 fontdef.dtx 662

1 Introduction 662

2 Customization 662

3 The `docstrip` modules 663

4 A driver for this document 663

5 The `fonttext.ltx` file 663
5.1 Encodings 664
5.2 Defaults 666

6 The fontmath.ltx file **668**
6.1 The font encodings used . . . 668
6.1.1 Symbolfont and Alphabet declarations . . . 669
6.2 Math font sizes . . . 669
6.3 The math symbol assignments . . . 670
6.3.1 The letters . . . 670
6.3.2 The digits . . . 671
6.3.3 Punctuation, brace, etc. keys . . . 671
6.3.4 Delimitercodes for characters . . . 672
6.4 Symbols accessed via control sequences . . . 672
6.4.1 Greek letters . . . 672
6.4.2 Ordinary symbols . . . 673
6.4.3 Large Operators . . . 674
6.4.4 Binary symbols . . . 674
6.4.5 Relations . . . 675
6.4.6 Arrows . . . 676
6.4.7 Punctuation symbols . . . 677
6.4.8 Math accents . . . 678
6.4.9 Radicals . . . 678
6.4.10 Over and under something, etc . . . 678
6.4.11 Delimiters . . . 679
6.5 Math versions of text commands . . . 680
6.6 Other special functions and parameters . . . 680
6.6.1 Biggggg . . . 680
6.6.2 The log-like functions . . . 681
6.6.3 Parameters . . . 681

7 Default cfg files **681**

31 preload.dtx **683**

1 Overview **683**

2 Customization **683**

3 Module switches for the DOCSTRIP program **683**

4 A driver for this document **684**

5 The code **684**

32 ltfntcmd.dtx **686**

1 Introduction **686**

2 The implementation **688**

3 Initialization **694**

33 lttextcomp.dtx 695

1 Sub-encodings 699
1.1 Unavailable in sub-encoding 1 and higher (drop symbols not working in Latin Modern) . . . 700
1.2 Unavailable in sub-encoding 2 (majority of new OTF fonts via autoinst) and higher . . . 700
1.3 Unavailable in sub-encoding 3 and higher . . . 704
1.4 Unavailable in sub-encoding 4 and higher . . . 704
1.5 Unavailable in sub-encoding 5 (most older PS fonts) and higher . . . 704
1.6 Unavailable in sub-encoding 6 and higher . . . 705
1.7 Unavailable in sub-encoding 7 and higher . . . 705
1.8 Unavailable in sub-encoding 8 and higher . . . 705
1.9 Unavailable in Sub-encoding 9 (most missing) . . . 705

2 Unicode engine specials 706

3 Font family sub-encodings setup 706

4 Legacy symbol support for lists and footnote symbols 711

5 The textcomp package 716
5.1 The old textcomp package code . . . 717
5.1.1 Supporting oldstyle digits . . . 725
5.1.2 Subset encoding defaults . . . 725

6 The checkencodingsubset.tex file 727

34 ltpageno.dtx 737

1 Page Numbering 737

35 ltxref.dtx 738

1 Cross Referencing 738
1.1 Cross Referencing . . . 738

36 ltproperties.dtx 746

1 Introduction 746

2 Design discussion 746

3 Handling unknown labels and properties 747

4 Rerun messages 747

5 Open points 747

6 Code interfaces **747**

7 Auxiliary file interfaces **749**

8 LaTeX 2_ε interface **749**

9 Pre-declared properties **750**

10 The Implementation **751**
10.1 Reference commands 753
10.2 Tests and warnings 755
10.3 Predeclared properties 757
10.4 Messages 759

37 ltmiscen.dtx 760

1 Miscellaneous Environments **760**
1.1 Environments 760
1.2 Center, Flushright, Flushleft 774
1.3 Verbatim 777

38 ltmath.dtx 785

1 Math setup **785**
1.1 Math commands based on plain TeX 785
1.1.1 The log-like functions 785
1.1.2 Biggggg 786
1.1.3 The UNSORTED Rest 786
1.2 Math Environments 793
1.3 External options to the standard document classes 798
1.3.1 Left equation numbering 798
1.3.2 Flush left equations 798

39 ltlists.dtx 801

1 List, and related environments **801**
1.1 List and Trivlist 802
1.2 Vertical Spacing (skips) 803
1.3 Penalties 803
1.4 Horizontal Spacing (dimens) 803
1.5 Default Values 803
1.6 Itemize and Enumerate 816

40 ltboxes.dtx 818

1 LaTeX Box commands **818**
1.1 Some low-level constructs 836

41 lttab.dtx 837

1 Tabbing, Tabular and Array Environments 837
1.1 tabbing 837
1.2 array and tabular environments 846

42 ltpictur.dtx 862

1 Picture Mode 862
1.1 Curves 889

43 ltthm.dtx 894

1 Theorem Environments 894

44 ltsect.dtx 898

1 Sectioning Commands 898
1.1 The Title 898
1.2 Sectioning 899
1.2.1 Initializations 906
1.3 Table of Contents etc. 906
1.3.1 Convention 906
1.3.2 Commands 906

45 ltfloat.dtx 911

1 Floats 911
1.1 Floating Environments 911
1.2 Footnotes 925

46 ltidxglo.dtx 934

1 Index and Glossary Generation 934

47 ltbibl.dtx 937

1 Bibliography Generation 937
1.1 Default definitions 941

48 ltmarks.dtx 942

1 Introduction 942

2 Design-level and code-level interfaces **943**
2.1 Use cases for conditionals . . . 945
2.2 Debugging mark code . . . 946

3 Application examples **946**

4 Legacy LaTeX 2ε interface **946**
4.1 Legacy design-level and document-level interfaces . . . 946
4.2 Legacy interface extensions . . . 947

5 Notes on the mechanism **947**

6 Internal output routine functions **949**

7 The Implementation **950**
7.1 Allocating new mark classes . . . 950
7.2 Updating mark structures . . . 952
7.3 Placing and retrieving marks . . . 959
7.4 Comparing mark values . . . 961
7.5 Messages . . . 962
7.6 Debugging the mark structures . . . 962
7.7 Designer-level interfaces . . . 964

8 LaTeX 2ε integration **965**
8.1 Core LaTeX 2ε integration . . . 965
8.2 Other LaTeX 2ε output routines . . . 968
8.3 Rollback information . . . 968

49 ltpage.dtx **969**

1 Page styles and related commands **969**
1.1 Page Style Commands . . . 969
1.2 How a page style makes running heads and feet . . . 969
1.3 marking conventions . . . 969

50 ltclass.dtx **974**

1 Introduction **974**

2 User interface **974**
2.1 Option processing . . . 975

3 Class and Package interface **976**
3.1 Class name and version . . . 976
3.2 Package name and version . . . 976
3.3 Requiring other packages . . . 976
3.4 Declaring new options . . . 977
3.5 Safe Input Macros . . . 978

4 Implementation **978**
4.1 Hooks . . . 1007
4.2 Providing shipment . . . 1010

5 Package/class rollback mechanism **1018**

6 After Preamble **1026**

51 ltkeys.dtx 1027

1 Creating and using keyval options **1027**
1.1 Implementation of ltkeys . . . 1028
1.2 Key properties . . . 1028
1.3 Main mechanism . . . 1029
1.4 The document interfaces . . . 1033
1.5 Option usage scope . . . 1034
1.6 General key setting . . . 1035

52 ltfilehook.dtx 1036

1 Introduction **1036**
1.1 Provided hooks . . . 1036
1.2 General hooks for file reading . . . 1036
1.3 Hooks for package and class files . . . 1037
1.4 Hooks for `\include` files . . . 1038
1.5 High-level interfaces for LaTeX . . . 1039
1.6 Kernel, class, and package interfaces for LaTeX . . . 1040
1.7 A sample package for structuring the log output . . . 1040

2 The Implementation **1041**
2.1 Document and package-level commands . . . 1041
2.2 expl3 helpers . . . 1042
2.3 Declaring the file-related hooks . . . 1045
2.4 Patching LaTeX's `\InputIfFileExists` command . . . 1045
2.5 Declaring a file substitution . . . 1047
2.6 Selecting a file (`\set@curr@file`) . . . 1049
2.7 Replacing a file and detecting loops . . . 1052
2.7.1 The Tortoise and Hare algorithm . . . 1053
2.8 Preventing a package from loading . . . 1055
2.9 High-level interfaces for LaTeX . . . 1056
2.10 Internal commands needed elsewhere . . . 1056

3 A sample package for structuring the log output **1057**

4 Package emulations **1058**
4.1 Package atveryend emulation . . . 1058

53 ltshipout.dtx 1059

1 Introduction **1059**
1.1 Overloading the \shipout primitive . . . 1059
1.2 Provided hooks . . . 1060
1.3 Legacy LaTeX commands . . . 1062
1.4 Special commands for use inside the hooks . . . 1063
1.5 Provided LuaTeX callbacks . . . 1063
1.6 Information counters . . . 1064
1.7 Debugging shipout code . . . 1064

2 Emulating commands from other packages **1065**
2.1 Emulating atbegshi . . . 1065
2.2 Emulating everyshi . . . 1066
2.3 Emulating atenddvi . . . 1066
2.4 Emulating everypage . . . 1066

3 The Implementation **1067**
3.1 Debugging . . . 1067
3.2 Handling the end of job hook . . . 1079

4 Legacy LaTeX 2_ε interfaces **1082**

5 Internal commands needed elsewhere **1083**

6 Package emulation for compatibility **1084**
6.1 Package atenddvi emulation . . . 1084
6.2 Package atbegshi emulation . . . 1085
6.3 Package everyshi emulation . . . 1086

54 ltoutput.dtx **1087**

1 Output Routine **1087**
1.1 Floats . . . 1087
1.1.1 Kludgeins . . . 1143
1.1.2 Float control . . . 1145
1.1.3 Float placement parameters . . . 1158

55 lttagging.dtx **1161**

1 General support for tagged output **1161**

2 Implementation **1162**
2.1 Tagging sockets . . . 1163
2.1.1 Tagging support for paragraph setup . . . 1163
2.1.2 Tagging socket for targets . . . 1164
2.1.3 Tagging sockets for toc . . . 1164
2.1.4 Tagging support for table/tabular packages . . . 1164
2.1.5 Tagging Support for floats . . . 1166

3 For lttab.dtx parked here for now **1166**
3.1 Variables for row, column and span counting 1166
3.2 Tracing/debugging 1167
3.3 Interface commands 1167

56 lthyphen.dtx **1172**

57 ltfinal.dtx **1174**

1 Final settings **1174**
1.1 Debugging 1174
1.2 Typesetting parameters 1174
1.3 Lccodes for hyphenation 1178
1.4 Hyphenation 1180
1.5 Font loading 1181
1.6 Input encoding 1182
1.7 Lccodes and uccodes 1187
1.8 Applying Patch files 1190
1.9 Freeing Memory 1191
1.10 Initialise file list 1191
1.11 Preparation for supporting PDF in backends 1192
1.12 Do some temporary work for pre-release 1192
1.13 Some last minute initializations 1192
1.14 Dumping the format 1192

Change History **1193**

Index **1271**

File 01
ltdirchk.dtx

1 LaTeX System Dependent Initializations

This file implements the semi-automatic determination of various system dependent parts of the initialization. The actual definitions may be placed in a file `texsys.cfg`. Thus for operating systems for which the tests here do not result in acceptable settings, a 'hand written' `texsys.cfg` may be produced.

The macros that must be defined are:

`\@currdir` `\@currdir`⟨*filename*⟩⟨*space*⟩ should expand to a form of the filename that uniquely refers to the 'current directory' if this is possible. (The expansion should also end with a space.) on UNIX, this is `\def\@currdir{./}`. For more exotic operating systems you may want to make `\@currdir` a macro with arguments delimited by `.` and/or ⟨*space*⟩. If the operating system has no concept of directory structure, this macro should be defined to be empty.

`\input@path` If the primitive `\openin` searches the same directories as the primitive `\input`, then it is possible to tell (using `\ifeof`) whether a file exists before trying to input it. For systems like this, `\input@path` should be left undefined.

If `\openin` does not 'follow' `\input` then `\input@path` must be defined to be a list of directories to search for input files. The format for each directory is as for `\@currdir`, normally just a prefix is required, but it may be a macro with space-delimited argument. That is, if ⟨*dir*⟩ is an entry in the input path, TeX will try to load the expansion of ⟨*dir*⟩⟨*filename*⟩⟨*space*⟩

So either ⟨*dir*⟩ should be defined as a macro with argument delimited by space, or it should just expand to a directory name, including the final directory separator, so that it may be concatenated with the ⟨*filename*⟩. This means that for UNIX-like syntax, each ⟨*dir*⟩ should end with a slash, `/`.

`\input@path` should expand to a list of such directories, each in a `{}` group.

`\filename@parse` After a call of the form: `\filename@parse{`⟨*filename*⟩`}`, the three macros `\filename@area`, `\filename@base` and `\filename@ext` should be defined to be the 'area' (or directory), basename and extension respectively. If there was no extension specified in ⟨*filename*⟩, `\filename@ext` should be `\let` to `\relax` (so this case may be tested with `\@ifundefined{filename@ext}` and, perhaps a default extension substituted).

Normally one would not need to define this macro in `texsys.cfg` as the automatic tests can supply parsers that work with UNIX and VMS and Macintosh syntax, as well as a basic parser that will cover many other cases. However some operating systems may need a 'hand produced' parser in which case it should be defined in this file.

The UNIX parser also works for most MSDOS TeX versions. Currently if the UNIX, VMS or Macintosh parser is not used, `\filename@parse` is defined to always return an empty area, and to split the argument into basename and extension at the first '`.`' that occurs in the name. Parsers for other formats may be defined in `texsys.cfg`, in which case they will be used in preference to the default definitions.

`\@TeXversion` `\@TeXversion` is now set automatically by the initialization tests in this file. You should not need to set it in `texsys.cfg`, however the following documentation is left for information. LaTeX does not set this variable exactly, the automatic tests set it to:

`2` for any version, v, $v < 3.0$

`3` for any version, v, $3.0 \leq v \leq 3.14$

⟨*undefined*⟩ otherwise.
However these values are accurate enough for LaTeX to take appropriate action for these old TeXs.

If your TeX is older than version 3.141, then you should define `\@TeXversion` (using `\def`) to be the version number. If you do not do this[1] , LaTeX will not work around a bug in old TeX versions, and so error messages will appear in a very strange format, with `^^J` appearing instead of line breaks:

```
 LaTeX Error: \rubbish undefined.^^J^^JSee the LaTeX manual or LaTeX=
Companion
for explanation.^^JType  H <return>  for immediate help.
...

.3 \renewcommand{\rubbish}
                          {}
```

However if you put `\def\@TeXversion{3.14}` in `texsys.cfg` the following format will be used:

```
 LaTeX Error: \rubbish undefined.

ee the LaTeX manual or LaTeX Companion for explanation.
ype  H <return>  for immediate help.
 .
...

.3 \renewcommand{\rubbish}
                          {}
```

Note that this has an extra line `! .` which does not appear in error messages that use the default settings with a current version of TeX, but this should not cause any confusion we hope.

2 Initialization

As this file is read at a very early stage, some definitions that are normally considered to be part of the format must be made here.

Most such definitions are repeated later in the "right" place, usually (but not always) with different implementations. To be able to spot this more easily if you look into the file `latex.ltx` (which is stripped of comments) we add some comment lines to that effect that survive the stripping process by `docstrip`.

```
⟨*dircheck⟩
%% ---- START temporary definitions for bootstrapping; later overwritten ----
⟨/dircheck⟩
```

[1] Actually if your TeX is really old, version 2, LaTeX can detect this, and sets `\@TeXversion` to 2 if it is not set in the `cfg` file.

2.1 INITEX

```
⟨*dircheck⟩
⟨*initex⟩
⟨initex⟩\ifnum\catcode`\{=1
⟨initex⟩   \errmessage
⟨initex⟩   {LaTeX must be made using an initex with no format preloaded}
⟨initex⟩\fi
\catcode`\{=1
\catcode`\}=2
```

If LuaTEX is in use the extensions and other new primitives have to be activated: this is done as early as possible. Older versions of LuaTEX do not hide the primitives: a version check is not needed as the version itself will be missing in the case where action is needed!

```
\ifx\directlua\undefined
\else
  \ifx\luatexversion\undefined
```

Enable e-TeX/pdfTeX/Umath primitives with their natural names

```
    \directlua{tex.enableprimitives("",%
                  tex.extraprimitives('etex', 'pdftex', 'umath'))}
```

In current formats enable primitives with unprefixed names. the latexrelease guards allow the primitives to be defined with a `\luatex` prefix if older formats are specified.

The unprefixed forms are *not* undefined for improved compatibility with external packages when rolling back the format.

```
⟨/initex⟩
⟨/dircheck⟩
⟨*initex, latexrelease⟩
⟨latexrelease⟩\ifx\directlua\undefined\else
⟨latexrelease⟩\IncludeInRelease{2015/10/01}{\luatexluafunction}
⟨latexrelease⟩                              {LuaTeX (prefixed names)}%
    \directlua{tex.enableprimitives("",%
                  tex.extraprimitives("omega", "aleph", "luatex"))}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\luatexluafunction}
⟨latexrelease⟩                              {LuaTeX (prefixed names)}%
⟨latexrelease⟩\directlua{
⟨latexrelease⟩  tex.enableprimitives(
⟨latexrelease⟩    "luatex",
⟨latexrelease⟩    tex.extraprimitives("core","omega", "aleph", "luatex")
⟨latexrelease⟩  )
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\fi
⟨/initex, latexrelease⟩
⟨*dircheck⟩
⟨*initex⟩
  \fi
\fi
```

A test can now be made for eTEX.

```
⟨initex⟩\ifx\eTeXversion\undefined
⟨initex⟩   \errmessage
⟨initex⟩     {LaTeX requires e-TeX}
⟨initex⟩   \expandafter\endinput
```

```
⟨initex⟩\fi
```

That distraction over, back to the basics of a format.

```
\catcode`\#=6
\catcode`\^=7
\chardef\active=13
\catcode`\@=11
\countdef\count@=255
\let\bgroup={ \let\egroup=}
\ifx\@@input\@undefined\let\@@input\input\fi
\ifx\@@end\@undefined\let\@@end\end\fi
\chardef\@inputcheck0
\chardef\sixt@@n=16
\newlinechar`\^^J
\def\typeout{\immediate\write17}
\def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
  \do\#\do\^\do\_\do\%\do\~}
\def\@makeother#1{\catcode`#1=12\relax}
\def\space{ }
\def\@tempswafalse{\let\if@tempswa\iffalse}
\def\@tempswatrue{\let\if@tempswa\iftrue}
\let\if@tempswa\iffalse
\def\loop#1\repeat{\def\iterate{#1\relax\expandafter\iterate\fi}%
  \iterate \let\iterate\relax}
\let\repeat\fi
⟨/initex⟩
```

2.2 Some bits of 2e

```
⟨*2ekernel⟩
\def\two@digits#1{\ifnum#1<10 0\fi\number#1}
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}
```

This is a special version of `\ProvidesFile` for initex use.

```
\def\ProvidesFile#1{%
  \begingroup
    \catcode`\ 10 %
    \ifnum \endlinechar<256 %
      \ifnum \endlinechar>\m@ne
        \catcode\endlinechar 10 %
      \fi
    \fi
    \@makeother\/%
    \@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}}
\def\@providesfile#1[#2]{%
    \wlog{File: #1 #2}%
    \@addtofilelist{ #2}%
    \endgroup}
\long\def\@addtofilelist#1{}
```

```
\def\@empty{}
\catcode`\%=12
\def\@percentchar{%}
\catcode`\%=14
\let\@currdir\@undefined
\let\input@path\@undefined
```

```
\let\filename@parse\@undefined
```

\strip@prefix

```
\def\strip@prefix#1>{}
⟨/2ekernel⟩
```

(*End of definition for* \strip@prefix.)

3 texsys.cfg

As mentioned above, any site specific definitions required to describe the filename handling must be entered into a file texsys.cfg. If texsys.cfg can not be located by \openin, we write a default version out. The default version only contains comments, so we do not actually input the file in that case. The automatic tests later will, hopefully, correctly define the required macros.

The tricky code below checks to see if texsys.cfg exists. If it does not, all the text in this file between START and END is copied verbatim to a new file texsys,cfg. If texsys.cfg is found, then it is simply input. This is only done when this file is being used unstripped.

```
⟨*docstrip⟩
\openin15=texsys.cfg
\ifeof15
\typeout{** Writing a default texsys.cfg}
\immediate\openout15=texsys.cfg
\begingroup
\catcode`\^^M\active%
\let^^M\par%
\def\reserved@a#1^^M{%
 \def\reserved@b{#1}%
 \ifx\reserved@b\reserved@c\endgroup\else%
     \immediate\write15{#1}%
     \expandafter\reserved@a\fi}%
\def\reserved@d#1START^^M{\let\do\@makeother\dospecials\reserved@a}%
\catcode`\%=12
\def\reserved@c{%END}
\reserved@d
```

START

3.1 texsys.cfg

This file contains the site specific definitions of the four macros \@currdir, \input@path, \filename@parse and \@TeXversion.

As distributed it only contains comments, however this 'empty' file will work on many systems because of the automatic tests built into ltdirchk.dtx. You *are* allowed to edit this file to add definitions of these macros appropriate to your system.

The macros that must be defined are:

\@currdir

\@currdir⟨*filename*⟩⟨*space*⟩ should expand to a form of the filename that uniquely refers to the 'current directory' if this is possible. (The expansion should also end with a space.) on UNIX, this is \def\@currdir{./}. For more exotic operating systems you may want to make \@currdir a macro with arguments delimited by . and/or ⟨*space*⟩. If

the operating system has no concept of directory structure, this macro should be defined to be empty.

`\input@path` If the primitive `\openin` searches the same directories as the primitive `\input`, then it is possible to tell (using `\ifeof`) whether a file exists before trying to input it. For systems like this, `\input@path` should be left undefined.

If `\openin` does not 'follow' `\input` then `\input@path` must be defined to be a list of directories to search for input files. The format for each directory is as for `\@currdir`, normally just a prefix is required, but it may be a macro with space-delimited argument. That is, if ⟨*dir*⟩ is an entry in the input path, TEXwill try to load the expansion of

⟨*dir*⟩⟨*filename*⟩⟨*space*⟩

So either ⟨*dir*⟩ should be defined as a macro with argument delimited by space, or it should just expand to a directory name, including the final directory separator, so that it may be concatenated with the ⟨*filename*⟩. This means that for UNIX-like syntax, each ⟨*dir*⟩ should end with a slash, `/`. One exception to this rule is that the input path should *always* contain the empty directory `{}` as this will allow 'full pathnames' to be used, and the 'current directory' to be searched.

`\input@path` should expand to a list of such directories, each in a `{}` group.

`\filename@parse` After a call of the form: `\filename@parse{`⟨*filename*⟩`}`, the three macros `\filename@area`, `\filename@base`, `\filename@ext` should be defined to be the 'area' (or directory), basename and extension respectively. If there was no extension specified in ⟨*filename*⟩, `\filename@ext` should be `\let` to `\relax` (so this case may be tested with `\@ifundefined{filename@ext}` and, perhaps a default extension substituted).

Normally one would not need to define this macro in `texsys.cfg` as the automatic tests can supply parsers that work with UNIX and VMS syntax, as well as a basic parser that will cover many other cases. However some operating systems may need a 'hand produced' parser in which case it should be defined in this file.

The UNIX parser also works for most MSDOS TEX versions. Currently if the UNIX or VMS parser is not used, `\filename@parse` is defined to always return an empty area, and to split the argument into basename and extension at the first '`.`' that occurs in the name. Parsers for other formats may be defined in `texsys.cfg`, in which case they will be used in preference to the default definitions.

`\@TeXversion` You should not need to set this macro in `texsys.cfg`. LATEX tests to set this automatically. See the comments in the opening section of `ltdirchk.dtx`.

The following sections give examples of definitions which might work on various systems. These are currently mainly untested as I only have access to a few systems, all of which do not need this file as the automatic tests work. All the code is commented out.

3.2 UNIX (web2c)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
%\def\@currdir{./}
%\let\input@path\@undefined
```

3.3 UNIX (other)

Apparently some commercial UNIX implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever directories are used at your site): note that the directory names should end with `/`.

```
% \def\@currdir{./}
% \def\input@path{%
%   {/usr/local/lib/tex/inputs/distrib/}%
%   {/usr/local/lib/tex/inputs/contrib/}%
%   {/usr/local/lib/tex/inputs/local/}%
% }
```

3.4 MSDOS (emtex)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
% \def\@currdir{./}
% \let\input@path\@undefined
```

3.5 MSDOS (other)

Some PC implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever directories are used at your site): note that the directory names should end with `/`. This assumes the implementation uses UNIX style `/` as the directory separator.

```
% \def\@currdir{./}
% \def\input@path{%
%   {c:/tex/inputs/distrib/}%
%   {c:/tex/inputs/contrib/}%
%   {c:/tex/inputs/local/}%
% }
```

3.6 VMS (DECUS TEX, PD VMS 3.6)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
% \def\@currdir{[]}
% \let\input@path\@undefined
```

3.7 VMS (???)

Some VMS implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following:

```
% \def\@currdir{[]}
% \def\input@path{%
%   {tex_inputs:}%
%   {SOMEDISK:[SOME.TEX.DIRECTORY]}%
% }
```

3.8 MACINTOSH (OzTeX 1.6)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
% \def\@currdir{:}
% \let\input@path\@undefined
```

3.9 MACINTOSH (other)

Some Macintosh implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever folders are used on your machine): note that the directory names should end with `:`, and they should contain *no* spaces.

```
% \def\@currdir{:}
% \def\input@path{%
%   {Hard-Disk:Applications:TeX:TeX-inputs:}%
%   {Hard-Disk:Applications:TeX:My-inputs:}%
% }
```

3.10 FAKE EXAMPLE

This example is for an operating system that has filenames of the form `<area>name` For maximum compatibility with macro sets, you want `name.ext` to be mapped to `<ext>name`. and `<area>name.ext` to be mapped to `<area.ext>name`. `\input` does this mapping automatically, but `\openin` does not, and does not look in the same places as `\input`. `<>name` is the desired 'current directory' syntax.

the following code would possibly work:

```
% \def\@dir#1#2 {%
%   \@d@r{#1}#2..\@nil}
% \def\@d@r#1#2.#3.#4\@nil{%
%   <\ifx\@dir#1\@dir\else#1\ifx\@dir#3\@dir\else.\fi\fi#3>#2 }
%
% \def\@currdir{\@dir{}}
% \def\input@path{%
%   {\@dir{area.one}}%
%   {\@dir{area.two}}%
% }
```

END

```
\immediate\closeout15
```

If `texsys.cfg` did exist, then input it.

```
\else
\typeout{** Using the existing texsys.cfg}
\closein15
\input texsys.cfg
\fi
⟨/docstrip⟩
```

If the stripped version of this file is being used (in latex2e.ltx) then texsys.cfg should be there, so just input it.

```
⟨dircheck⟩\input texsys.cfg
```

4 Setting \@currdir

\@currdir
\IfFileExists

This is a local definition of \IfFileExists. It tries to relocate texsxys.aux. If it succeeds, then the \@currdir syntax has been determined. If all the tests fail then \@currdir will be set to \@empty, and ltxcheck will warn of this when it checks the format.

```
\begingroup
\count@\time
\divide\count@ 60
\count2=-\count@
\multiply\count2 60
\advance\count2 \time
```

The current date and time stamp.

\today

```
\edef\today{%
  \the\year/\two@digits{\the\month}/\two@digits{\the\day}:%
    \two@digits{\the\count@}:\two@digits{\the\count2}}
```

Create a file texsys.aux (hopefully in the current directory), then try to locate it again.

```
\immediate\openout15=texsys.aux
\immediate\write15{\today^^J}
\immediate\closeout15 %
```

#1 is the file to try, #2 is what to do on success, #3 on failure. Note that this definition is overwritten later on again!

```
\def\IfFileExists#1#2#3{%
  \openin\@inputcheck#1 %
  \ifeof\@inputcheck
     #3\relax
  \else
    \read\@inputcheck to \reserved@a
    \ifx\reserved@a\today
      \typeout{#1 found}#2\relax
    \else
      \typeout{BAD: old file \reserved@a (should be \today)}%
      #3\relax
    \fi
  \fi
  \closein\@inputcheck}
```

```
\endlinechar=-1
```

If \@currdir has not been pre-defined in texsys.cfg then test for UNIX, VMS and Oz-TEX-Mac. syntax.

```
\ifx\@currdir\@undefined
  \IfFileExists{./texsys.aux}{\gdef\@currdir{./}}%
    {\IfFileExists{[]texsys.aux}{\gdef\@currdir{[]}}%
      {\IfFileExists{:texsys.aux}{\gdef\@currdir{:}}{}}}
```

If it is still undefined at this point, all the above tests failed. Earlier versions interactively prompted for a definition at this point, but it seems impossible to reliably obtain information from users at this point in the installation. This version of the file produces

a format with no user-interaction. Later if the format is not suitable for the system, `texsys.cfg` may be edited and the format re-made.

```
\ifx\@currdir\@undefined
  \global\let\@currdir\@empty
  \typeout{^^J^^J%
    !! No syntax for the current directory could be found^^J%
    }%
\fi
```

Otherwise `\@currdir` was defined in `texsys.cfg`. In this case check that the syntax specified works on this system. (In case a complete LaTeX system has been copied from one system to another.) If the test fails, give up. The installer should remove or correct the offending `texsys.cfg` and try again.

```
\else
  \IfFileExists{\@currdir texsys.aux}{}{%
    \edef\reserved@a{\errhelp{%
      texsys.cfg specifies the current directory syntax to be^^J%
      \meaning\@currdir^^J%
      but this does not work on this system.^^J%
      Remove texsys.cfg and restart.}}\reserved@a
    \errmessage{Bad texsys.cfg file: \noexpand\@currdir}\@@end}
```

The version of `\@currdir` in `texsys.cfg` looks OK.

```
\fi

\immediate\closeout15 %
\endgroup

\typeout{^^J^^J%
         \noexpand\@currdir set to:
           \expandafter\strip@prefix\meaning\@currdir.^^J%
         }
```

(*End of definition for* `\@currdir`, `\IfFileExists`, *and* `\today`.)

Stop here if the file is being used unstripped.

```
⟨*docstrip⟩
\relax\endinput
⟨/docstrip⟩
```

5 Setting `\input@path`

Earlier versions of this file attempted to automatically test whether `\input@path` was required, and interactively prompt for a path if necessary. This was not found to be very reliable The first-time installer of LaTeX 2_ε can not be expected to have enough information to supply the correct information to the prompts. Now the interaction is omitted. After the format is made the installer can attempt to run the test document `ltxcheck.tex` through LaTeX 2_ε. This will check, among other things, whether `texsys.cfg` will need to be edited and the format remade.

`\input@path` Now set up the `\input@path`.

`\input@path` should either be undefined, or a list of directories as described in the introduction.

```
\typeout{^^J%
```

```
    Assuming \noexpand\openin and \noexpand\input^^J%
    \ifx\input@path\@undefined
```

`\input@path` has not been pre-defined.

```
      have the same search path.^^J%
    \else
```

`\input@path` has been defined in `texsys.cfg`.

```
      have different  search paths.^^J%
      LaTeX will use the path specified by \noexpand\input@path:^^J%
    \fi
    }
```

(*End of definition for* `\input@path`.)

6 Filename Parsing

`\filename@parse` Split a filename into its components.

```
\ifx\filename@parse\@undefined
  \def\reserved@a{./}\ifx\@currdir\reserved@a
```

`\filename@parse` was not specified in `texsys.cfg`, but `\@currdir` looks like UNIX...

```
    \typeout{^^JDefining UNIX/DOS style filename parser.^^J}
    \def\filename@parse#1{%
      \let\filename@area\@empty
      \expandafter\filename@path#1/\\}
```

Search for the last `/`.

```
    \def\filename@path#1/#2\\{%
      \ifx\\#2\\%
         \def\reserved@a{\filename@simple#1.\\}%
      \else
         \edef\filename@area{\filename@area#1/}%
         \def\reserved@a{\filename@path#2\\}%
      \fi
      \reserved@a}
```

```
  \else\def\reserved@a{[]}\ifx\@currdir\reserved@a
```

`\filename@parse` was not specified in `texsys.cfg`, but `\@currdir` looks like VMS...

```
    \typeout{^^JDefining VMS style filename parser.^^J}
    \def\filename@parse#1{%
      \let\filename@area\@empty
      \expandafter\filename@path#1]\\}
```

Search for the last `]`.

```
    \def\filename@path#1]#2\\{%
      \ifx\\#2\\%
         \def\reserved@a{\filename@simple#1.\\}%
      \else
         \edef\filename@area{\filename@area#1]}%
         \def\reserved@a{\filename@path#2\\}%
      \fi
      \reserved@a}
```

```
  \else\def\reserved@a{:}\ifx\@currdir\reserved@a
```

`\filename@parse` was not specified in `texsys.cfg`, but `\@currdir` looks like Macintosh...

```
    \typeout{^^JDefining Mac style filename parser.^^J}
    \def\filename@parse#1{%
      \let\filename@area\@empty
      \expandafter\filename@path#1:\\}
```

Search for the last `:`.

```
    \def\filename@path#1:#2\\{%
      \ifx\\#2\\%
         \def\reserved@a{\filename@simple#1.\\}%
      \else
         \edef\filename@area{\filename@area#1:}%
         \def\reserved@a{\filename@path#2\\}%
      \fi
      \reserved@a}
  \else
```

`\filename@parse` was not specified in `texsys.cfg`. So just make a simple parser that always sets `\filename@area` to empty.

```
    \typeout{^^JDefining generic filename parser.^^J}
    \def\filename@parse#1{%
      \let\filename@area\@empty
      \expandafter\filename@simple#1.\\}
  \fi\fi\fi
```

`\filename@simple` is used by all three versions. Finally we can split off the extension.

```
⟨/dircheck⟩
⟨*dircheck, latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}{\filename@simple}
⟨latexrelease⟩                              {Final dot for extension}%
\def\filename@simple#1.#2\\{%
  \ifx\\#2\\%
    \let\filename@ext\relax
    \edef\filename@base{#1}%
  \else
    \filename@dots{#1}#2\\%
  \fi}
\def\filename@dots#1#2.#3\\{%
  \ifx\\#3\\%
    \def\filename@ext{#2}%
    \edef\filename@base{#1}%
  \else
    \filename@dots{#1.#2}#3\\%
  \fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\filename@simple}
⟨latexrelease⟩                              {Final dot for extension}%
⟨latexrelease⟩   \def\filename@simple#1.#2\\{%
⟨latexrelease⟩     \ifx\\#2\\%
⟨latexrelease⟩        \let\filename@ext\relax
⟨latexrelease⟩     \else
⟨latexrelease⟩        \edef\filename@ext{\filename@dot#2\\}%
```

```
⟨latexrelease⟩      \fi
⟨latexrelease⟩      \edef\filename@base{#1}}
⟨latexrelease⟩ \EndIncludeInRelease
⟨/dircheck, latexrelease⟩
⟨*dircheck⟩
```

Remove a final dot, added earlier.

```
  \def\filename@dot#1.\\{#1}
\else
```

Otherwise, `\filename@parse` was specified in `texsys.cfg`.

```
  \typeout{^^J^^J%
    \noexpand\filename@parse was defined in texsys.cfg:^^J%
    \expandafter\strip@prefix\meaning\filename@parse.^^J%
    }
\fi
```

(*End of definition for* `\filename@parse`.)

7 TEX Versions

`\@TeXversion` TEX versions older than 3.141 require `\@TeXversion` to be set. This can be determined automatically due to a trick suggested by Bernd Raichle. Actually this will not always get the correct version number, e.g., TEX3.14 would be detected as TEX3, but LATEX only needs to take account of TEX's older than 3, or between 3 and 3.14.

```
\ifx\@TeXversion\@undefined
  \ifx\@undefined\inputlineno
    \def\@TeXversion{2}
  \else
   {\catcode`\^^J=\active
     \def\reserved@a#1#2\@@{\if#1\string^3\fi}
     \edef\reserved@a{\expandafter\reserved@a\string^^J\@@}
     \ifx\reserved@a\@empty\else\gdef\@TeXversion{3}\fi}
  \fi
\fi
```

(*End of definition for* `\@TeXversion`.)

```
%% ---- END temporary definitions for bootstrapping ----
⟨/dircheck⟩
```

8 ltxcheck.tex

After the format has been made, and article.cls moved with the other files to the 'standard input directory' as specified in `install.txt`, the format may be checked by running the file `ltxcheck.tex`.

File 02
ltplain.dtx

1 Plain TEX

LATEX includes almost all of the functionality of Knuth's original 'Basic Macros' That is, the plain TEX format described in Appendix B of the TEXBook. However, some of the user commands are not much use so, in order to save memory, we may remove them from the kernel into a package. Here is a list of the commands that may be removed (PROBABLY NOT COMPLETE).

```
\magstep     \magstephalf
\mathhexbox
\vglue       \vgl@
\hglue       \hgl@
```

This file is by now very small as most of it has been moved to more appropriate kernel files: it may disappear completely one day.

LATEX font definitions are done using NFSS2 so none of PLAIN's font definitions are in LATEX.

LATEX has its own tabbing environment, so PLAIN's is disabled.

LATEX uses its own output routine, so most of the plain one was removed.

```
⟨*2ekernel⟩
\catcode`\{=1 % left brace is begin-group character
\catcode`\}=2 % right brace is end-group character
\catcode`\$=3 % dollar sign is math shift
\catcode`\&=4 % ampersand is alignment tab
\catcode`\#=6 % hash mark is macro parameter character
\catcode`\^=7 % circumflex and uparrow are for superscripts
\catcode`\_=8 % underline and downarrow are for subscripts
\catcode`\^^I=10 % ascii tab is a blank space
\chardef\active=13 \catcode`\~=\active % tilde is active
\catcode`\^^L=\active \def^^L{\par}% ascii form-feed is \par
\message{catcodes,}
```

We had to define the `\catcodes` right away, before the message line, since `\message` uses the `{` and `}` characters. When INITEX (the TEX initializer) starts up, it has defined the following `\catcode` values:

`\catcode`\^^@=9 %` ascii null is ignored
`\catcode`\^^M=5 %` ascii return is end-line
`\catcode`\\=0 %` backslash is TeX escape character
`\catcode`\%=14 %` percent sign is comment character
`\catcode`\ =10 %` ascii space is blank space
`\catcode`\^^?=15 %` ascii delete is invalid
`\catcode`\A=11 ... \catcode`\Z=11 %` uppercase letters
`\catcode`\a=11 ... \catcode`\z=11 %` lowercase letters
all others are type 12 (other)

Here is a list of the characters that have been specially catcoded:

```
\def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
  \do\#\do\^\do\_\do\%\do\~}
```

(not counting ascii null, tab, linefeed, formfeed, return, delete) Each symbol in the list is preceded by , which can be defined if you want to do something to every item in the list.

We make `@` signs act like letters, temporarily, to avoid conflict between user names and internal control sequences of plain format.

```
\catcode`@=11
```

To make the plain macros more efficient in time and space, several constant values are declared here as control sequences. If they were changed, anything could happen; so they are private symbols.

`\@ne` `\tw@` `\thr@@` `\sixt@@n` `\@cclv`

Small constants are defined using `\chardef`.

```
\chardef\@ne=1
\chardef\tw@=2
\chardef\thr@@=3
\chardef\sixt@@n=16
\chardef\@cclv=255
```

(*End of definition for* `\@ne` *and others.*)

`\@cclvi` `\@m` `\@M` `\@MM`

Constants above 255 defined using `\mathchardef`.

```
\mathchardef\@cclvi=256
\mathchardef\@m=1000
\mathchardef\@M=10000
\mathchardef\@MM=20000
```

(*End of definition for* `\@cclvi` *and others.*)

Allocation of registers

Here are macros for the automatic allocation of `\count`, `\box`, `\dimen`, `\skip`, `\muskip`, and `\toks` registers, as well as `\read` and `\write` stream numbers, `\fam` codes, `\language` codes, and `\insert` numbers.

```
\message{registers,}
```

When a register is used only temporarily, it need not be allocated; grouping can be used, making the value previously in the register return after the close of the group. The main use of these macros is for registers that are defined by one macro and used by others, possibly at different nesting levels. All such registers should be defined through these macros; otherwise conflicts may occur, especially when two or more macro packages are being used at the same time.

Historical LaTeX 2.09 comments (not necessarily accurate any more):

The following counters are reserved:

0 to 9	page numbering
10	count allocation
11	dimen allocation
12	skip allocation
13	muskip allocation
14	box allocation
15	toks allocation
16	read file allocation
17	write file allocation
18	math family allocation
19	language allocation
20	insert allocation
21	the most recently allocated number

22 constant -1

End of historical LaTeX 2.09 comments.

New counters are allocated starting with 23, 24, etc. Other registers are allocated starting with 10. This leaves 0 through 9 for the user to play with safely, except that counts 0 to 9 are considered to be the page and subpage numbers (since they are displayed during output). In this scheme, `\count` 10 always contains the number of the highest-numbered counter that has been allocated, `\count` 14 the highest-numbered box, etc. Inserts are given numbers 254, 253, etc., since they require a `\count`, `\dimen`, `\skip`, and `\box` all with the same number; `\count` 20 contains the lowest-numbered insert that has been allocated. Of course, `\box`255 is reserved for `\output`; `\count`255, `\dimen`255, and `\skip`255 can be used freely.

It is recommended that macro designers always use `\global` assignments with respect to registers numbered
1, 3, 5, 7, 9,
and always non-`\global` assignments with respect to registers
0, 2, 4, 6, 8, 255.
This will prevent "save stack buildup" that might otherwise occur.

```
\count10=22 % allocates \count registers 23, 24, ...
\count11=9 % allocates \dimen registers 10, 11, ...
\count12=9 % allocates \skip registers 10, 11, ...
\count13=9 % allocates \muskip registers 10, 11, ...
\count14=9 % allocates \box registers 10, 11, ...
\count15=9 % allocates \toks registers 10, 11, ...
\count16=-1 % allocates input streams 0, 1, ...
\count17=-1 % allocates output streams 0, 1, ...
\count18=3 % allocates math families 4, 5, ...
\count19=0 % allocates \language codes 1, 2, ...
\count20=255 % allocates insertions 254, 253, ...
```

`\insc@unt` `\allocationnumber`

The insertion counter and most recent allocation.

```
\countdef\insc@unt=20
\countdef\allocationnumber=21
```

(*End of definition for* `\insc@unt` *and* `\allocationnumber`.)

`\m@ne`

The constant −1.

```
\countdef\m@ne=22 \m@ne=-1
```

(*End of definition for* `\m@ne`.)

`\wlog`

Write on log file (only)

```
\def\wlog{\immediate\write\m@ne}
```

(*End of definition for* `\wlog`.)

`\count@` `\dimen@` `\dimen@i` `\dimen@ii` `\skip@` `\toks@`

Here are abbreviations for the names of scratch registers that don't need to be allocated.

```
\countdef\count@=255
\dimendef\dimen@=0
\dimendef\dimen@i=1 % global only
\dimendef\dimen@ii=2
\skipdef\skip@=0
\toksdef\toks@=0
```

(*End of definition for* `\count@` *and others.*)

`\newcount` `\newdimen` `\newskip` `\newmuskip` `\newbox` `\newtoks` `\newread` `\newwrite` `\newfam` `\newlanguage`

Now, we define `\newcount`, `\newbox`, etc. so that you can say `\newcount\foo` and `\foo` will be defined (with `\countdef`) to be the next counter.

To find out which counter `\foo` is, you can look at `\allocationnumber`.

Since there's no `\boxdef` command, `\chardef` is used to define a `\newbox`, `\newinsert`, `\newfam`, and so on.

LaTeX change: remove `\outer` from `\newcount` and `\newdimen` (FMi) This is necessary to use `\newcount` inside `\if...` later on. Also remove from `\newskip`, `\newbox` `\newwrite` and `\newfam` (DPC) to save later redefinition.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\newcount}{Extended Allocation}%
```

```
\def\newcount {\e@alloc\count \countdef {\count10}\insc@unt\float@count}
\def\newdimen {\e@alloc\dimen \dimendef {\count11}\insc@unt\float@count}
\def\newskip  {\e@alloc\skip  \skipdef  {\count12}\insc@unt\float@count}
\def\newmuskip
           {\e@alloc\muskip\muskipdef{\count13}\m@ne\e@alloc@top}
```

For compatibility use `\chardef` in the classical range.

```
\def\newbox   {\e@alloc\box
                  {\ifnum\allocationnumber<\@cclvi
                     \expandafter\chardef
                   \else
                     \expandafter\e@alloc@chardef
                   \fi}
                                          {\count14}\insc@unt\float@count}
\def\newtoks  {\e@alloc\toks \toksdef{\count15}\m@ne\e@alloc@top}
\def\newread  {\e@alloc\read \chardef{\count16}\m@ne\sixt@@n}
```

Skip `\write18` due to its traditional use as a shell-escape.

```
\ifx\directlua\@undefined
  \def\newwrite   {\e@alloc\write \chardef{\count17}\m@ne\sixt@@n}
\else
  \def\newwrite   {\e@alloc\write
                   {\ifnum\allocationnumber=18
                     \advance\count17\@ne
                     \allocationnumber\count17 %
                    \fi
                    \global\chardef}%
                   {\count17}%
                   \m@ne
                   {128}}
\fi
```

```
\def\new@mathgroup
  {\e@alloc\mathgroup\chardef{\count18}\m@ne\e@mathgroup@top}
\let\newfam\new@mathgroup
```

```
\ifx\directlua\@undefined
  \def\newlanguage  {\e@alloc\language \chardef{\count19}\m@ne\@cclvi}
\else
  \def\newlanguage  {\e@alloc\language \chardef{\count19}\m@ne{16384}}
```

```
\fi
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\newcount}{Extended Allocation}%
⟨latexrelease⟩\def\newcount{\alloc@0\count\countdef\insc@unt}
⟨latexrelease⟩\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
⟨latexrelease⟩\def\newskip{\alloc@2\skip\skipdef\insc@unt}
⟨latexrelease⟩\def\newmuskip{\alloc@3\muskip\muskipdef\@cclvi}
⟨latexrelease⟩\def\newbox{\alloc@4\box\chardef\insc@unt}
⟨latexrelease⟩\def\newtoks{\alloc@5\toks\toksdef\@cclvi}
⟨latexrelease⟩\def\newread{\alloc@6\read\chardef\sixt@@n}
⟨latexrelease⟩\def\newwrite{\alloc@7\write\chardef\sixt@@n}
⟨latexrelease⟩\def\new@mathgroup{\alloc@8\fam\chardef\sixt@@n}
⟨latexrelease⟩\def\newlanguage{\alloc@9\language\chardef\@cclvi}
⟨latexrelease⟩\let\newfam\new@mathgroup
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\newcount` *and others.*)

`\e@alloc@chardef`
`\e@alloc@top`

The upper limit of extended registers, which leaves this number (eg `\dimen32767`) always unallocated by these macros. cf traditional `\dimen255`.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\e@alloc@chardef}{Extended Allocation}%
```

```
\ifx\directlua\@undefined
  \ifx\widowpenalties\@undefined
```

classic tex has 2^8 registers.

```
    \mathchardef\e@alloc@top=255
    \let\e@alloc@chardef\chardef
  \else
```

etex and xetex have 2^{15} registers.

```
    \mathchardef\e@alloc@top=32767
    \let\e@alloc@chardef\mathchardef
  \fi
\else
```

luatex has 2^{16} registers.

```
  \chardef\e@alloc@top=65535
  \let\e@alloc@chardef\chardef
\fi
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\e@alloc@chardef}{Extended Allocation}%
⟨latexrelease⟩\let\e@alloc@top\@undefined
⟨latexrelease⟩\let\e@alloc@chardef\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\e@alloc@chardef` *and* `\e@alloc@top`.)

`\e@mathgroup@top` The upper limit of extended math groups (`\fam`) 16 in classic TEX and e-TEX, but 256 in Unicode TeX variants.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\e@mathgroup@top}{Extended Allocation}%
```

```
\ifx\Umathcode\@undefined
```

classic and e tex have 16 fam (0–15).

```
  \chardef\e@mathgroup@top=16
\else
```

xetex and luatex have 256 fam (0–255).

```
  \chardef\e@mathgroup@top=256
\fi
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\e@mathgroup@top}{Extended Allocation}%
⟨latexrelease⟩\let\e@mathgroup@top\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\e@mathgroup@top`.)

`\e@alloc` A modified version of `\alloc@` that takes the count register rather than just the final digit of its number (assuming `\count1`x). It also has an extra argument to give the top of the extended range.

```
                #1  #2      #3     #4    #5          #6
  \e@alloc type defcmd current top extended-top newname
```

Note that if just a single allocation range is required (not omitting a range up to 255 for inserts) then -1 should be used for the first upper bound argument, `#4`.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\e@alloc}{Extended Allocation}%
```

```
\def\e@alloc#1#2#3#4#5#6{%
  \global\advance#3\@ne
  \e@ch@ck{#3}{#4}{#5}#1%
  \allocationnumber#3\relax
  \global#2#6\allocationnumber
  \wlog{\string#6=\string#1\the\allocationnumber}}%
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\e@alloc}{Extended Allocation}%
⟨latexrelease⟩\let\e@alloc\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\e@alloc`.)

`\e@ch@ck` Extended check command. If the first range is exceeded, bump to 256 (or 266 for counts) and try again, testing the extended range.

Allocate matching registers from the top of the extended range and add to `\@freelist`.

`\extrafloats`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/10/01}
⟨latexrelease⟩                  {\e@ch@ck}{Extended Allocation (checking)}%
\gdef\e@ch@ck#1#2#3#4{%
  \ifnum#1<#2\else
```

If we've reached the classical top limit, bump to 256 or 266 for counts (count 256–265 are reserved by the allocation system).

```
    \ifnum#1=#2\relax
      \global#1\@cclvi
      \ifx\count#4\global\advance#1 10 \fi
    \fi
```

Check we are below the extended limit.

```
    \ifnum#1<#3\relax
    \else
      \errmessage{No room for a new \string#4}%
    \fi
  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\e@ch@ck}{Extended Allocation (checking)}%
⟨latexrelease⟩\gdef\e@ch@ck#1#2#3#4{%
⟨latexrelease⟩  \ifnum#1<#2\else
⟨latexrelease⟩    \ifnum#1=#2\relax
⟨latexrelease⟩      #1\@cclvi
⟨latexrelease⟩      \ifx\count#4\advance#1 10 \fi
⟨latexrelease⟩    \fi
⟨latexrelease⟩    \ifnum#1<#3\relax
⟨latexrelease⟩    \else
⟨latexrelease⟩      \errmessage{No room for a new #4}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\e@ch@ck}{Extended Allocation (checking)}%
⟨latexrelease⟩\let\e@ch@ck\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\extrafloats}{Extra floats}%
\let\float@count\e@alloc@top
```

`\extrafloats`

```
\ifx\numexpr\@undefined
```

In classic TeX use `\newinsert` to allocate float boxes.

```
\def\extrafloats#1{%
\count@#1\relax
\ifnum\count@>\z@
\newinsert\reserved@a
\global\expandafter\chardef
```

```
            \csname bx@\the\allocationnumber\endcsname\allocationnumber
\@cons\@freelist{\csname bx@\the\allocationnumber\endcsname}%
\advance\count@\m@ne
\expandafter\extrafloats
\expandafter\count@
\fi
}%
\else
```

In e-tex take float boxes from the top of the extended range.

```
\def\extrafloats#1{%
\ifnum#1>\z@
\count@\numexpr\float@count-1\relax
  \ifnum\count@<266 \ch@ck0\m@ne\insert\fi
  \ch@ck0\count@\count
  \ch@ck1\count@\dimen
  \ch@ck2\count@\skip
  \ch@ck4\count@\box
\global\e@alloc@chardef\float@count\count@
\global\expandafter\e@alloc@chardef
            \csname bx@\the\float@count\endcsname\float@count
\@cons\@freelist{\csname bx@\the\float@count\endcsname}%
\expandafter\extrafloats\expandafter{\the\numexpr#1-1\expandafter}%
\fi}%
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\extrafloats}{Extra floats}%
⟨latexrelease⟩\let\float@count\@undefined
⟨latexrelease⟩\let\extrafloats\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\e@ch@ck`*,* `\extrafloats`*, and* `\extrafloats`*.*)

`\alloc@` Since `\e@alloc` was added in 2015, `\@alloc` has not been used, but was left as some legacy code calls it. However the original definition gives spurious errors once the "classic" registers run out, so it is now defined to call `\e@alloc` internally.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}
⟨latexrelease⟩                  {\alloc@}{emulate alloc@}%
\def\alloc@#1#2#3#4{\e@alloc#2#3{\count1#1}#4\float@count}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\alloc@}{emulate alloc@}%
⟨latexrelease⟩\def\alloc@#1#2#3#4#5{\global\advance\count1#1\@ne
⟨latexrelease⟩   \ch@ck#1#4#2%
⟨latexrelease⟩   \allocationnumber\count1#1%
```

```
⟨latexrelease⟩   \global#3#5\allocationnumber
⟨latexrelease⟩   \wlog{\string#5=\string#2\the\allocationnumber}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\alloc@`.)

`\newinsert`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/10/01}
⟨latexrelease⟩                  {\newinsert}{Extended \newinsert}%
\ifx\numexpr\@undefined
```

If e-TEX is not available use the original plain TEX definition of `\newinsert`.

```
\def\newinsert#1{\global\advance\insc@unt \m@ne
  \ch@ck0\insc@unt\count
  \ch@ck1\insc@unt\dimen
  \ch@ck2\insc@unt\skip
  \ch@ck4\insc@unt\box
  \allocationnumber\insc@unt
  \global\chardef#1\allocationnumber
  \wlog{\string#1=\string\insert\the\allocationnumber}}
\else
```

The highest register allowed with `\insert`.

```
\ifx\directlua\@undefined
  \chardef\e@insert@top255
\else
  \chardef\e@insert@top\e@alloc@top
\fi
```

If the classic registers are exhausted, take an insert from the free float list and use `\extrafloats` to add a new float to that list.

```
\def\newinsert#1{%
\@tempswafalse
\global\advance\insc@unt\m@ne
\ifnum\count10<\insc@unt
\ifnum\count11<\insc@unt
\ifnum\count12<\insc@unt
\ifnum\count14<\insc@unt
  \@tempswatrue
\fi\fi\fi\fi
\if@tempswa
\allocationnumber\insc@unt
\else
\global\advance\insc@unt\@ne
  \extrafloats\@ne
  \@next\@currbox\@freelist
    {\ifnum\@currbox<\e@insert@top
      \allocationnumber\@currbox
     \else
     \ch@ck0\m@ne\insert
     \fi}%
```

```
     {\ch@ck0\m@ne\insert}%
\fi
\global\chardef#1\allocationnumber
\wlog{\string#1=\string\insert\the\allocationnumber}%
}
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\newinsert}{Extended \newinsert}%
⟨latexrelease⟩\let\e@insert@top\@undefined
⟨latexrelease⟩\def\newinsert#1{\global\advance\insc@unt \m@ne
⟨latexrelease⟩  \ch@ck0\insc@unt\count
⟨latexrelease⟩  \ch@ck1\insc@unt\dimen
⟨latexrelease⟩  \ch@ck2\insc@unt\skip
⟨latexrelease⟩  \ch@ck4\insc@unt\box
⟨latexrelease⟩  \allocationnumber\insc@unt
⟨latexrelease⟩  \global\chardef#1\allocationnumber
⟨latexrelease⟩  \wlog{\string#1=\string\insert\the\allocationnumber}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\newinsert`.)

`\ch@ck`

```
\gdef\ch@ck#1#2#3{%
  \ifnum\count1#1<#2\else
    \errmessage{No room for a new #3}%
  \fi}
```

(*End of definition for* `\ch@ck`.)

`\newhelp`

```
\def\newhelp#1#2{\newtoks#1#1\expandafter{\csname#2\endcsname}}
```

(*End of definition for* `\newhelp`.)

`\@inputcheck` `\@unused`

Allocate read stream for testing and output stream that is never open an thus writes to the terminal.

```
\newread\@inputcheck
\newwrite\@unused
```

(*End of definition for* `\@inputcheck` *and* `\@unused`.)

`\maxdimen` `\hideskip`

Here are some examples of allocation.

```
\newdimen\maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
\newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow
```

(*End of definition for* `\maxdimen` *and* `\hideskip`.)

`\p@` `\z@` `\z@skip` `\voidb@x`

```
\newdimen\p@ \p@=1pt % this saves macro space and time
\newdimen\z@ \z@=0pt % can be used both for 0pt and 0
\newskip\z@skip \z@skip=0pt plus0pt minus0pt
\newbox\voidb@x % permanently void box register
```

(*End of definition for* `\p@` *and others.*)

Assign initial values to TEX's parameters

```
\message{parameters,}
```

All of TEX's numeric parameters are listed here, but the code is commented out if no special value needs to be set. INITEX makes all parameters zero except where noted.

```
\pretolerance=100
\tolerance=200 % INITEX sets this to 10000
\hbadness=1000
\vbadness=1000
\linepenalty=10
\hyphenpenalty=50
\exhyphenpenalty=50
\binoppenalty=700
\relpenalty=500
\clubpenalty=150
\widowpenalty=150
\displaywidowpenalty=50
\brokenpenalty=100
\predisplaypenalty=10000
```

```
% \postdisplaypenalty=0
% \interlinepenalty=0
% \floatingpenalty=0, set during \insert
% \outputpenalty=0, set before TeX enters \output
```

```
\doublehyphendemerits=10000
\finalhyphendemerits=5000
\adjdemerits=10000
```

```
% \looseness=0, cleared by TeX after each paragraph
% \pausing=0
% \holdinginserts=0
% \tracingonline=0
% \tracingmacros=0
% \tracingstats=0
% \tracingparagraphs=0
% \tracingpages=0
% \tracingoutput=0
```

In the past LATEX used the default value of `1` for `\tracinglostchars` because this was the best it could do. This way one would at least get a warning in the `.log` file. e-TEX improved on that and supported a value of `2` to show the warning on the terminal, so we could have changed the default when we made the e-TEX extensions required—however, we overlooked that oportunity. In 2021 this parameter was improved on again and now also accepts the value `3` (error on the terminal). This made us realize that we should change the default. Using `3` would really be the best, but for compatibility reasons we only use `2`.

```
\tracinglostchars=2
```

```
% \tracingcommands=0
% \tracingrestores=0
```

`\tracingstacklevels` For LuaTEX, the `\tracingstacklevels` functionality was implemented as a callback, so here we just define the count register to hold the value of the parameter.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\tracingstacklevels}%
⟨latexrelease⟩                  {tracingstacklevels}%
\ifx\directlua\@undefined
  % \tracingstacklevels=0 % added in 2021
\else
  \newcount\tracingstacklevels
  % Code for \tracingstacklevels defined in ltfinal.dtx
\fi
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\tracingstacklevels}%
⟨latexrelease⟩                  {tracingstacklevels}%
⟨latexrelease⟩\ifx\directlua\@undefined
⟨latexrelease⟩\else
⟨latexrelease⟩  \let\tracingstacklevels\@undefined
⟨latexrelease⟩\fi
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
```

(*End of definition for* `\tracingstacklevels`.)

```
\uchyph=1
```

```
% \lefthyphenmin=2 \righthyphenmin=3 set below
% \globaldefs=0
% \maxdeadcycles=25 % INITEX does this
% \hangafter=1 % INITEX does this, also TeX after each paragraph
% \fam=0
% \mag=1000 % INITEX does this
% \escapechar=`\\ % INITEX does this
```

```
\defaulthyphenchar=`\-
\defaultskewchar=-1
```

```
% \endlinechar=`\^^M % INITEX does this
% \newlinechar=-1     \LaTeX\ sets this in ltdefns.dtx.
```

```
\delimiterfactor=901
```

```
% \time=now % TeX does this at beginning of job
% \day=now % TeX does this at beginning of job
% \month=now % TeX does this at beginning of job
% \year=now % TeX does this at beginning of job
```

In LaTeX we don't want box information in the transcript unless we do a full tracing.

```
\showboxbreadth=-1
\showboxdepth=-1
\errorcontextlines=-1
```

```
\hfuzz=0.1pt
\vfuzz=0.1pt
\overfullrule=5pt
\maxdepth=4pt
\splitmaxdepth=\maxdimen
\boxmaxdepth=\maxdimen
```

```
% \lineskiplimit=0pt, changed by \normalbaselines
\delimitershortfall=5pt
\nulldelimiterspace=1.2pt
\scriptspace=0.5pt
% \mathsurround=0pt
% \predisplaysize=0pt, set before TeX enters $$
% \displaywidth=0pt, set before TeX enters $$
% \displayindent=0pt, set before TeX enters $$
\parindent=20pt
% \hangindent=0pt, zeroed by TeX after each paragraph
% \hoffset=0pt
% \voffset=0pt
%
% \baselineskip=0pt, changed by \normalbaselines
% \lineskip=0pt, changed by \normalbaselines
\parskip=0pt plus 1pt
\abovedisplayskip=12pt plus 3pt minus 9pt
\abovedisplayshortskip=0pt plus 3pt
\belowdisplayskip=12pt plus 3pt minus 9pt
\belowdisplayshortskip=7pt plus 3pt minus 4pt
% \leftskip=0pt
% \rightskip=0pt
\topskip=10pt
\splittopskip=10pt
% \tabskip=0pt
% \spaceskip=0pt
% \xspaceskip=0pt
\parfillskip=0pt plus 1fil
```

`\normalbaselineskip` `\normallineskip` `\normallineskiplimit`

We also define special registers that function like parameters:

```
\newskip\normalbaselineskip \normalbaselineskip=12pt
\newskip\normallineskip \normallineskip=1pt
\newdimen\normallineskiplimit \normallineskiplimit=0pt
```

(*End of definition for* `\normalbaselineskip`, `\normallineskip`, *and* `\normallineskiplimit`.)

`\interfootlinepenalty`

```
\newcount\interfootnotelinepenalty \interfootnotelinepenalty=100
```

(*End of definition for* `\interfootlinepenalty`.)

Definitions for preloaded fonts

`\magstephalf` `\magstep`

```
\def\magstephalf{1095 }
\def\magstep#1{\ifcase#1 \@m\or 1200\or 1440\or 1728\or
                2074\or 2488\fi\relax}
```

(*End of definition for* `\magstephalf` *and* `\magstep`.)

Macros for setting ordinary text

\frenchspacing
\nonfrenchspacing

```
\def\frenchspacing{\sfcode`\.\@m \sfcode`\?\@m \sfcode`\!\@m
  \sfcode`\:\@m \sfcode`\;\@m \sfcode`\,\@m}
\def\nonfrenchspacing{\sfcode`\.3000\sfcode`\?3000\sfcode`\!3000%
  \sfcode`\:2000\sfcode`\;1500\sfcode`\,1250 }
```

(*End of definition for* \frenchspacing *and* \nonfrenchspacing.)

\normalbaselines

```
\def\normalbaselines{\lineskip\normallineskip
  \baselineskip\normalbaselineskip \lineskiplimit\normallineskiplimit}
```

(*End of definition for* \normalbaselines.)

\M
\I

Save a bit of space by using \let here.

```
\def\^^M{\ } % control <return> = control <space>
\let\^^I\^^M % same for <tab>
```

(*End of definition for* \M *and* \I.)

\lq
\rq

```
\def\lq{`}
\def\rq{'}
```

(*End of definition for* \lq *and* \rq.)

\lbrack
\rbrack

```
\def\lbrack{[}
\def\rbrack{]}
```

(*End of definition for* \lbrack *and* \rbrack.)

\aa
\AA

These are not from plain.tex but they are similar to other commands found here and nowhere else, being alternate input forms for characters.

```
\def \aa {\r a}
\def \AA {\r A}
```

(*End of definition for* \aa *and* \AA.)

\endgraf
\endline

```
\let\endgraf=\par
\let\endline=\cr
```

(*End of definition for* \endgraf *and* \endline. *These functions are documented on page 405.*)

\space

```
\def\space{ }
```

(*End of definition for* \space.)

\empty

This probably ought to go altogether, but let it to the LaTeX version to save space.

```
\let\empty\@empty
```

(*End of definition for* \empty.)

\null

```
\def\null{\hbox{}}
```

(*End of definition for* `\null`.)

\bgroup
\egroup

```
\let\bgroup={
\let\egroup=}
```

(*End of definition for* `\bgroup` *and* `\egroup`.)

\obeylines
\obeyspaces

In `\obeylines`, we say `\let^^M=\obeyedline` instead of `\def^^M{\obeyedline}` since this allows, for example, `\let\obeyedline=\cr \obeylines \halign{....`

This is essentially a plain TEX trick and in its original version where you had to use to use `\let\par=\cr` not really a safe idea in LATEX. If anybody used this trick this now breaks (and one needs to use `\obeyedline` instead).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/06/01}{\obeylines}%
⟨latexrelease⟩     {Add a redirection to obeylines and obeyspaces}%
```

If the active `^^M` escapes, e.g., into a `\write` (which is effectively in a different context) then we don't want the definition from `\obeylines` but rather a simple `\par` (in fact even the primitive one, not the LATEX version `\para_end:` which is only defined later).

```
\begingroup
\catcode`\^^M=\active % these lines must end with %
  \gdef\obeylines{\catcode`\^^M\active%
    \let^^M\obeyedline%
```

The next line ending the definition is rather curious and it took me awhile to understand why rollback fails. The problem is the following: if `latexrelease` is used, then blocks of `\IncludeInRelease` ... `\EndIncludeInRelease` are bypassed at high speed by grabbing each as a delimited argument. However, in that case `^^M` is seen not as code but as line ending characters and in that mode TEX discards everything from that point onwards to the real end of the line so it works like a comment — pretty strange really (and I think due to the fact that the original pascal compiler could have some garbage showing up after the normal line ending character. Thus we really have to make sure that any closing braces is not one the same line as an `^^M`, because otherwise it would get dropped and we end with unbalanced braces and never see the `\EndIncludeInRelease` — weird. In other places it doesn't matter because we aren't using the incomplete result.

```
  }%
  \global\let^^M\par % this is in case ^^M appears in a \write
\endgroup
```

The `\obeyedline` expands by default to `\par` with whatever definition `\par` has when it is executed. It can, however, be redefined (before calling `\obeylines`!) to achieve some special effects. If you want to alter this definition when already in the scope of `\obeylines`, it has no effect (because `\let` is used above). In that case simply make another call to `\obeylines` immediately. As you are in a restricted scope all that happens is that your redefinition is applied.

\obeyedline

For the default definition we have to use `\def` not `\let` because the meaning of `\par` can change and we want to use the one that is current when `\obeylines` act.

There is a small subtlety here: in an \edef the active ^^M stayed put (because it was equal to the primitive \par), now \obeyedline expands and you get what it contains, i.e., in that case \par, into the \edef or \mark unless we use \protected on it.

```
\protected\gdef\obeyedline{\par}
```

The definition of \obeyspaces is changed in the same way and now executes \obeyedspace for each active space.

\obeyedspace

```
\global\let\obeyedspace\space
```

```
\begingroup
\catcode`\ =\active%
\gdef\obeyspaces{\catcode`\ \active\let =\obeyedspace}%
```

An active space elsewhere generates \space by default (for example in a \write).

```
\global\let =\space%
\endgroup
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\obeylines}%
⟨latexrelease⟩    {Add a redirection to obeylines and obeyspaces}%
⟨latexrelease⟩
```

From 2019 onwards the commands are made robust (somewhat later in the kernel sources). So if we roll back they are robust, so when redefining them we have get rid of the robust payload first. Otherwise that is seen by the later rollback below, which then installs a fragile version of the new definition on top of the one we roll back to here, sigh. \kernel@make@fragile also changes its definition (later own) so this is done directly.

```
⟨latexrelease⟩\expandafter\let\csname obeylines \endcsname\@undefined
⟨latexrelease⟩\expandafter\let\csname obeyspace \endcsname\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\begingroup
⟨latexrelease⟩\catcode`\^^M=\active % these lines must end with %
⟨latexrelease⟩  \gdef\obeylines{\catcode`\^^M\active \let^^M\par %
```

Closing brace on a separate line (see comment above).

```
⟨latexrelease⟩                  }%
```

Another pitfall: if we do a rollback \par is no longer the primitive, so the roll back definition needs \let to what is new the primitive.

```
⟨latexrelease⟩  \global\let^^M\RawParEnd % this is in case ^^M appears in a \write
⟨latexrelease⟩\endgroup
⟨latexrelease⟩\def\obeyspaces{\catcode`\ \active}
⟨latexrelease⟩
⟨latexrelease⟩\let\obeyedline\@undefined
⟨latexrelease⟩\let\obeyedspace\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \obeylines *and others.*)

\loop
\iterate
\repeat

We use Kabelschacht's method of doing loops, see TUB 8#2 (1987). (unless that breaks something :-). It turned out to need an extra \relax: see pr/642 (\loop could do one iteration too much in certain cases).

```
\long\def \loop #1\repeat{%
```

```
  \def\iterate{#1\relax  % Extra \relax
               \expandafter\iterate\fi
               }%
  \iterate
  \let\iterate\relax
}
```

This setting of `\repeat` is needed to make `\loop...\if...\repeat` skippable within another `\if...`.

```
\let\repeat=\fi
```

(*End of definition for* `\loop`, `\iterate`, *and* `\repeat`.)

LaTeX defines `\smallskip`, etc. in `ltspace.dtx`.

`\nointerlineskip`
`\offinterlineskip`

```
\def\nointerlineskip{\prevdepth-\@m\p@}
\def\offinterlineskip{\baselineskip-\@m\p@
  \lineskip\z@ \lineskiplimit\maxdimen}
```

(*End of definition for* `\nointerlineskip` *and* `\offinterlineskip`.)

`\vglue`
`\hglue`

```
\def\vglue{\afterassignment\vgl@\skip@=}
\def\vgl@{\par \dimen@\prevdepth \hrule \@height\z@
  \nobreak\vskip\skip@ \prevdepth\dimen@}
\def\hglue{\afterassignment\hgl@\skip@=}
\def\hgl@{\leavevmode \count@\spacefactor \vrule \@width\z@
  \nobreak\hskip\skip@ \spacefactor\count@}
```

(*End of definition for* `\vglue` *and* `\hglue`.)

LaTeX defines `~` in `ltdefns.dtx`.

`\slash` This generates a `/` acting a bit like `-` but still allows hyphenation in the word part preceding it (but not after).

```
\def\slash{/\penalty\exhyphenpenalty}
```

(*End of definition for* `\slash`.)

`\break`
`\nobreak`
`\allowbreak`

```
\def\break{\penalty-\@M}
\def\nobreak{\penalty \@M}
\def\allowbreak{\penalty \z@}
```

(*End of definition for* `\break`, `\nobreak`, *and* `\allowbreak`.)

`\filbreak`
`\goodbreak`

```
\def\filbreak{\par\vfil\penalty-200\vfilneg}
\def\goodbreak{\par\penalty-500 }
```

(*End of definition for* `\filbreak` *and* `\goodbreak`.)

`\eject` Define `\eject` as in plain TeX but define `\superejecct` only in the compatibility file.

```
\def\eject{\par\break}
```

(*End of definition for* `\eject`.)

\removelastskip

```
\def\removelastskip{\ifdim\lastskip=\z@\else\vskip-\lastskip\fi}
```

(*End of definition for* \removelastskip.)

\smallbreak
\medbreak
\bigbreak

```
\def\smallbreak{\par\ifdim\lastskip<\smallskipamount
  \removelastskip\penalty-50\smallskip\fi}
\def\medbreak{\par\ifdim\lastskip<\medskipamount
  \removelastskip\penalty-100\medskip\fi}
\def\bigbreak{\par\ifdim\lastskip<\bigskipamount
  \removelastskip\penalty-200\bigskip\fi}
```

(*End of definition for* \smallbreak, \medbreak, *and* \bigbreak.)

\m@th

```
\def\m@th{\mathsurround\z@}
```

(*End of definition for* \m@th.)

\underbar

Due to LaTeX's redefinition of \underline plain TeX's \underbar can be done in a simpler fashion (but do we need it at all?).

```
\def\underbar#1{\underline{\sbox\tw@{#1}\dp\tw@\z@\box\tw@}}
```

(*End of definition for* \underbar.)

\strutbox
\strut

LaTeX sets \strutbox in \set@fontsize.

```
\newbox\strutbox
\def\strut{\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}
```

(*End of definition for* \strutbox *and* \strut.)

\hidewidth

For alignment entries that can stick out.

```
\def\hidewidth{\hskip\hideskip}
```

(*End of definition for* \hidewidth.)

\narrower

```
\def\narrower{%
  \advance\leftskip\parindent
  \advance\rightskip\parindent}
```

(*End of definition for* \narrower.)

LaTeX defines \ae and similar commands elsewhere.

```
\chardef\%=`\%
\chardef\&=`\&
\chardef\#=`\#
```

Most text commands are actually encoding specific and therefore defined later, so commented out or removed from this file.

\leavevmode

begins a paragraph, if necessary

```
\def\leavevmode{\unhbox\voidb@x}
```

(*End of definition for* \leavevmode.)

\mathhexbox

```
\def\mathhexbox#1#2#3{\mbox{$\m@th \mathchar"#1#2#3$}}
```

(*End of definition for* \mathhexbox.)

\ialign

```
\def\ialign{\everycr{}\tabskip\z@skip\halign} % initialized \halign
```

(*End of definition for* \ialign.)

\oalign
\o@lign
\ooalign

```
\def\oalign#1{\leavevmode\vtop{\baselineskip\z@skip \lineskip.25ex%
  \ialign{##\crcr#1\crcr}}}
\def\o@lign{\lineskiplimit\z@ \oalign}
\def\ooalign{\lineskiplimit-\maxdimen \oalign}
```

(*End of definition for* \oalign, \o@lign, *and* \ooalign.)

\sh@ft

The definition of this macro in plain.tex was improved in about 1997; but as a result its usage was changed and its new definition is not appropriate for LaTeX.

Since the version given here has been in use by LaTeX for many years it does not seem prudent to remove it now. As far as we can tell it has only been used to define \b and \d but this cannot be certain.

```
\def\sh@ft#1{\dimen@.00#1ex\multiply\dimen@\fontdimen1\font
  \kern-.0156\dimen@} % compensate for slant in lowered accents
```

(*End of definition for* \sh@ft.)

\ltx@sh@ft

This is the LaTeX version of the second incarnation of the plain macro \sh@ft, which takes a dimension as its argument. It shifts a pseudo-accent horizontally by an amount proportional to the product of its argument and the slant-per-point (fontdimen 1).

```
\def\ltx@sh@ft #1{%
  \dimen@ #1%
  \kern \strip@pt
    \fontdimen1\font \dimen@
  } % kern by #1 times the current slant
```

(*End of definition for* \ltx@sh@ft.)

LaTeX change: the text commands such as \d, \b, \c, \copyright, \TeX are now defined elsewhere.

LaTeX change: Make \t work in a moving argument. Now defined elsewhere.

\hrulefill
\dotfill

LaTeX change: \kern\z@ added to end of \hrulefill and \dotfill to make them work in 'tabular' and 'array' environments. (Change made 24 July 1987). LaTeX change: \leavevmode added at beginning of \dotfill and \hrulefill so that they work as expected in vertical mode.

```
\def\hrulefill{\leavevmode\leaders\hrule\hfill\kern\z@}
```

The box in \dotfill originally contained (in plain.tex):
\mkern 1.5mu .\mkern 1.5mu;
the width of .44em differs from this by .04pt which is probably an acceptable difference within leaders.

```
\def\dotfill{%
  \leavevmode
  \cleaders \hb@xt@ .44em{\hss.\hss}\hfill
  \kern\z@}
```

(*End of definition for* `\hrulefill` *and* `\dotfill`.)

INITEX sets `\sfcode x=1000` for all x, except that `\sfcode‘X=999` for uppercase letters. The following changes are needed:

```
\sfcode‘\)=0 \sfcode‘\’=0 \sfcode‘\]=0
```

The `\nonfrenchspacing` macro will make further changes to `\sfcode` values.

Definitions related to output

`\magnification` doesn't work in LaTeX.

```
def\magnification{\afterassignment\m@g\count@}
def\m@g{\mag\count@
 \hsize6.5truein\vsize8.9truein\dimen\footins8truein}
```

`\showoverfull`

The following commands are used in debugging:

```
\def\showoverfull{\tracingonline\@ne}
```

(*End of definition for* `\showoverfull`.)

`\showoutput`
`\loggingoutput`

```
\gdef\loggingoutput{\tracingoutput\@ne
    \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}
\gdef\showoutput{\loggingoutput\showoverfull}
⟨/2ekernel⟩
```

(*End of definition for* `\showoutput` *and* `\loggingoutput`.)

`\tracingall`
`\loggingall`

```
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\loggingall}
⟨latexrelease⟩                  {\tracingstacklevels and \tracinglostchars=3}%
⟨*2ekernel | latexrelease⟩
\edef\loggingall{%
  \tracingstats\tw@
  \tracingpages\@ne
  \tracinglostchars\thr@@
  \tracingparagraphs\@ne
  \tracinggroups\@ne
  \tracingifs\@ne
  \tracingscantokens\@ne
  \tracingnesting\@ne
  \errorcontextlines\maxdimen
  \ifdefined\tracingstacklevels \tracingstacklevels\maxdimen \fi
  \noexpand \loggingoutput
  \tracingmacros\tw@
  \tracingcommands\thr@@
  \tracingrestores\@ne
  \tracingassigns\@ne
}%
\def\tracingall{\showoverfull\loggingall}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\loggingall}{etex tracing}%
⟨latexrelease⟩\ifx\tracingscantokens\@undefined
⟨latexrelease⟩\gdef\loggingall{%
```

```
⟨latexrelease⟩  \tracingstats\tw@
⟨latexrelease⟩  \tracingpages\@ne
⟨latexrelease⟩  \tracinglostchars\@ne
⟨latexrelease⟩  \tracingparagraphs\@ne
⟨latexrelease⟩  \errorcontextlines\maxdimen
⟨latexrelease⟩  \loggingoutput
⟨latexrelease⟩  \tracingmacros\tw@
⟨latexrelease⟩  \tracingcommands\tw@
⟨latexrelease⟩  \tracingrestores\@ne
⟨latexrelease⟩  }%
⟨latexrelease⟩\else
⟨latexrelease⟩\gdef\loggingall{%
⟨latexrelease⟩  \tracingstats\tw@
⟨latexrelease⟩  \tracingpages\@ne
⟨latexrelease⟩  \tracinglostchars\tw@
⟨latexrelease⟩  \tracingparagraphs\@ne
⟨latexrelease⟩  \tracinggroups\@ne
⟨latexrelease⟩  \tracingifs\@ne
⟨latexrelease⟩  \tracingscantokens\@ne
⟨latexrelease⟩  \tracingnesting\@ne
⟨latexrelease⟩  \errorcontextlines\maxdimen
⟨latexrelease⟩  \loggingoutput
⟨latexrelease⟩  \tracingmacros\tw@
⟨latexrelease⟩  \tracingcommands\thr@@
⟨latexrelease⟩  \tracingrestores\@ne
⟨latexrelease⟩  \tracingassigns\@ne
⟨latexrelease⟩}%
⟨latexrelease⟩\fi
⟨latexrelease⟩\gdef\tracingall{\showoverfull\loggingall}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\loggingall}{etex tracing}%
⟨latexrelease⟩\gdef\loggingall{\tracingcommands\tw@\tracingstats\tw@
⟨latexrelease⟩  \tracingpages\@ne\tracinglostchars\@ne
⟨latexrelease⟩  \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne
⟨latexrelease⟩  \errorcontextlines\maxdimen\loggingoutput}
⟨latexrelease⟩  \gdef\tracingall{\loggingall\showoverfull}
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\tracingall` *and* `\loggingall`.)

`\tracingnone`

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\tracingnone}%
⟨latexrelease⟩                              {turn off etex tracing}%
⟨*2ekernel | latexrelease⟩
\edef\tracingnone{%
  \tracingassigns\z@
  \tracingrestores\z@
  \tracingonline\z@
  \tracingcommands\z@
  \showboxdepth\m@ne
  \showboxbreadth\m@ne
  \tracingoutput\z@
  \errorcontextlines\m@ne
```

```
  \ifdefined\tracingstacklevels \tracingstacklevels\z@ \fi
  \tracingnesting\z@
  \tracingscantokens\z@
  \tracingifs\z@
  \tracinggroups\z@
  \tracingparagraphs\z@
  \tracingmacros\z@
```

None really means go back to the LaTeX "default" and for `\tracinglostchars` this should therefore be 2 these days.

```
  \tracinglostchars\tw@
  \tracingpages\z@
  \tracingstats\z@
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\tracingnone}%
⟨latexrelease⟩                              {turn off etex tracing}%
⟨latexrelease⟩\ifx\tracingscantokens\@undefined
⟨latexrelease⟩\def\tracingnone{%
⟨latexrelease⟩  \tracingonline\z@
⟨latexrelease⟩  \tracingcommands\z@
⟨latexrelease⟩  \showboxdepth\m@ne
⟨latexrelease⟩  \showboxbreadth\m@ne
⟨latexrelease⟩  \tracingoutput\z@
⟨latexrelease⟩  \errorcontextlines\m@ne
⟨latexrelease⟩  \tracingrestores\z@
⟨latexrelease⟩  \tracingparagraphs\z@
⟨latexrelease⟩  \tracingmacros\z@
⟨latexrelease⟩  \tracinglostchars\@ne
⟨latexrelease⟩  \tracingpages\z@
⟨latexrelease⟩  \tracingstats\z@
⟨latexrelease⟩}%
⟨latexrelease⟩\else
⟨latexrelease⟩\def\tracingnone{%
⟨latexrelease⟩  \tracingassigns\z@
⟨latexrelease⟩  \tracingrestores\z@
⟨latexrelease⟩  \tracingonline\z@
⟨latexrelease⟩  \tracingcommands\z@
⟨latexrelease⟩  \showboxdepth\m@ne
⟨latexrelease⟩  \showboxbreadth\m@ne
⟨latexrelease⟩  \tracingoutput\z@
⟨latexrelease⟩  \errorcontextlines\m@ne
⟨latexrelease⟩  \tracingnesting\z@
⟨latexrelease⟩  \tracingscantokens\z@
⟨latexrelease⟩  \tracingifs\z@
⟨latexrelease⟩  \tracinggroups\z@
⟨latexrelease⟩  \tracingparagraphs\z@
⟨latexrelease⟩  \tracingmacros\z@
⟨latexrelease⟩  \tracinglostchars\@ne
⟨latexrelease⟩  \tracingpages\z@
⟨latexrelease⟩  \tracingstats\z@
⟨latexrelease⟩}%
```

```
⟨latexrelease⟩\fi
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\tracingnone}%
⟨latexrelease⟩                              {turn off etex tracing}%
⟨latexrelease⟩\let\tracingnone\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\tracingnone`.)

`\hideoutput`

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\hideoutput}%
⟨latexrelease⟩                              {hide output from tracing}%
\def\hideoutput{%
  \tracingoutput\z@
  \showboxbreadth\m@ne
  \showboxdepth\m@ne
  \tracingonline\m@ne
}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\hideoutput}%
⟨latexrelease⟩                              {hide output from tracing}%
⟨latexrelease⟩\let\hideoutput\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
```

(*End of definition for* `\hideoutput`.)

LaTeX change: `\showhyphens` Defined later.

Punctuation affects the spacing.

```
⟨*2ekernel⟩
\nonfrenchspacing
⟨/2ekernel⟩
```

File 03
ltvers.dtx

1 Version Identification

First we identify the date and version number of this release of LaTeX, and set `\everyjob` so that it is printed at the start of every LaTeX run.

`\fmtname`
`\fmtversion`
`\latexreleaseversion`
`\patch@level`

A `\patch@level` of `0` or higher denotes an official public release. A negative value indicates a candidate release that is not distributed.

If we put code updates into the kernel that are supposed to go into the next release we set the `\patch@level` to `-1` and the `\fmtversion` / `\latexreleaseversion` to the dated of the next release (guessed, the real value is not so important and will get corrected when we make the release official).

If the `\patch@level` is already at `-1` we do nothing here and use the `\fmtversion` date for any new`\IncludeInRelease` line when we add further code.

Finally, if we do make a public release we either just set the `\patch@level` to zero (if our initial guess was good) or we also change the date and then have to additionally change to that date on all the `\IncludeInRelease` statements that used the "guessed" date.

```
⟨*2ekernel⟩
\def\fmtname{LaTeX2e}
\edef\fmtversion
⟨/2ekernel⟩
⟨latexrelease⟩\edef\latexreleaseversion
⟨*2ekernel | latexrelease⟩
   {2024-11-01}
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
\def\patch@level{0}
```

For more fine grain control there is the possibility to name the current development branch. This is only used when the `\patch@level` is negative (i.e., a pre-release format) and is intended to help us internally when we locally install a format out of some development branch.

`\development@branch@name`

```
\edef\development@branch@name{}
```

(*End of definition for* `\fmtname` *and others.*)

Check that the format being made is not too old. The error message complains about 'more than 5 years' but in fact the error is not triggered until 65 months.

This code is currently not activated as we don't know if we already got to the last official 2e version (due to staff shortage or due to a successor (think positive:-)).

```
\iffalse
\def\reserved@a#1/#2/#3\@nil{%
  \count@\year
  \advance\count@-#1\relax
  \multiply\count@ by 12\relax
  \advance\count@\month
  \advance\count@-#2\relax}
\expandafter\reserved@a\fmtversion\@nil
```

\count@ is now the age of this file in months. Take a generous definition of 'year' so this message is not generated too often.

```
\ifnum\count@>65
  \typeout{^^J%
!!!
```

```
    \fi
  \fi
⟨/2ekernel⟩
```

\IncludeInRelease
\EndIncludeInRelease
\@IncludeInRelease
\@IncludeInRele@se
\@gobble@IncludeInRelease
\@check@IncludeInRelease

```
⟨2ekernel⟩\let\@currname\@empty
```

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\newif\if@includeinrelease
⟨latexrelease⟩\@includeinreleasefalse
```

```
\def\IncludeInRelease#1{%
 \if@includeinrelease
   \PackageError{latexrelease}{mis-matched IncludeInRelease}%
                 {There is an \string\EndIncludeRelease\space missing}%
  \@includeinreleasefalse
  \fi
  \ifnum0%
      \ifx\new@moduledate\@empty\else 1\fi
      \ifnum \expandafter\@parse@version#1//00\@nil=0 1\fi
      =11
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi
    {\finish@module@release{#1}}%
    {\kernel@ifnextchar[%
      {\@IncludeInRelease{#1}}
      {\@IncludeInRelease{#1}[#1]}}}
\def\finish@module@release#1#2#3{%
  \toks@{[#1] #3}%
  \begingroup
    \edef\x{\detokenize\expandafter{\new@modulename}}%
    \edef\y{\detokenize{#2}}%
  \expandafter\endgroup
  \ifx\x\y \else
    \@latex@error{\noexpand\IncludeInRelease dated #1 in a module is not
      allowed.\MessageBreak Use a date at least equal to \new@moduledate
      \space for complete rollback}\@ehd
  \fi
  \ifnum\expandafter\@parse@version\new@moduledate//00\@nil
       >\expandafter\@parse@version\fmtversion//00\@nil
    \GenericInfo{}{Applying: \the\toks@}%
  \else
    \GenericInfo{}{Skipping: \the\toks@}%
    \expandafter\gobble@finish@module@release
  \fi}
\long\def\gobble@finish@module@release#1\EndModuleRelease{%
  \EndModuleRelease}
```

If a specific date has not been specified in `latexrelease` use '`#1`'.

```
\def\@IncludeInRelease#1[#2]{\@IncludeInRele@se{#2}}
```

```
\def\@IncludeInRele@se#1#2#3{%
  \toks@{[#1] #3}%
  \expandafter\ifx\csname\string#2+\@currname+IIR\endcsname\relax
```

If we roll back and the first patch already match then applying that is actually reapplying what is already in the format, i.e., it is useless and possibly allocating new registers. However, it makes the logic simpler so this is the way it is for now. In theory we could always jump over the first patch because that is only really needed for rolling forward. So maybe one day ...

```
    \ifnum\expandafter\@parse@version#1//00\@nil
         >\expandafter\@parse@version\fmtversion//00\@nil
      \GenericInfo{}{Skipping: \the\toks@}%
      \expandafter\expandafter\expandafter\@gobble@IncludeInRelease
    \else
      \GenericInfo{}{Applying: \the\toks@}%
      \@includeinreleasetrue
      \expandafter\let\csname\string#2+\@currname+IIR\endcsname\@empty
    \fi
  \else
    \GenericInfo{}{Already applied: \the\toks@}%
    \expandafter\@gobble@IncludeInRelease
  \fi
}
```

```
\def\EndIncludeInRelease{%
\if@includeinrelease
  \@includeinreleasefalse
\else
  \PackageError{latexrelease}{mis-matched EndIncludeInRelease}{}%
\fi
\if@skipping@module
  \expandafter\new@module@skip
\fi}
```

```
\long\def\@gobble@IncludeInRelease#1\EndIncludeInRelease{%
  \@includeinreleasefalse
  \@check@IncludeInRelease#1\IncludeInRelease\@check@IncludeInRelease
  \@end@check@IncludeInRelease}
```

```
\long\def\@check@IncludeInRelease#1\IncludeInRelease
                                    #2#3\@end@check@IncludeInRelease{%
  \ifx\@check@IncludeInRelease#2\else
    \PackageError{latexrelease}{skipped IncludeInRelease for tag \string#2}{}%
  \fi
  \if@skipping@module
    \expandafter\new@module@skip
  \fi}
```

(*End of definition for* `\IncludeInRelease` *and others.*)

1.1 Declaring an all-new module

`\if@skipping@module`
`\NewModuleRelease`
`\EndModuleRelease`
`\new@module@skip`
`\new@modulename`
`\new@moduledate`

When we have a whole new module, we can't roll back to a date where such module exists, otherwise hundreds of "command already defined" errors will pop up. But we can't skip it altogether either, because the module might have changes we still want applied, so a more detailed cherry-picking of code chunks have to be done.

```
\let\if@skipping@module\iffalse
\def\@skipping@moduletrue{\let\if@skipping@module\iftrue}
\def\@skipping@modulefalse{\let\if@skipping@module\iffalse}
```

```
\let\new@modulename\@empty
\let\new@moduledate\@empty
\def\NewModuleRelease#1#2#3{%
  \ifx\new@modulename\@empty \else
    \@latex@error{Nested \noexpand\NewModuleRelease forbidden.}\@ehd \fi
  \edef\new@moduledate{#1}%
  \edef\new@modulename{#2}%
  \GenericInfo{}{BEGIN module: \new@modulename\space (\new@moduledate)}%
  \GenericInfo{}{ \@spaces\@spaces\@spaces\space#3\@gobble}%
  \ifnum\sourceLaTeXdate<%
        \expandafter\@parse@version\new@moduledate//00\@nil\relax
    \ifnum\expandafter\@parse@version\fmtversion//00\@nil<%
          \expandafter\@parse@version\new@moduledate//00\@nil\relax
      \GenericInfo{}{Skipping module \new@modulename}%
      \expandafter\expandafter
      \expandafter\gobble@finish@module@release
    \else
      \GenericInfo{}{Applying module \new@modulename}
      \@skipping@modulefalse
    \fi
  \else
    \GenericInfo{}{Skipping module \new@modulename}
    \@skipping@moduletrue
    \expandafter\new@module@skip
  \fi}
\long\def\new@module@skip#1\IncludeInRelease{%
  \long\def\reserved@a##1\EndModuleRelease{}%
  \if\relax\detokenize\expandafter{\reserved@a#1{}{}\EndModuleRelease}\relax
  \else
    \@latex@error{Missing mandatory \string\IncludeInRelease{0000/00/00}}\@ehc
    \expandafter\@secondoftwo
  \fi
  \@gobble
    {\@expandtwoargs\IncludeInRelease
         {0000/00/00}{\new@modulename}%
         {ERROR! Emergency recovery}%
     #1}%
  \IncludeInRelease}
\def\EndModuleRelease{%
  \ifx\new@modulename\@empty
    \@latex@error{Extra \string\EndModuleRelease.}\@eha
  \else
    \GenericInfo{}{END module: \new@modulename\space (\new@moduledate)}%
    \let\new@modulename\@empty
    \let\new@moduledate\@empty
    \@skipping@modulefalse
  \fi}
```

(*End of definition for* `\if@skipping@module` *and others.*)

```
⟨/2ekernel | latexrelease⟩
```

File 04
ltluatex.dtx

1 Overview

LuaTEX adds a number of engine-specific functions to TEX. Several of these require set up that is best done in the kernel or need related support functions. This file provides *basic* support for LuaTEX at the LATEX 2_ε kernel level plus as a loadable file which can be used with plain TEX and LATEX.

This file contains code for both TEX (to be stored as part of the format) and Lua (to be loaded at the start of each job). In the Lua code, the kernel uses the namespace `luatexbase`.

The following `\count` registers are used here for register allocation:

`\e@alloc@attribute@count` Attributes (default 258)

`\e@alloc@ccodetable@count` Category code tables (default 259)

`\e@alloc@luafunction@count` Lua functions (default 260)

`\e@alloc@whatsit@count` User whatsits (default 261)

`\e@alloc@bytecode@count` Lua bytecodes (default 262)

`\e@alloc@luachunk@count` Lua chunks (default 263)

(`\count 256` is used for `\newmarks` allocation and `\count 257` is used for `\newXeTeXintercharclass` with XeTEX, with code defined in `ltfinal.dtx`). With any LATEX 2_ε kernel from 2015 onward these registers are part of the block in the extended area reserved by the kernel (prior to 2015 the LATEX 2_ε kernel did not provide any functionality for the extended allocation area).

2 Core TEX functionality

The commands defined here are defined for possible inclusion in a future LATEX format, however also extracted to the file `ltluatex.tex` which may be used with older LATEX formats, and with plain TEX.

`\newattribute` `\newattribute{`⟨*attribute*⟩`}`
Defines a named `\attribute`, indexed from 1 (*i.e.* `\attribute0` is never defined). Attributes initially have the marker value `-"7FFFFFFF` ('unset') set by the engine.

`\newcatcodetable` `\newcatcodetable{`⟨*catcodetable*⟩`}`
Defines a named `\catcodetable`, indexed from 1 (`\catcodetable0` is never assigned). A new catcode table will be populated with exactly those values assigned by IniTEX (as described in the LuaTEX manual).

`\newluafunction` `\newluafunction{`⟨*function*⟩`}`
Defines a named `\luafunction`, indexed from 1. (Lua indexes tables from 1 so `\luafunction0` is not available).

`\newluacmd` `\newluadef{`⟨*function*⟩`}`
Like `\newluafunction`, but defines the command using `\luadef` instead of just assigning an integer.

`\newprotectedluacmd` `\newluadef{⟨function⟩}`
Like `\newluacmd`, but the defined command is not expandable.

`\newwhatsit` `\newwhatsit{⟨whatsit⟩}`
Defines a custom `\whatsit`, indexed from 1.

`\newluabytecode` `\newluabytecode{⟨bytecode⟩}`
Allocates a number for Lua bytecode register, indexed from 1.

`\newluachunkname` `newluachunkname{⟨chunkname⟩}`
Allocates a number for Lua chunk register, indexed from 1. Also enters the name of the register (without backslash) into the `lua.name` table to be used in stack traces.

`\catcodetable@initex` `\catcodetable@string` `\catcodetable@latex` `\catcodetable@atletter`
Predefined category code tables with the obvious assignments. Note that the `latex` and `atletter` tables set the full Unicode range to the codes predefined by the kernel.

`\setattribute` `\setattribute{⟨attribute⟩}{⟨value⟩}`
`\unsetattribute` `\unsetattribute{⟨attribute⟩}`
Set and unset attributes in a manner analogous to `\setlength`. Note that attributes take a marker value when unset so this operation is distinct from setting the value to zero.

3 Plain TEX interface

The ltluatex interface may be used with plain TEX using `\input{ltluatex}`. This inputs `ltluatex.tex` which inputs `etex.src` (or `etex.sty` if used with LATEX) if it is not already input, and then defines some internal commands to allow the ltluatex interface to be defined.

The luatexbase package interface may also be used in plain TEX, as before, by inputting the package `\input luatexbase.sty`. The new version of luatexbase is based on this ltluatex code but implements a compatibility layer providing the interface of the original package.

4 Lua functionality

4.1 Allocators in Lua

`new_attribute` `luatexbase.new_attribute(⟨attribute⟩)`
Returns an allocation number for the ⟨*attribute*⟩, indexed from 1. The attribute will be initialised with the marker value `-"7FFFFFFF` ('unset'). The attribute allocation sequence is shared with the TEX code but this function does *not* define a token using `\attributedef`. The attribute name is recorded in the `attributes` table. A metatable is provided so that the table syntax can be used consistently for attributes declared in TEX or Lua.

`new_whatsit` `luatexbase.new_whatsit(⟨whatsit⟩)`
Returns an allocation number for the custom ⟨*whatsit*⟩, indexed from 1.

`new_bytecode` `luatexbase.new_bytecode(⟨bytecode⟩)`
Returns an allocation number for a bytecode register, indexed from 1. The optional ⟨*name*⟩ argument is just used for logging.

`new_chunkname` `luatexbase.new_chunkname(⟨chunkname⟩)`
Returns an allocation number for a Lua chunk name for use with `\directlua` and `\latelua`, indexed from 1. The number is returned and also ⟨*name*⟩ argument is added to the `lua.name` array at that index.

`new_luafunction` `luatexbase.new_luafunction(⟨functionname⟩)`

Returns an allocation number for a lua function for use with `\luafunction`, `\lateluafunction`, and `\luadef`, indexed from 1. The optional ⟨*functionname*⟩ argument is just used for logging.

These functions all require access to a named TEX count register to manage their allocations. The standard names are those defined above for access from TEX, *e.g.* \e@alloc@attribute@count, but these can be adjusted by defining the variable ⟨*type*⟩`_count_name` before loading `ltluatex.lua`, for example

```
local attribute_count_name = "attributetracker"
require("ltluatex")
```

would use a TEX `\count` (`\countdef`'d token) called `attributetracker` in place of \e@alloc@attribute@count.

4.2 Lua access to TEX register numbers

registernumber `luatexbase.registernumer(`⟨*name*⟩`)`

Sometimes (notably in the case of Lua attributes) it is necessary to access a register *by number* that has been allocated by TEX. This package provides a function to look up the relevant number using LuaTEX's internal tables. After for example `\newattribute\myattrib`, `\myattrib` would be defined by (say) `\myattrib=\attribute15`. `luatexbase.registernumer("myattrib")` would then return the register number, 15 in this case. If the string passed as argument does not correspond to a token defined by `\attributedef`, `\countdef` or similar commands, the Lua value `false` is returned.

As an example, consider the input:

```
\newcommand\test[1]{%
\typeout{#1: \expandafter\meaning\csname#1\endcsname^^J
\space\space\space\space
\directlua{tex.write(luatexbase.registernumber("#1") or "bad input")}%
}}

\test{undefinedrubbish}

\test{space}

\test{hbox}

\test{@MM}

\test{@tempdima}
\test{@tempdimb}

\test{strutbox}

\test{sixt@@n}

\attrbutedef\myattr=12
\myattr=200
\test{myattr}
```

If the demonstration code is processed with LuaLaTeX then the following would be produced in the log and terminal output.

```
undefinedrubbish: \relax
     bad input
space: macro:->
     bad input
hbox: \hbox
     bad input
@MM: \mathchar"4E20
     20000
@tempdima: \dimen14
     14
@tempdimb: \dimen15
     15
strutbox: \char"B
     11
sixt@@n: \char"10
     16
myattr: \attribute12
     12
```

Notice how undefined commands, or commands unrelated to registers do not produce an error, just return `false` and so print `bad input` here. Note also that commands defined by `\newbox` work and return the number of the box register even though the actual command holding this number is a `\chardef` defined token (there is no `\boxdef`).

4.3 Module utilities

`provides_module` `luatexbase.provides_module(`⟨*info*⟩`)`
This function is used by modules to identify themselves; the `info` should be a table containing information about the module. The required field `name` must contain the name of the module. It is recommended to provide a field `date` in the usual LaTeX format `yyyy/mm/dd`. Optional fields `version` (a string) and `description` may be used if present. This information will be recorded in the log. Other fields are ignored. If the `version` begins with a digit, a `v` will be added at the start in the log.

`module_info` `luatexbase.module_info(`⟨*module*⟩, ⟨*text*⟩`)`
`module_warning` `luatexbase.module_warning(`⟨*module*⟩, ⟨*text*⟩`)`
`module_error` `luatexbase.module_error(`⟨*module*⟩, ⟨*text*⟩`)`
These functions are similar to LaTeX's `\PackageError`, `\PackageWarning` and `\PackageInfo` in the way they format the output. No automatic line breaking is done, you may still use `\n` as usual for that, and the name of the package will be prepended to each output line.

Note that `luatexbase.module_error` raises an actual Lua error with `error()`, which currently means a call stack will be dumped. While this may not look pretty, at least it provides useful information for tracking the error down.

4.4 Callback management

`add_to_callback` `luatexbase.add_to_callback(`⟨*callback*⟩, ⟨*function*⟩, ⟨*description*⟩`)` Registers the ⟨*function*⟩ into the ⟨*callback*⟩ with a textual ⟨*description*⟩ of the function. Functions are inserted into the callback in the order loaded.

`remove_from_callback` `luatexbase.remove_from_callback(⟨callback⟩, ⟨description⟩)` Removes the callback function with ⟨*description*⟩ from the ⟨*callback*⟩. The removed function and its description are returned as the results of this function.

`in_callback` `luatexbase.in_callback(⟨callback⟩, ⟨description⟩)` Checks if the ⟨*description*⟩ matches one of the functions added to the list for the ⟨*callback*⟩, returning a boolean value.

`disable_callback` `luatexbase.disable_callback(⟨callback⟩)` Sets the ⟨*callback*⟩ to `false` as described in the LuaTEX manual for the underlying `callback.register` built-in. Callbacks will only be set to false (and thus be skipped entirely) if there are no functions registered using the callback.

`callback_descriptions` A list of the descriptions of functions registered to the specified callback is returned. `{}` is returned if there are no functions registered.

`create_callback` `luatexbase.create_callback(⟨name⟩,⟨type⟩,⟨default⟩)` Defines a user defined callback. The last argument is a default function or `false`.

`call_callback` `luatexbase.call_callback(⟨name⟩,...)` Calls a user defined callback with the supplied arguments.

`declare_callback_rule` `luatexbase.declare_callback_rule(⟨name⟩, ⟨first⟩, ⟨relation⟩, ⟨second⟩)` Adds an ordering constraint between two callback functions for callback ⟨*name*⟩.

The kind of constraint added depends on ⟨*relation*⟩:

before The callback function with description ⟨*first*⟩ will be executed before the function with description ⟨*second*⟩.

after The callback function with description ⟨*first*⟩ will be executed after the function with description ⟨*second*⟩.

incompatible-warning When both a callback function with description ⟨*first*⟩ and with description ⟨*second*⟩ is registered, then a warning is printed when the callback is executed.

incompatible-error When both a callback function with description ⟨*first*⟩ and with description ⟨*second*⟩ is registered, then an error is printed when the callback is executed.

unrelated Any previously declared callback rule between ⟨*first*⟩ and ⟨*second*⟩ gets disabled.

Every call to `declare_callback_rule` with a specific callback ⟨*name*⟩ and descriptions ⟨*first*⟩ and ⟨*second*⟩ overwrites all previous calls with same callback and descriptions.

The callback functions do not have to be registered yet when the functions is called. Only the constraints for which both callback descriptions refer to callbacks registered at the time the callback is called will have an effect.

5 Implementation

```
⟨*2ekernel | tex | latexrelease⟩
⟨2ekernel | latexrelease⟩\ifx\directlua\@undefined\else
```

5.1 Minimum LuaTEX version

LuaTEX has changed a lot over time. In the kernel support for ancient versions is not provided: trying to build a format with a very old binary therefore gives some information

in the log and loading stops. The cut-off selected here relates to the tree-searching behaviour of `require()`: from version 0.60, LuaTEX will correctly find Lua files in the `texmf` tree without 'help'.

```
⟨latexrelease⟩\IncludeInRelease{2015/10/01}
⟨latexrelease⟩                  {\newluafunction}{LuaTeX}%
\ifnum\luatexversion<60 %
  \wlog{**************************************************}
  \wlog{* LuaTeX version too old for ltluatex support *}
  \wlog{**************************************************}
  \expandafter\endinput
\fi
```

Two simple LaTeX macros from `ltdefns.dtx` have to be defined here because ltdefns.dtx is not loaded yet when ltluatex.dtx is executed.

```
\long\def\@gobble#1{}
\long\def\@firstofone#1{#1}
```

5.2 Older LaTeX/Plain TeX setup

```
⟨*tex⟩
```

Older LaTeX formats don't have the primitives with 'native' names: sort that out. If they already exist this will still be safe.

```
\directlua{tex.enableprimitives("",tex.extraprimitives("luatex"))}
\ifx\e@alloc\@undefined
```

In pre-2014 LaTeX, or plain TeX, load `etex.{sty,src}`.

```
  \ifx\documentclass\@undefined
    \ifx\loccount\@undefined
      \input{etex.src}%
    \fi
    \catcode`\@=11 %
    \outer\expandafter\def\csname newfam\endcsname
                          {\alloc@8\fam\chardef\et@xmaxfam}
  \else
    \RequirePackage{etex}
    \expandafter\def\csname newfam\endcsname
                     {\alloc@8\fam\chardef\et@xmaxfam}
    \expandafter\let\expandafter\new@mathgroup\csname newfam\endcsname
  \fi
```

5.2.1 Fixes to `etex.src/etex.sty`

These could and probably should be made directly in an update to `etex.src` which already has some LuaTEX-specific code, but does not define the correct range for LuaTEX.

2015-07-13 higher range in luatex.

```
\edef \et@xmaxregs {\ifx\directlua\@undefined 32768\else 65536\fi}
```

luatex/xetex also allow more math fam.

```
\edef \et@xmaxfam {\ifx\Umathcode\@undefined\sixt@@n\else\@cclvi\fi}
\count 270=\et@xmaxregs % locally allocates \count registers
\count 271=\et@xmaxregs % ditto for \dimen registers
\count 272=\et@xmaxregs % ditto for \skip registers
\count 273=\et@xmaxregs % ditto for \muskip registers
\count 274=\et@xmaxregs % ditto for \box registers
```

```
\count 275=\et@xmaxregs % ditto for \toks registers
\count 276=\et@xmaxregs % ditto for \marks classes
```

and 256 or 16 fam. (Done above due to plain/LaTeX differences in ltluatex.)

```
% \outer\def\newfam{\alloc@8\fam\chardef\et@xmaxfam}
```

End of proposed changes to `etex.src`

5.2.2 luatex specific settings

Switch to global cf `luatex.sty` to leave room for inserts not really needed for luatex but possibly most compatible with existing use.

```
\expandafter\let\csname newcount\expandafter\expandafter\endcsname
                \csname globcount\endcsname
\expandafter\let\csname newdimen\expandafter\expandafter\endcsname
                \csname globdimen\endcsname
\expandafter\let\csname newskip\expandafter\expandafter\endcsname
                \csname globskip\endcsname
\expandafter\let\csname newbox\expandafter\expandafter\endcsname
                \csname globbox\endcsname
```

Define`\e@alloc` as in LaTeX (the existing macros in `etex.src` are hard to extend to further register types as they assume specific 26x and 27x count range). For compatibility the existing register allocation is not changed.

```
\chardef\e@alloc@top=65535
\let\e@alloc@chardef\chardef
```

```
\def\e@alloc#1#2#3#4#5#6{%
  \global\advance#3\@ne
  \e@ch@ck{#3}{#4}{#5}#1%
  \allocationnumber#3\relax
  \global#2#6\allocationnumber
  \wlog{\string#6=\string#1\the\allocationnumber}}%
```

```
\gdef\e@ch@ck#1#2#3#4{%
  \ifnum#1<#2\else
    \ifnum#1=#2\relax
      #1\@cclvi
      \ifx\count#4\advance#1 10 \fi
    \fi
    \ifnum#1<#3\relax
    \else
      \errmessage{No room for a new \string#4}%
    \fi
  \fi}%
```

Fix up allocations not to clash with `etex.src`.

```
\expandafter\csname newcount\endcsname\e@alloc@attribute@count
\expandafter\csname newcount\endcsname\e@alloc@ccodetable@count
\expandafter\csname newcount\endcsname\e@alloc@luafunction@count
\expandafter\csname newcount\endcsname\e@alloc@whatsit@count
\expandafter\csname newcount\endcsname\e@alloc@bytecode@count
\expandafter\csname newcount\endcsname\e@alloc@luachunk@count
```

End of conditional setup for plain TeX / old LaTeX.

```
\fi
```

⟨/tex⟩

5.3 Attributes

\newattribute

As is generally the case for the LuaTEX registers we start here from 1. Notably, some code assumes that \attribute0 is never used so this is important in this case.

```
\ifx\e@alloc@attribute@count\@undefined
  \countdef\e@alloc@attribute@count=258
  \e@alloc@attribute@count=\z@
\fi
\def\newattribute#1{%
  \e@alloc\attribute\attributedef
    \e@alloc@attribute@count\m@ne\e@alloc@top#1%
}
```

(*End of definition for* \newattribute.)

\setattribute
\unsetattribute

Handy utilities.

```
\def\setattribute#1#2{#1=\numexpr#2\relax}
\def\unsetattribute#1{#1=-"7FFFFFFF\relax}
```

(*End of definition for* \setattribute *and* \unsetattribute.)

5.4 Category code tables

\newcatcodetable

Category code tables are allocated with a limit half of that used by LuaTEX for everything else. At the end of allocation there needs to be an initialization step. Table 0 is already taken (it's the global one for current use) so the allocation starts at 1.

```
\ifx\e@alloc@ccodetable@count\@undefined
  \countdef\e@alloc@ccodetable@count=259
  \e@alloc@ccodetable@count=\z@
\fi
\def\newcatcodetable#1{%
  \e@alloc\catcodetable\chardef
    \e@alloc@ccodetable@count\m@ne{"8000}#1%
  \initcatcodetable\allocationnumber
}
```

(*End of definition for* \newcatcodetable.)

\catcodetable@initex
\catcodetable@string
\catcodetable@latex
\catcodetable@atletter

Save a small set of standard tables. The Unicode data is read here in using a parser simplified from that in load-unicode-data: only the nature of letters needs to be detected.

```
\newcatcodetable\catcodetable@initex
\newcatcodetable\catcodetable@string
\begingroup
  \def\setrangecatcode#1#2#3{%
    \ifnum#1>#2 %
      \expandafter\@gobble
    \else
      \expandafter\@firstofone
    \fi
      {%
        \catcode#1=#3 %
        \expandafter\setrangecatcode\expandafter
          {\number\numexpr#1 + 1\relax}{#2}{#3}
      }%
```

```
  }
  \@firstofone{%
    \catcodetable\catcodetable@initex
      \catcode0=12 %
      \catcode13=12 %
      \catcode37=12 %
      \setrangecatcode{65}{90}{12}%
      \setrangecatcode{97}{122}{12}%
      \catcode92=12 %
      \catcode127=12 %
      \savecatcodetable\catcodetable@string
    \endgroup
  }%
\newcatcodetable\catcodetable@latex
\newcatcodetable\catcodetable@atletter
\begingroup
  \def\parseunicodedataI#1;#2;#3;#4\relax{%
    \parseunicodedataII#1;#3;#2 First>\relax
  }%
  \def\parseunicodedataII#1;#2;#3 First>#4\relax{%
    \ifx\relax#4\relax
      \expandafter\parseunicodedataIII
    \else
      \expandafter\parseunicodedataIV
    \fi
      {#1}#2\relax%
  }%
  \def\parseunicodedataIII#1#2#3\relax{%
    \ifnum 0%
      \if L#21\fi
      \if M#21\fi
      >0 %
      \catcode"#1=11 %
    \fi
  }%
  \def\parseunicodedataIV#1#2#3\relax{%
    \read\unicoderead to \unicodedataline
    \if L#2%
      \count0="#1 %
      \expandafter\parseunicodedataV\unicodedataline\relax
    \fi
  }%
  \def\parseunicodedataV#1;#2\relax{%
    \loop
      \unless\ifnum\count0>"#1 %
        \catcode\count0=11 %
        \advance\count0 by 1 %
    \repeat
  }%
  \def\storedpar{\par}%
  \chardef\unicoderead=\numexpr\count16 + 1\relax
  \openin\unicoderead=UnicodeData.txt %
  \loop\unless\ifeof\unicoderead %
    \read\unicoderead to \unicodedataline
```

```
    \unless\ifx\unicodedataline\storedpar
      \expandafter\parseunicodedataI\unicodedataline\relax
    \fi
  \repeat
  \closein\unicoderead
  \@firstofone{%
    \catcode64=12 %
    \savecatcodetable\catcodetable@latex
    \catcode64=11 %
    \savecatcodetable\catcodetable@atletter
   }
\endgroup
```

(*End of definition for* `\catcodetable@initex` *and others.*)

5.5 Named Lua functions

`\newluafunction` Much the same story for allocating LuaTEX functions except here they are just numbers so they are allocated in the same way as boxes. Lua indexes from 1 so once again slot 0 is skipped.

```
\ifx\e@alloc@luafunction@count\@undefined
  \countdef\e@alloc@luafunction@count=260
  \e@alloc@luafunction@count=\z@
\fi
\def\newluafunction{%
  \e@alloc\luafunction\e@alloc@chardef
    \e@alloc@luafunction@count\m@ne\e@alloc@top
}
```

(*End of definition for* `\newluafunction`.)

`\newluacmd` `\newprotectedluacmd` Additionally two variants are provided to make the passed control sequence call the function directly.

```
\def\newluacmd{%
  \e@alloc\luafunction\luadef
    \e@alloc@luafunction@count\m@ne\e@alloc@top
}
\def\newprotectedluacmd{%
  \e@alloc\luafunction{\protected\luadef}
    \e@alloc@luafunction@count\m@ne\e@alloc@top
}
```

(*End of definition for* `\newluacmd` *and* `\newprotectedluacmd`.)

5.6 Custom whatsits

`\newwhatsit` These are only settable from Lua but for consistency are definable here.

```
\ifx\e@alloc@whatsit@count\@undefined
  \countdef\e@alloc@whatsit@count=261
  \e@alloc@whatsit@count=\z@
\fi
\def\newwhatsit#1{%
  \e@alloc\whatsit\e@alloc@chardef
    \e@alloc@whatsit@count\m@ne\e@alloc@top#1%
}
```

(*End of definition for* `\newwhatsit`.)

5.7 Lua bytecode registers

`\newluabytecode` These are only settable from Lua but for consistency are definable here.

```
\ifx\e@alloc@bytecode@count\@undefined
  \countdef\e@alloc@bytecode@count=262
  \e@alloc@bytecode@count=\z@
\fi
\def\newluabytecode#1{%
  \e@alloc\luabytecode\e@alloc@chardef
    \e@alloc@bytecode@count\m@ne\e@alloc@top#1%
}
```

(*End of definition for* `\newluabytecode`.)

5.8 Lua chunk registers

`\newluachunkname` As for bytecode registers, but in addition we need to add a string to the `lua.name` table to use in stack tracing. We use the name of the command passed to the allocator, with no backslash.

```
\ifx\e@alloc@luachunk@count\@undefined
  \countdef\e@alloc@luachunk@count=263
  \e@alloc@luachunk@count=\z@
\fi
\def\newluachunkname#1{%
  \e@alloc\luachunk\e@alloc@chardef
    \e@alloc@luachunk@count\m@ne\e@alloc@top#1%
    {\escapechar\m@ne
    \directlua{lua.name[\the\allocationnumber]="\string#1"}}%
}
```

(*End of definition for* `\newluachunkname`.)

5.9 Lua loader

Lua code loaded in the format often has to be loaded again at the beginning of every job, so we define a helper which allows us to avoid duplicated code:

```
\def\now@and@everyjob#1{%
  \everyjob\expandafter{\the\everyjob
    #1%
  }%
  #1%
}
```

Load the Lua code at the start of every job. For the conversion of TEX into numbers at the Lua side we need some known registers: for convenience we use a set of systematic names, which means using a group around the Lua loader.

```
⟨2ekernel⟩\now@and@everyjob{%
  \begingroup
    \attributedef\attributezero=0 %
    \chardef     \charzero     =0 %
```

Note name change required on older luatex, for hash table access.

```
    \countdef    \CountZero    =0 %
    \dimendef    \dimenzero    =0 %
    \mathchardef \mathcharzero =0 %
    \muskipdef   \muskipzero   =0 %
    \skipdef     \skipzero     =0 %
    \toksdef     \tokszero     =0 %
    \directlua{require("ltluatex")}
  \endgroup
⟨2ekernel⟩}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}
⟨latexrelease⟩                  {\newluafunction}{LuaTeX}%
⟨latexrelease⟩\let\e@alloc@attribute@count\@undefined
⟨latexrelease⟩\let\newattribute\@undefined
⟨latexrelease⟩\let\setattribute\@undefined
⟨latexrelease⟩\let\unsetattribute\@undefined
⟨latexrelease⟩\let\e@alloc@ccodetable@count\@undefined
⟨latexrelease⟩\let\newcatcodetable\@undefined
⟨latexrelease⟩\let\catcodetable@initex\@undefined
⟨latexrelease⟩\let\catcodetable@string\@undefined
⟨latexrelease⟩\let\catcodetable@latex\@undefined
⟨latexrelease⟩\let\catcodetable@atletter\@undefined
⟨latexrelease⟩\let\e@alloc@luafunction@count\@undefined
⟨latexrelease⟩\let\newluafunction\@undefined
⟨latexrelease⟩\let\e@alloc@luafunction@count\@undefined
⟨latexrelease⟩\let\newwhatsit\@undefined
⟨latexrelease⟩\let\e@alloc@whatsit@count\@undefined
⟨latexrelease⟩\let\newluabytecode\@undefined
⟨latexrelease⟩\let\e@alloc@bytecode@count\@undefined
⟨latexrelease⟩\let\newluachunkname\@undefined
⟨latexrelease⟩\let\e@alloc@luachunk@count\@undefined
⟨latexrelease⟩\directlua{luatexbase.uninstall()}
⟨latexrelease⟩\EndIncludeInRelease
```

In `\everyjob`, if luaotfload is available, load it and switch to TU.

```
⟨latexrelease⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease⟩                  {\fontencoding}{TU in everyjob}%
⟨latexrelease⟩\fontencoding{TU}\let\encodingdefault\f@encoding
⟨latexrelease⟩\ifx\directlua\@undefined\else
⟨2ekernel⟩\everyjob\expandafter{%
⟨2ekernel⟩   \the\everyjob
⟨*2ekernel, latexrelease⟩
  \directlua{%
  if xpcall(function ()%
              require('luaotfload-main')%
             end,texio.write_nl) then %
  local _void = luaotfload.main ()%
  else %
  texio.write_nl('Error in luaotfload: reverting to OT1')%
  tex.print('\string\\def\string\\encodingdefault{OT1}')%
  end %
  }%
```

```
  \let\f@encoding\encodingdefault
  \expandafter\let\csname ver@luaotfload.sty\endcsname\fmtversion
⟨/2ekernel, latexrelease⟩
⟨latexrelease⟩\fi
⟨2ekernel⟩   }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\fontencoding}{TU in everyjob}%
⟨latexrelease⟩\fontencoding{OT1}\let\encodingdefault\f@encoding
⟨latexrelease⟩\EndIncludeInRelease
⟨2ekernel | latexrelease⟩\fi
⟨/2ekernel | tex | latexrelease⟩
```

5.10 Lua module preliminaries

```
⟨*lua⟩
```

Some set up for the Lua module which is needed for all of the Lua functionality added here.

`luatexbase` Set up the table for the returned functions. This is used to expose all of the public functions.

```
luatexbase       = luatexbase or { }
local luatexbase = luatexbase
```

(*End of definition for* `luatexbase`.)

Some Lua best practice: use local versions of functions where possible.

```
local string_gsub      = string.gsub
local tex_count        = tex.count
local tex_setcount     = tex.setcount
local texio_write_nl   = texio.write_nl
local flush_list       = node.flush_list
```

```
local luatexbase_warning
local luatexbase_error
```

5.11 Lua module utilities

5.11.1 Module tracking

`modules` To allow tracking of module usage, a structure is provided to store information and to return it.

```
local modules = modules or { }
```

(*End of definition for* `modules`.)

`provides_module` Local function to write to the log.

```
local function luatexbase_log(text)
  texio_write_nl("log", text)
end
```

Modelled on \ProvidesPackage, we store much the same information but with a little more structure.

```
local function provides_module(info)
  if not (info and info.name) then
    luatexbase_error("Missing module name for provides_module")
  end
  local function spaced(text)
    return text and (" " .. text) or ""
  end
  luatexbase_log(
    "Lua module: " .. info.name
      .. spaced(info.date)
      .. spaced(info.version and string_gsub(info.version or "","^(%d)","v%1"))
      .. spaced(info.description)
  )
  modules[info.name] = info
end
luatexbase.provides_module = provides_module
```

(*End of definition for* provides_module.)

5.11.2 Module messages

There are various warnings and errors that need to be given. For warnings we can get exactly the same formatting as from TEX. For errors we have to make some changes. Here we give the text of the error in the LATEX format then force an error from Lua to halt the run. Splitting the message text is done using \n which takes the place of \MessageBreak.

First an auxiliary for the formatting: this measures up the message leader so we always get the correct indent.

```
local function msg_format(mod, msg_type, text)
  local leader = ""
  local cont
  local first_head
  if mod == "LaTeX" then
    cont = string_gsub(leader, ".", " ")
    first_head = leader .. "LaTeX: "
  else
    first_head = leader .. "Module "  .. msg_type
    cont = "(" .. mod .. ")"
      .. string_gsub(first_head, ".", " ")
    first_head =  leader .. "Module "  .. mod .. " " .. msg_type  .. ":"
  end
  if msg_type == "Error" then
    first_head = "\n" .. first_head
  end
  if string.sub(text,-1) ~= "\n" then
    text = text .. " "
  end
  return first_head .. " "
    .. string_gsub(
         text
  .. "on input line "
```

```
          .. tex.inputlineno, "\n", "\n" .. cont .. " "
       )
    .. "\n"
end
```

module_info
module_warning
module_error

Write messages.

```
local function module_info(mod, text)
  texio_write_nl("log", msg_format(mod, "Info", text))
end
luatexbase.module_info = module_info
local function module_warning(mod, text)
  texio_write_nl("term and log",msg_format(mod, "Warning", text))
end
luatexbase.module_warning = module_warning
local function module_error(mod, text)
  error(msg_format(mod, "Error", text))
end
luatexbase.module_error = module_error
```

(*End of definition for* `module_info`, `module_warning`, *and* `module_error`.)

Dedicated versions for the rest of the code here.

```
function luatexbase_warning(text)
  module_warning("luatexbase", text)
end
function luatexbase_error(text)
  module_error("luatexbase", text)
end
```

5.12 Accessing register numbers from Lua

Collect up the data from the TEX level into a Lua table: from version 0.80, LuaTEX makes that easy.

```
local luaregisterbasetable = { }
local registermap = {
  attributezero = "assign_attr"    ,
  charzero      = "char_given"     ,
  CountZero     = "assign_int"     ,
  dimenzero     = "assign_dimen"   ,
  mathcharzero  = "math_given"     ,
  muskipzero    = "assign_mu_skip" ,
  skipzero      = "assign_skip"    ,
  tokszero      = "assign_toks"    ,
}
local createtoken
if tex.luatexversion > 81 then
  createtoken = token.create
elseif tex.luatexversion > 79 then
  createtoken = newtoken.create
end
local hashtokens    = tex.hashtokens()
local luatexversion = tex.luatexversion
for i,j in pairs (registermap) do
  if luatexversion < 80 then
```

```
    luaregisterbasetable[hashtokens[i][1]] =
      hashtokens[i][2]
  else
    luaregisterbasetable[j] = createtoken(i).mode
  end
end
```

registernumber

Working out the correct return value can be done in two ways. For older LuaTEX releases it has to be extracted from the `hashtokens`. On the other hand, newer LuaTEX's have `newtoken`, and whilst `.mode` isn't currently documented, Hans Hagen pointed to this approach so we should be OK.

```
local registernumber
if luatexversion < 80 then
  function registernumber(name)
    local nt = hashtokens[name]
    if(nt and luaregisterbasetable[nt[1]]) then
      return nt[2] - luaregisterbasetable[nt[1]]
    else
      return false
    end
  end
else
  function registernumber(name)
    local nt = createtoken(name)
    if(luaregisterbasetable[nt.cmdname]) then
      return nt.mode - luaregisterbasetable[nt.cmdname]
    else
      return false
    end
  end
end
luatexbase.registernumber = registernumber
```

(*End of definition for* `registernumber`.)

5.13 Attribute allocation

new_attribute

As attributes are used for Lua manipulations its useful to be able to assign from this end.

```
local attributes=setmetatable(
{},
{
__index = function(t,key)
return registernumber(key) or nil
end}
)
luatexbase.attributes = attributes
```

```
local attribute_count_name =
                     attribute_count_name or "e@alloc@attribute@count"
local function new_attribute(name)
  tex_setcount("global", attribute_count_name,
                         tex_count[attribute_count_name] + 1)
  if tex_count[attribute_count_name] > 65534 then
    luatexbase_error("No room for a new \\attribute")
```

```
  end
  attributes[name]= tex_count[attribute_count_name]
  luatexbase_log("Lua-only attribute " .. name .. " = " ..
                 tex_count[attribute_count_name])
  return tex_count[attribute_count_name]
end
luatexbase.new_attribute = new_attribute
```

(*End of definition for* `new_attribute`.)

5.14 Custom whatsit allocation

`new_whatsit` Much the same as for attribute allocation in Lua.

```
local whatsit_count_name = whatsit_count_name or "e@alloc@whatsit@count"
local function new_whatsit(name)
  tex_setcount("global", whatsit_count_name,
                         tex_count[whatsit_count_name] + 1)
  if tex_count[whatsit_count_name] > 65534 then
    luatexbase_error("No room for a new custom whatsit")
  end
  luatexbase_log("Custom whatsit " .. (name or "") .. " = " ..
                 tex_count[whatsit_count_name])
  return tex_count[whatsit_count_name]
end
luatexbase.new_whatsit = new_whatsit
```

(*End of definition for* `new_whatsit`.)

5.15 Bytecode register allocation

`new_bytecode` Much the same as for attribute allocation in Lua. The optional ⟨*name*⟩ argument is used in the log if given.

```
local bytecode_count_name =
                         bytecode_count_name or "e@alloc@bytecode@count"
local function new_bytecode(name)
  tex_setcount("global", bytecode_count_name,
                         tex_count[bytecode_count_name] + 1)
  if tex_count[bytecode_count_name] > 65534 then
    luatexbase_error("No room for a new bytecode register")
  end
  luatexbase_log("Lua bytecode " .. (name or "") .. " = " ..
                 tex_count[bytecode_count_name])
  return tex_count[bytecode_count_name]
end
luatexbase.new_bytecode = new_bytecode
```

(*End of definition for* `new_bytecode`.)

5.16 Lua chunk name allocation

`new_chunkname` As for bytecode registers but also store the name in the `lua.name` table.

```
local chunkname_count_name =
                        chunkname_count_name or "e@alloc@luachunk@count"
local function new_chunkname(name)
```

```
  tex_setcount("global", chunkname_count_name,
                         tex_count[chunkname_count_name] + 1)
  local chunkname_count = tex_count[chunkname_count_name]
  chunkname_count = chunkname_count + 1
  if chunkname_count > 65534 then
    luatexbase_error("No room for a new chunkname")
  end
  lua.name[chunkname_count]=name
  luatexbase_log("Lua chunkname " .. (name or "") .. " = " ..
                 chunkname_count .. "\n")
  return chunkname_count
end
luatexbase.new_chunkname = new_chunkname
```

(*End of definition for* `new_chunkname`.)

5.17 Lua function allocation

`new_luafunction` Much the same as for attribute allocation in Lua. The optional ⟨*name*⟩ argument is used in the log if given.

```
local luafunction_count_name =
                         luafunction_count_name or "e@alloc@luafunction@count"
local function new_luafunction(name)
  tex_setcount("global", luafunction_count_name,
                         math.max(
                           #(lua.get_functions_table()),
                           tex_count[luafunction_count_name])
                          + 1)
  lua.get_functions_table()[tex_count[luafunction_count_name]] = false
  if tex_count[luafunction_count_name] > 65534 then
    luatexbase_error("No room for a new luafunction register")
  end
  luatexbase_log("Lua function " .. (name or "") .. " = " ..
                 tex_count[luafunction_count_name])
  return tex_count[luafunction_count_name]
end
luatexbase.new_luafunction = new_luafunction
```

(*End of definition for* `new_luafunction`.)

5.18 Lua callback management

The native mechanism for callbacks in LuaTEX allows only one per function. That is extremely restrictive and so a mechanism is needed to add and remove callbacks from the appropriate hooks.

5.18.1 Housekeeping

The main table: keys are callback names, and values are the associated lists of functions. More precisely, the entries in the list are tables holding the actual function as `func` and the identifying description as `description`. Only callbacks with a non-empty list of functions have an entry in this list.

Actually there are two tables: `realcallbacklist` directly contains the entries as described above while `callbacklist` only directly contains the already sorted entries. Other entries can be queried through `callbacklist` too which triggers a resort.

Additionally `callbackrules` describes the ordering constraints: It contains two element tables with the descriptions of the constrained callback implementations. It can additionally contain a `type` entry indicating the kind of rule. A missing value indicates a normal ordering constraint.

```
local realcallbacklist = {}
local callbackrules = {}
local callbacklist = setmetatable({}, {
  __index = function(t, name)
    local list = realcallbacklist[name]
    local rules = callbackrules[name]
    if list and rules then
      local meta = {}
      for i, entry in ipairs(list) do
        local t = {value = entry, count = 0, pos = i}
        meta[entry.description], list[i] = t, t
      end
      local count = #list
      local pos = count
      for i, rule in ipairs(rules) do
        local rule = rules[i]
        local pre, post = meta[rule[1]], meta[rule[2]]
        if pre and post then
          if rule.type then
            if not rule.hidden then
              assert(rule.type == 'incompatible-warning' and luatexbase_warning
                or rule.type == 'incompatible-error' and luatexbase_error)(
                  "Incompatible functions \"" .. rule[1] .. "\" and \"" .. rule[2]
                  .. "\" specified for callback \"" .. name .. "\".")
              rule.hidden = true
            end
          else
            local post_count = post.count
            post.count = post_count+1
            if post_count == 0 then
              local post_pos = post.pos
              if post_pos ~= pos then
                local new_post_pos = list[pos]
                new_post_pos.pos = post_pos
                list[post_pos] = new_post_pos
              end
              list[pos] = nil
              pos = pos - 1
            end
            pre[#pre+1] = post
          end
        end
      end
      for i=1, count do -- The actual sort begins
        local current = list[i]
        if current then
```

```
        meta[current.value.description] = nil
        for j, cur in ipairs(current) do
          local count = cur.count
          if count == 1 then
            pos = pos + 1
            list[pos] = cur
          else
            cur.count = count - 1
          end
        end
        list[i] = current.value
      else
        -- Cycle occurred. TODO: Show cycle for debugging
        -- list[i] = ...
        local remaining = {}
        for name, entry in next, meta do
          local value = entry.value
          list[#list + 1] = entry.value
          remaining[#remaining + 1] = name
        end
        table.sort(remaining)
        local first_name = remaining[1]
        for j, name in ipairs(remaining) do
          local entry = meta[name]
          list[i + j - 1] = entry.value
          for _, post_entry in ipairs(entry) do
            local post_name = post_entry.value.description
            if not remaining[post_name] then
              remaining[post_name] = name
            end
          end
        end
        local cycle = {first_name}
        local index = 1
        local last_name = first_name
        repeat
          cycle[last_name] = index
          last_name = remaining[last_name]
          index = index + 1
          cycle[index] = last_name
        until cycle[last_name]
        local length = index - cycle[last_name] + 1
        table.move(cycle, cycle[last_name], index, 1)
        for i=2, length//2 do
          cycle[i], cycle[length + 1 - i] = cycle[length + 1 - i], cycle[i]
        end
        error('Cycle occurred at ' .. table.concat(cycle, ' -> ', 1, length))
      end
    end
  end
  realcallbacklist[name] = list
  t[name] = list
  return list
end
```

```
})
```

Numerical codes for callback types, and name-to-value association (the table keys are strings, the values are numbers).

```
local list, data, exclusive, simple, reverselist = 1, 2, 3, 4, 5
local types   = {
  list        = list,
  data        = data,
  exclusive   = exclusive,
  simple      = simple,
  reverselist = reverselist,
}
```

Now, list all predefined callbacks with their current type, based on the LuaTEX manual version 1.01. A full list of the currently-available callbacks can be obtained using

```
\directlua{
  for i,_ in pairs(callback.list()) do
    texio.write_nl("- " .. i)
  end
}
\bye
```

in plain LuaTEX. (Some undocumented callbacks are omitted as they are to be removed.)

```
local callbacktypes = callbacktypes or {
```

Section 8.2: file discovery callbacks.

```
  find_read_file     = exclusive,
  find_write_file    = exclusive,
  find_font_file     = data,
  find_output_file   = data,
  find_format_file   = data,
  find_vf_file       = data,
  find_map_file      = data,
  find_enc_file      = data,
  find_pk_file       = data,
  find_data_file     = data,
  find_opentype_file = data,
  find_truetype_file = data,
  find_type1_file    = data,
  find_image_file    = data,

  open_read_file     = exclusive,
  read_font_file     = exclusive,
  read_vf_file       = exclusive,
  read_map_file      = exclusive,
  read_enc_file      = exclusive,
  read_pk_file       = exclusive,
  read_data_file     = exclusive,
  read_truetype_file = exclusive,
  read_type1_file    = exclusive,
  read_opentype_file = exclusive,
```

Not currently used by luatex but included for completeness. may be used by a font handler.

```
  find_cidmap_file   = data,
  read_cidmap_file   = exclusive,
```

Section 8.3: data processing callbacks.

```
  process_input_buffer  = data,
  process_output_buffer = data,
  process_jobname       = data,
```

Section 8.4: node list processing callbacks.

```
  contribute_filter      = simple,
  buildpage_filter       = simple,
  build_page_insert      = exclusive,
  pre_linebreak_filter   = list,
  linebreak_filter       = exclusive,
  append_to_vlist_filter = exclusive,
  post_linebreak_filter  = reverselist,
  hpack_filter           = list,
  vpack_filter           = list,
  hpack_quality          = exclusive,
  vpack_quality          = exclusive,
  pre_output_filter      = list,
  process_rule           = exclusive,
  hyphenate              = simple,
  ligaturing             = simple,
  kerning                = simple,
  insert_local_par       = simple,
% mlist_to_hlist         = exclusive,
  new_graf               = exclusive,
```

Section 8.5: information reporting callbacks.

```
  pre_dump             = simple,
  start_run            = simple,
  stop_run             = simple,
  start_page_number    = simple,
  stop_page_number     = simple,
  show_error_hook      = simple,
  show_warning_message = simple,
  show_error_message   = simple,
  show_lua_error_hook  = simple,
  start_file           = simple,
  stop_file            = simple,
  call_edit            = simple,
  finish_synctex       = simple,
  wrapup_run           = simple,
```

Section 8.6: PDF-related callbacks.

```
  finish_pdffile            = data,
  finish_pdfpage            = data,
  page_objnum_provider      = data,
  page_order_index          = data,
  process_pdf_image_content = data,
```

Section 8.7: font-related callbacks.

```
  define_font                     = exclusive,
  glyph_info                      = exclusive,
  glyph_not_found                 = exclusive,
```

```
  glyph_stream_provider           = exclusive,
  make_extensible                 = exclusive,
  font_descriptor_objnum_provider = exclusive,
  input_level_string              = exclusive,
  provide_charproc_data           = exclusive,

}
luatexbase.callbacktypes=callbacktypes
```

Sometimes multiple callbacks correspond to a single underlying engine level callback. Then the engine level callback should be registered as long as at least one of these callbacks is in use. This is implemented though a shared table which counts how many of the involved callbacks are currently in use. The enging level callback is registered iff this count is not 0.

We add `mlist_to_hlist` directly to the list to demonstrate this, but the handler gets added later when it is actually defined.

All callbacks in this list are treated as user defined callbacks.

```
local shared_callbacks = {
  mlist_to_hlist = {
    callback = "mlist_to_hlist",
    count = 0,
    handler = nil,
  },
}
shared_callbacks.pre_mlist_to_hlist_filter = shared_callbacks.mlist_to_hlist
shared_callbacks.post_mlist_to_hlist_filter = shared_callbacks.mlist_to_hlist
```

`callback.register` Save the original function for registering callbacks and prevent the original being used. The original is saved in a place that remains available so other more sophisticated code can override the approach taken by the kernel if desired.

```
local callback_register = callback_register or callback.register
function callback.register()
  luatexbase_error("Attempt to use callback.register() directly\n")
end
```

(*End of definition for* `callback.register`.)

5.18.2 Handlers

The handler function is registered into the callback when the first function is added to this callback's list. Then, when the callback is called, the handler takes care of running all functions in the list. When the last function is removed from the callback's list, the handler is unregistered.

More precisely, the functions below are used to generate a specialized function (closure) for a given callback, which is the actual handler.

The way the functions are combined together depends on the type of the callback. There are currently 4 types of callback, depending on the calling convention of the functions the callback can hold:

simple is for functions that don't return anything: they are called in order, all with the same argument;

data is for functions receiving a piece of data of any type except node list head (and possibly other arguments) and returning it (possibly modified): the functions are called in order, and each is passed the return value of the previous (and the other arguments untouched, if any). The return value is that of the last function;

list is a specialized variant of *data* for functions filtering node lists. Such functions may return either the head of a modified node list, or the boolean values `true` or `false`. The functions are chained the same way as for *data* except that for the following. If one function returns `false`, then `false` is immediately returned and the following functions are *not* called. If one function returns `true`, then the same head is passed to the next function. If all functions return `true`, then `true` is returned, otherwise the return value of the last function not returning `true` is used.

reverselist is a specialized variant of *list* which executes functions in inverse order.

exclusive is for functions with more complex signatures; functions in this type of callback are *not* combined: An error is raised if a second callback is registered.

Handler for `data` callbacks.

```
local function data_handler(name)
  return function(data, ...)
    for _,i in ipairs(callbacklist[name]) do
      data = i.func(data,...)
    end
    return data
  end
end
```

Default for user-defined `data` callbacks without explicit default.

```
local function data_handler_default(value)
  return value
end
```

Handler for `exclusive` callbacks. We can assume `callbacklist[name]` is not empty: otherwise, the function wouldn't be registered in the callback any more.

```
local function exclusive_handler(name)
  return function(...)
    return callbacklist[name][1].func(...)
  end
end
```

Handler for `list` callbacks.

```
local function list_handler(name)
  return function(head, ...)
    local ret
    for _,i in ipairs(callbacklist[name]) do
      ret = i.func(head, ...)
      if ret == false then
        luatexbase_warning(
          "Function ‘" .. i.description .. "’ returned false\n"
            .. "in callback ‘" .. name .."’"
         )
        return false
      end
      if ret ~= true then
```

```
        head = ret
      end
    end
    return head
  end
end
```

Default for user-defined `list` and `reverselist` callbacks without explicit default.

```
local function list_handler_default(head)
return head
end
```

Handler for `reverselist` callbacks.

```
local function reverselist_handler(name)
  return function(head, ...)
    local ret
    local callbacks = callbacklist[name]
    for i = #callbacks, 1, -1 do
      local cb = callbacks[i]
      ret = cb.func(head, ...)
      if ret == false then
        luatexbase_warning(
          "Function ‘" .. cb.description .. "’ returned false\n"
            .. "in callback ‘" .. name .."’"
         )
        return false
      end
      if ret ~= true then
        head = ret
      end
    end
    return head
  end
end
```

Handler for `simple` callbacks.

```
local function simple_handler(name)
  return function(...)
    for _,i in ipairs(callbacklist[name]) do
      i.func(...)
    end
  end
end
```

Default for user-defined `simple` callbacks without explicit default.

```
local function simple_handler_default()
end
```

Keep a handlers table for indexed access and a table with the corresponding default functions.

```
local handlers  = {
  [data]        = data_handler,
  [exclusive]   = exclusive_handler,
  [list]        = list_handler,
  [reverselist] = reverselist_handler,
  [simple]      = simple_handler,
```

```
}
local defaults = {
  [data]        = data_handler_default,
  [exclusive]   = nil,
  [list]        = list_handler_default,
  [reverselist] = list_handler_default,
  [simple]      = simple_handler_default,
}
```

5.18.3 Public functions for callback management

Defining user callbacks perhaps should be in package code, but impacts on `add_to_callback`. If a default function is not required, it may be declared as `false`. First we need a list of user callbacks.

```
local user_callbacks_defaults = {}
```

`create_callback` The allocator itself.

```
local function create_callback(name, ctype, default)
  local ctype_id = types[ctype]
  if not name  or name  == ""
  or not ctype_id
  then
    luatexbase_error("Unable to create callback:\n" ..
                     "valid callback name and type required")
  end
  if callbacktypes[name] then
    luatexbase_error("Unable to create callback ‘" .. name ..
                     "’:\ncallback is already defined")
  end
  default = default or defaults[ctype_id]
  if not default then
    luatexbase_error("Unable to create callback ‘" .. name ..
                     "’:\ndefault is required for ‘" .. ctype ..
                     "’ callbacks")
  elseif type (default) ~= "function" then
    luatexbase_error("Unable to create callback ‘" .. name ..
                     "’:\ndefault is not a function")
  end
  user_callbacks_defaults[name] = default
  callbacktypes[name] = ctype_id
end
luatexbase.create_callback = create_callback
```

(*End of definition for* `create_callback`.)

`call_callback` Call a user defined callback. First check arguments.

```
local function call_callback(name,...)
  if not name or name == "" then
    luatexbase_error("Unable to create callback:\n" ..
                     "valid callback name required")
  end
  if user_callbacks_defaults[name] == nil then
    luatexbase_error("Unable to call callback ‘" .. name
                     .. "’:\nunknown or empty")
```

```
  end
  local l = callbacklist[name]
  local f
  if not l then
    f = user_callbacks_defaults[name]
  else
    f = handlers[callbacktypes[name]](name)
  end
  return f(...)
end
luatexbase.call_callback=call_callback
```

(*End of definition for* `call_callback`.)

`add_to_callback` Add a function to a callback. First check arguments.

```
local function add_to_callback(name, func, description)
  if not name or name == "" then
    luatexbase_error("Unable to register callback:\n" ..
                     "valid callback name required")
  end
  if not callbacktypes[name] or
    type(func) ~= "function" or
    not description or
    description == "" then
    luatexbase_error(
      "Unable to register callback.\n\n"
        .. "Correct usage:\n"
        .. "add_to_callback(<callback>, <function>, <description>)"
    )
  end
```

Then test if this callback is already in use. If not, initialise its list and register the proper handler.

```
  local l = realcallbacklist[name]
  if l == nil then
    l = { }
    realcallbacklist[name] = l
```

Handle count for shared engine callbacks.

```
    local shared = shared_callbacks[name]
    if shared then
      shared.count = shared.count + 1
      if shared.count == 1 then
        callback_register(shared.callback, shared.handler)
      end
```

If it is not a user defined callback use the primitive callback register.

```
    elseif user_callbacks_defaults[name] == nil then
      callback_register(name, handlers[callbacktypes[name]](name))
    end
  end
```

Actually register the function and give an error if more than one `exclusive` one is registered.

```
  local f = {
    func        = func,
```

```
    description = description,
  }
  if callbacktypes[name] == exclusive then
    if #l == 1 then
      luatexbase_error(
        "Cannot add second callback to exclusive function\n‘" ..
        name .. "’")
    end
  end
  table.insert(l, f)
  callbacklist[name] = nil
```

Keep user informed.

```
  luatexbase_log(
    "Inserting ‘" .. description .. "’ in ‘" .. name .. "’."
  )
end
luatexbase.add_to_callback = add_to_callback
```

(*End of definition for* `add_to_callback`.)

`declare_callback_rule` Add an ordering constraint between two callback implementations

```
local function declare_callback_rule(name, desc1, relation, desc2)
  if not callbacktypes[name] or
    not desc1 or not desc2 or
    desc1 == "" or desc2 == "" then
    luatexbase_error(
      "Unable to create ordering constraint. "
        .. "Correct usage:\n"
        .. "declare_callback_rule(<callback>, <description_a>, <description_b>)"
    )
  end
  if relation == ’before’ then
    relation = nil
  elseif relation == ’after’ then
    desc2, desc1 = desc1, desc2
    relation = nil
  elseif relation == ’incompatible-warning’ or relation == ’incompatible-error’ then
  elseif relation == ’unrelated’ then
  else
    luatexbase_error(
      "Unknown relation type in declare_callback_rule"
    )
  end
  callbacklist[name] = nil
  local rules = callbackrules[name]
  if rules then
    for i, rule in ipairs(rules) do
      if rule[1] == desc1 and rule[2] == desc2 or rule[1] == desc2 and rule[2] == desc1 then
        if relation == ’unrelated’ then
          table.remove(rules, i)
        else
          rule[1], rule[2], rule.type = desc1, desc2, relation
        end
        return
```

```
      end
    end
    if relation ~= 'unrelated' then
      rules[#rules + 1] = {desc1, desc2, type = relation}
    end
  elseif relation ~= 'unrelated' then
    callbackrules[name] = {{desc1, desc2, type = relation}}
  end
end
luatexbase.declare_callback_rule = declare_callback_rule
```

(*End of definition for* `declare_callback_rule`.)

`remove_from_callback` Remove a function from a callback. First check arguments.

```
local function remove_from_callback(name, description)
  if not name or name == "" then
    luatexbase_error("Unable to remove function from callback:\n" ..
                     "valid callback name required")
  end
  if not callbacktypes[name] or
    not description or
    description == "" then
    luatexbase_error(
      "Unable to remove function from callback.\n\n"
        .. "Correct usage:\n"
        .. "remove_from_callback(<callback>, <description>)"
    )
  end
  local l = realcallbacklist[name]
  if not l then
    luatexbase_error(
      "No callback list for `" .. name .. "'\n")
  end
```

Loop over the callback's function list until we find a matching entry. Remove it and check if the list is empty: if so, unregister the callback handler.

```
  local index = false
  for i,j in ipairs(l) do
    if j.description == description then
      index = i
      break
    end
  end
  if not index then
    luatexbase_error(
      "No callback `" .. description .. "' registered for `" ..
      name .. "'\n")
  end
  local cb = l[index]
  table.remove(l, index)
  luatexbase_log(
    "Removing  `" .. description .. "' from `" .. name .. "'."
  )
  if #l == 0 then
    realcallbacklist[name] = nil
```

```
    callbacklist[name] = nil
    local shared = shared_callbacks[name]
    if shared then
      shared.count = shared.count - 1
      if shared.count == 0 then
        callback_register(shared.callback, nil)
      end
    elseif user_callbacks_defaults[name] == nil then
      callback_register(name, nil)
    end
  end
  return cb.func,cb.description
end
luatexbase.remove_from_callback = remove_from_callback
```

(*End of definition for* `remove_from_callback`.)

`in_callback` Look for a function description in a callback.

```
local function in_callback(name, description)
  if not name
    or name == ""
    or not realcallbacklist[name]
    or not callbacktypes[name]
    or not description then
      return false
  end
  for _, i in pairs(realcallbacklist[name]) do
    if i.description == description then
      return true
    end
  end
  return false
end
luatexbase.in_callback = in_callback
```

(*End of definition for* `in_callback`.)

`disable_callback` As we subvert the engine interface we need to provide a way to access this functionality.

```
local function disable_callback(name)
  if(realcallbacklist[name] == nil) then
    callback_register(name, false)
  else
    luatexbase_error("Callback list for " .. name .. " not empty")
  end
end
luatexbase.disable_callback = disable_callback
```

(*End of definition for* `disable_callback`.)

`callback_descriptions` List the descriptions of functions registered for the given callback. This will sort the list if necessary.

```
local function callback_descriptions (name)
  local d = {}
  if not name
    or name == ""
```

```
    or not realcallbacklist[name]
    or not callbacktypes[name]
    then
    return d
  else
  for k, i in pairs(callbacklist[name]) do
    d[k]= i.description
    end
  end
  return d
end
luatexbase.callback_descriptions =callback_descriptions
```

(*End of definition for* `callback_descriptions`.)

`uninstall` Unlike at the TEX level, we have to provide a back-out mechanism here at the same time as the rest of the code. This is not meant for use by anything other than latexrelease: as such this is *deliberately* not documented for users!

```
local function uninstall()
  module_info(
    "luatexbase",
    "Uninstalling kernel luatexbase code"
  )
  callback.register = callback_register
  luatexbase = nil
end
luatexbase.uninstall = uninstall
```

(*End of definition for* `uninstall`.)

`mlist_to_hlist` To emulate these callbacks, the "real" `mlist_to_hlist` is replaced by a wrapper calling the wrappers before and after.

```
create_callback('pre_mlist_to_hlist_filter', 'list')
create_callback('mlist_to_hlist', 'exclusive', node.mlist_to_hlist)
create_callback('post_mlist_to_hlist_filter', 'reverselist')
function shared_callbacks.mlist_to_hlist.handler(head, display_type, need_penalties)
  local current = call_callback("pre_mlist_to_hlist_filter", head, display_type, need_penalti
  if current == false then
    flush_list(head)
    return nil
  end
  current = call_callback("mlist_to_hlist", current, display_type, need_penalties)
  local post = call_callback("post_mlist_to_hlist_filter", current, display_type, need_penalt
  if post == false then
    flush_list(current)
    return nil
  end
  return post
end
```

(*End of definition for* `mlist_to_hlist`.)

```
⟨/lua⟩
```

Reset the catcode of `@`.

```
⟨tex⟩\catcode`\@=\etatcatcode\relax
```

File 05
ltexpl.dtx

1 expl3-dependent code

1.1 Loader

\@kernel@after@enddocument
\@kernel@after@enddocument@afterlastpage

These two kernel hooks are used by the shipout code. They are defined earlier here because the lthooks code adds material to them.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩  {kernel@enddocument hooks}{Define several kernel hooks}
```

We only initialize these kernel hooks if they are not already existing. Otherwise they would be set to \@empty on rollback which would be wrong because code that has been added to them may still have to be executed in the rollback situation . Instead code that writes to them needs to handle the rollback as needed. It is likely that we have to change that approach in the future, but for now it should do. (It is enough to test only for the existence of one hook, as all got added at the same time.)

```
\ifx\@kernel@after@enddocument\@undefined
  \let \@kernel@after@enddocument               \@empty
  \let \@kernel@after@enddocument@afterlastpage \@empty
```

For the similar reasons we also define those that are used in \document because they too get material added to in early modules.

\@kernel@before@begindocument
\@kernel@after@begindocument

```
  \let \@kernel@before@begindocument \@empty
  \let \@kernel@after@begindocument  \@empty
\fi
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {kernel@enddocument hooks}{Define several kernel hooks}
⟨latexrelease⟩\let\@kernel@after@enddocument\@undefined
⟨latexrelease⟩\let\@kernel@after@enddocument@afterlastpage\@undefined
⟨latexrelease⟩\let\@kernel@before@begindocument\@undefined
⟨latexrelease⟩\let\@kernel@after@begindocument\@undefined
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* \@kernel@after@enddocument *and others.*)

First define some blank commands, so that in case something goes wrong while loading expl3, we won't get strange Undefined control sequence errors.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩            {\@expl@sys@load@backend@@}{Roll forward support}%
\def\reserved@a#1{\ifdefined#1\else\def#1{}\fi}
\reserved@a\@expl@sys@load@backend@@
\reserved@a\@expl@push@filename@@
\reserved@a\@expl@push@filename@aux@@
\reserved@a\@expl@pop@filename@@
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
```

Create a hook for last-minute expl3 material.

```
⟨*2ekernel⟩
\def\@expl@finalise@setup@@{}
⟨/2ekernel⟩
```

Now define some basics to support loading expl3. These macros can be defined here safely, because they are redefined later on by the kernel, so we define simpler versions just to suit our needs.

```
⟨*2ekernel⟩
\long\def\@gobble#1{}
\long\def\@firstofone#1{#1}
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}
\long\def\IfFileExists#1{%
  \openin\@inputcheck"#1" %
  \ifeof\@inputcheck
    \expandafter\@secondoftwo
  \else
    \closein\@inputcheck
    \expandafter\@firstoftwo
  \fi}
\long\def\@ifnextchar#1#2#3{%
  \let\reserved@d=#1%
  \def\reserved@a{#2}%
  \def\reserved@b{#3}%
  \futurelet\@let@token\@ifnch}
\def\@ifnch{%
  \ifx\@let@token\reserved@d
    \expandafter\reserved@a
  \else
    \expandafter\reserved@b
  \fi}
⟨/2ekernel⟩
```

If we are doing a rollback with a format containing expl3 we aren't reloading it as that creates havoc. This may need a refined version!

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {expl3}{Pre-load expl3}%
\expandafter\ifx\csname tex\string _let:D\endcsname\relax
  \expandafter\@firstofone
\else
  \GenericInfo{}{Skipping: expl3 code already part of the format}%
⟨2ekernel⟩   \expandafter\endinput
⟨latexrelease⟩   \expandafter\@gobble
\fi
```

Check for the required primitive/engine support and the existence of a loader.

```
  {%
    \IfFileExists{expl3.ltx}
      {%
        \ifnum0%
          \ifdefined\pdffilesize 1\fi
          \ifdefined\filesize 1\fi
          \ifdefined\luatexversion\ifnum\luatexversion>94 1\fi\fi
```

```
          \ifdefined\kanjiskip 1\fi
            >0 %
          \expandafter\@firstofone
        \else
```

In 2ekernel mode, an error is fatal and building the format is aborted. Use `\batchmode \read -1 to \tokenlist`, which errors with

```
! Emergency stop. (cannot \read from terminal in nonstop modes)
```

and aborts the TEX run. In latexrelease mode, raise an error and do nothing. Both ways, the error message shows the minimum expl3 engine requirements.

```
⟨2ekernel⟩          \def~{ }\def\MessageBreak{^^J~~~~~~~~~~~~~~~}%
⟨2ekernel⟩          \errmessage{LaTeX Error:
⟨latexrelease⟩         \@latex@error{%
            LaTeX requires the e-TeX primitives and additional\MessageBreak
            functionality available in the engines:\MessageBreak
              - pdfTeX v1.40\MessageBreak
              - XeTeX v0.99992\MessageBreak
              - LuaTeX v0.95\MessageBreak
              - e-(u)pTeX mid-2012\MessageBreak
            or later%
⟨latexrelease⟩         }\@ehd \expandafter\@gobble
⟨2ekernel⟩          }\batchmode \read -1 to \reserved@a
        \fi
      }
      {%
⟨*2ekernel⟩
        \errmessage{LaTeX requires expl3}%
        \batchmode \read -1 to \reserved@a
⟨/2ekernel⟩
```

We do not support a roll forward across 2019. You need to start with 2019 if you want to get to 2020 or beyond.

```
⟨*latexrelease⟩
        \@latex@warning@no@line
          {You need a format that already contains a recent\MessageBreak
           expl3 as part of the kernel, e.g. at least a kernel\MessageBreak
           from 2019 to roll forward to that date!\MessageBreak
           --- I'm giving up!\MessageBreak\MessageBreak
           Note that manually loading the expl3 package\MessageBreak
           from your distribution is not enough}%
        \batchmode \read -1 to \reserved@a
⟨/latexrelease⟩
      }%
      {\input expl3.ltx }%
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
```

To support roll-forward for the case where xparse is fully integrated into the kernel, we do not need to repeat the complex test above as we can simply look for the marker command.

```
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {expl3}{Pre-load expl3}%
```

```
⟨latexrelease⟩\IfFileExists{expl3.ltx}
⟨latexrelease⟩  {%
⟨latexrelease⟩    \ifnum0%
⟨latexrelease⟩      \ifdefined\pdffilesize 1\fi
⟨latexrelease⟩      \ifdefined\filesize 1\fi
⟨latexrelease⟩      \ifdefined\luatexversion\ifnum\luatexversion>94 1\fi\fi
⟨latexrelease⟩      >0 %
⟨latexrelease⟩    \else
⟨latexrelease⟩      \message{Skipping expl3-dependent extensions}
⟨latexrelease⟩      \expandafter\@gobbletwo
⟨latexrelease⟩    \fi
⟨latexrelease⟩  }
⟨latexrelease⟩  {%
⟨latexrelease⟩    \message{Skipping expl3-dependent extensions}%
⟨latexrelease⟩    \@gobbletwo
⟨latexrelease⟩  }%
⟨latexrelease⟩\input{expl3.ltx}
⟨latexrelease⟩\EndIncludeInRelease
```

Now in `latexrelease` mode, redefine a few commands to avoid "already defined" errors.

```
⟨latexrelease⟩\@ifundefined{ExplSyntaxOff}{}{\latexrelease@postltexpl}
```

1.2 Using expl3 code

In order to ease the implementation of some new features in LaTeX 2_ε we may (temporarily) use some coding based on the expl3-code. Such macros will eventually vanish and may be changed unannounced. They are there for internal use in the LaTeX 2_ε kernel and are not meant to be used in third-party packages. These macros will always have the `@expl@` prefix in their name.

The rest of the name matches the expl3 name but with all underscores replaced by `@`s and the `:` replaced by `@@`, e.g.,

```
\cs_new_eq:NN \@expl@tl@trim@spaces@apply@@nN \tl_trim_spaces_apply:nN
```

if that expl3 command is needed in places that are others coded in LaTeX 2_ε conventions.

In this file, each release of LaTeX adds an `\IncludeInRelease` block, in which the macros copied for that release were defined. In case a rollback is requested, the entire block is changed.

Each macro copied has a `\changes` entry to explain when and why it was copied, so that further to that may spot it easily.

Here `\cs_gset_eq:NN` is used, instead of the `new` variant because if different releases use that same name for different purposes, each can copy the macro without worrying about redefinitions.

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\@expl@cs@to@str@@N}%
⟨latexrelease⟩        {expl3 macros added for the 2020-10-01 release}%
```

The expl3 activation needs to be inside the release guards as otherwise rolling forward is broken in old kernels that do not have expl3 loaded.

```
\ExplSyntaxOn
```

```
\cs_gset_eq:NN \@expl@cs@to@str@@N \cs_to_str:N
\cs_gset_eq:NN \@expl@str@if@eq@@nnTF \str_if_eq:nnTF
```

```
\cs_gset_eq:NN \@expl@cs@prefix@spec@@N \cs_prefix_spec:N
\cs_if_exist:NTF \cs_parameter_spec:N
  { \cs_gset_eq:NN \@expl@cs@parameter@spec@@N \cs_parameter_spec:N }
  { \cs_gset_eq:NN \@expl@cs@parameter@spec@@N \cs_argument_spec:N }
\cs_gset_eq:NN \__kernel_cs_parameter_spec:N \@expl@cs@parameter@spec@@N
\cs_gset_eq:NN \@expl@cs@replacement@spec@@N \cs_replacement_spec:N

\cs_gset_eq:NN \@expl@str@map@function@@NN \str_map_function:NN
\cs_gset_eq:NN \@expl@char@generate@@nn \char_generate:nn

\ExplSyntaxOff
```

Here we can't assume that expl3 is available. It will be if we roll back but if this code is executed rolling forward it needs to be pure 2e.

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@expl@cs@to@str@@N}%
⟨latexrelease⟩        {expl3 macros added for the 2020-10-01 release}%
⟨latexrelease⟩\let \@expl@cs@to@str@@N \@undefined
⟨latexrelease⟩\let \@expl@str@if@eq@@nnTF \@undefined
⟨latexrelease⟩\let \@expl@cs@prefix@spec@@N \@undefined
⟨latexrelease⟩\let \@expl@cs@parameter@spec@@N \@undefined
⟨latexrelease⟩\let \@expl@cs@replacement@spec@@N \@undefined
⟨latexrelease⟩\let \@expl@str@map@function@@NN \@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
```

2 Document-level command names for expl3 functions

Current home for L3 programing layer functions that we make directly available at the document level. This section may need to be moved later (after `\NewDocumentCommand` is defined in case we want to use that in the setup).

`\fpeval` The expandable command `\fpeval` takes as its argument a floating point expression and produces a result using the normal rules of mathematics. As this command is expandable it can be used where TEX requires a number and for example within a low-level `\edef` operation to give a purely numerical result. See `usrguide3` for further explanation.

`\inteval` `\dimeval` `\skipeval` The expandable command `\inteval` takes as its argument an integer expression and produces a result using the normal rules of mathematics. The operations recognised are `+`, `-`, `*` and `/` plus parentheses. Division occurs with *rounding*, and ties are rounded away from zero. As this command is expandable it can be used where TEX requires a number and for example within a low-level `\edef` operation to give a purely numerical result. See `usrguide3` for further explanation. `\dimeval` and `\skipeval` are similar, but generate fixed and rubber length values, respectively.

`\fpeval` `\inteval` `\dimeval` `\skipeval` A document level wrapper around the code level function for floating point calculations.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                  {\fpeval}{fp and int calculations}%
\ExplSyntaxOn
\cs_new_eq:NN \fpeval \fp_eval:n
```

And a few more, this time wrappers around the eTEX primitives.

```
\cs_new_eq:NN \inteval \int_eval:n
\cs_new_eq:NN \dimeval  \dim_eval:n
\cs_new_eq:NN \skipeval \skip_eval:n
\ExplSyntaxOff
```

(*End of definition for* `\fpeval` *and others.*)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\fpeval}{fp and int calculations}%
⟨latexrelease⟩
⟨latexrelease⟩\let\fpeval\@undefined
⟨latexrelease⟩\let\inteval\@undefined
⟨latexrelease⟩\let\dimeval\@undefined
⟨latexrelease⟩\let\skipeval\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

`\UseName` `\ExpandArgs` When declaring new commands with `\NewDocumentCommand` or `\NewCommandCopy` or similar, it is sometimes necessary to "construct" the csname. As a general mechanism the L3 programming layer has `\exp_args:N...` for this, but there is no mechanism for it if `\ExplSyntaxOn` is not active. We therefore offer a few of these commands also with CamelCase names.

`\UseName` `\ExpandArgs` A document wrapper for changing arguments to cs names for use with `\NewDocumentCommand` and similar functions.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                 {\ExpandArgs}{Some pre-expansion commands}%
\ExplSyntaxOn
\cs_new_eq:NN \UseName \use:c
\cs_new:Npn \ExpandArgs #1
  {
    \cs_if_exist_use:cF { exp_args:N #1 }
      { \msg_expandable_error:nnn { kernel } { unknown-arg-expansion } {#1} }
  }
\msg_new:nnn { kernel } { unknown-arg-expansion }
  { Unknown~arg~expansion~"#1" }
\ExplSyntaxOff
```

(*End of definition for* `\UseName` *and* `\ExpandArgs`*.*)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\ExpandArgs}{Some pre-expansion commands}%
⟨latexrelease⟩
⟨latexrelease⟩\let\UseName\@undefined
⟨latexrelease⟩\let\ExpandArgs\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

`\IfExplAtLeastTF` A pretty simple wrapper.

\IfExplAtLeastTF

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                  {\IfExplAtLeastTF}{Test for expl3 date}%
\def\IfExplAtLeastTF{\@ifl@t@r\ExplLoaderFileDate}
```

(*End of definition for* \IfExplAtLeastTF.)

We make sure the command is always available.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\IfExplAtLeastTF}{Test for expl3 date}%
⟨latexrelease⟩
⟨latexrelease⟩\def\IfExplAtLeastTF{\@ifl@t@r\ExplLoaderFileDate}
⟨latexrelease⟩\EndIncludeInRelease
```

File 06
ltdefns.dtx

1 Definitions

This section contains commands used in defining other macros.

```
⟨*2ekernel⟩
```

1.1 Initex initializations

`\two@digits` Prefix a number less than 10 with '0'.

```
\def\two@digits#1{\ifnum#1<10 0\fi\number#1}
```

(*End of definition for* `\two@digits`.)

`\typeout` Display something on the terminal.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\typeout}{Allow "par" in \typeout}%
\protected\long\def\typeout#1{\begingroup
  \set@display@protect
  \def\par{^^J^^J}%
  \immediate\write\@unused{#1}\endgroup}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\typeout}{Allow "par" in \typeout}%
⟨latexrelease⟩
⟨latexrelease⟩\def\typeout#1{\begingroup\set@display@protect
⟨latexrelease⟩    \immediate\write\@unused{#1}\endgroup}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\typeout`.)

`\newlinechar` A char to be used as new-line in output to files.

```
\newlinechar`\^^J
```

(*End of definition for* `\newlinechar`.)

1.2 Saved versions of TEX primitives

The TeX primitive `\foo` is saved as `\@@foo`. The following primitives are handled in this way:

`\@@par`

```
\let\@@par=\par
%\let\@@input=\input    %%% moved earlier
%\let\@@end=\end        %%%
```

(*End of definition for* `\@@par`.)

\@@hyph Save original primitive definition.

```
\let\@@hyph=\-
```

(*End of definition for* `\@@hyph`.)

\@@italiccorr Save the original italic correction.

```
\let\@@italiccorr=\/
```

(*End of definition for* `\@@italiccorr`.)

\@height
\@depth
\@width
\@minus
\@plus

The following definitions save token space. E.g., using `\@height` instead of height saves 5 tokens at the cost in time of one macro expansion.

```
\def\@height{height} \def\@depth{depth} \def\@width{width}
\def\@minus{minus}
\def\@plus{plus}
```

The next one is another 100 tokens worth.

```
\def\hb@xt@{\hbox to}
```

(*End of definition for* `\@height` *and others.*)

```
\message{hacks,}
```

\hb@xt@

1.3 Command definitions

This section defines the following commands:

\@namedef {⟨*NAME*⟩}
Expands to `\def\`⟨*NAME*⟩, except name can contain any characters.

\@nameuse {⟨*NAME*⟩}
Expands to `\`⟨*NAME*⟩.

\@ifnextchar X{⟨*YES*⟩}{⟨*NO*⟩}
Expands to ⟨*YES*⟩ if next character is an 'X', and to ⟨*NO*⟩ otherwise. (Uses `\reserved@a`–`\reserved@c`.) NOTE: GOBBLES ANY SPACE FOLLOWING IT.

\@ifstar {⟨*YES*⟩}{⟨*NO*⟩}
Gobbles following spaces and then tests if next the character is a '*'. If it is, then it gobbles the '*' and expands to ⟨*YES*⟩, otherwise it expands to ⟨*NO*⟩.

\@dblarg {⟨*CMD*⟩}{⟨*ARG*⟩}
Expands to `\{`⟨*CMD*⟩`}[`⟨*ARG*⟩`]{`⟨*ARG*⟩`}`. Use `\@dblarg\CS` when `\CS` takes arguments `[ARG1]{ARG2}`, where default is `ARG1` = `ARG2`.

\@ifundefined {⟨*NAME*⟩}{⟨*YES*⟩}{⟨*NO*⟩}
: If `\NAME` is undefined then it executes ⟨*YES*⟩, otherwise it executes ⟨*NO*⟩. More precisely, true if `\NAME` either undefined or = `\relax`.

\@ifdefinable `\NAME{`⟨*YES*⟩`}` Executes ⟨*YES*⟩ if the user is allowed to define `\NAME`, otherwise it gives an error. The user can define `\NAME` if `\@ifundefined{NAME}` is true, '`NAME`' ≠ '`relax`' and the first three letters of '`NAME`' are not '`end`', and if `\endNAME` is not defined.

\newcommand *{⟨\FOO⟩}[⟨*i*⟩]{⟨*TEXT*⟩}
User command to define `\FOO` to be a macro with i arguments (i = 0 if missing) having the definition ⟨*TEXT*⟩. Produces an error if `\FOO` already defined.

Normally the command is defined to be `\long` (ie it may take multiple paragraphs in its argument). In the star-form, the command is not defined as `\long` and a blank line in any argument to the command would generate an error.

\renewcommand *{⟨\FOO⟩}[⟨*i*⟩]{⟨*TEXT*⟩}

Same as \newcommand, except it checks if \FOO already defined.

\newenvironment *{⟨*FOO*⟩}[⟨*i*⟩]{⟨*DEF1*⟩}{⟨*DEF2*⟩}

equivalent to:

\newcommand{\FOO}[i]{DEF1} \def{\endFOO}{DEF2}

(or the appropriate star forms).

\renewenvironment

Obvious companion to \newenvironment.

\@cons : See description of \output routine.

\@car \@car T1 T2 ... Tn\@nil == T1 (unexpanded)

\@cdr \@cdr T1 T2 ... Tn\@nil == T2 ... Tn (unexpanded)

\typeout {⟨*message*⟩}

Produces a warning message on the terminal.

\typein {⟨*message*⟩}

Types message, asks the user to type in a command, then executes it

\typein [⟨\CS⟩]{⟨*MSG*⟩}

Same as above, except defines \CS to be the input instead of executing it.

\typein

```
\def\typein{%
  \let\@typein\relax
  \@testopt\@xtypein\@typein}
```

```
\ifx\directlua\@undefined
```

```
\def\@xtypein[#1]#2{%
 \typeout{#2}%
 \advance\endlinechar\@M
 \read\@inputcheck to#1%
 \advance\endlinechar-\@M
 \@typein}%
```

```
\else
```

```
\def\@xtypein[#1]#2{%
  \typeout{#2}%
  \begingroup \endlinechar\m@ne
  \read\@inputcheck to#1%
  \expandafter\endgroup
  \expandafter\def\expandafter#1\expandafter{#1}%
  \@typein}%
```

```
\fi
```

(*End of definition for* \typein.)

\@namedef

```
\def\@namedef#1{\expandafter\def\csname #1\endcsname}
```

(*End of definition for* \@namedef.)

\@nameuse

```
\def\@nameuse#1{\csname #1\endcsname}
```

(*End of definition for* \@nameuse.)

\@cons

```
\def\@cons#1#2{\begingroup\let\@elt\relax\xdef#1{#1\@elt #2}\endgroup}
```

(*End of definition for* \@cons.)

\@car
\@cdr

```
\def\@car#1#2\@nil{#1}
\def\@cdr#1#2\@nil{#2}
```

(*End of definition for* \@car *and* \@cdr.)

\@carcube

\@carcube T1 ... Tn\@nil = T1 T2 T3 , $n > 3$

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\@carcube}{Make \@carcube long}%
⟨*2ekernel | latexrelease⟩
\long\def\@carcube#1#2#3#4\@nil{#1#2#3}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@carcube}{Undo: Make \@carcube long}%
⟨latexrelease⟩\def\@carcube#1#2#3#4\@nil{#1#2#3}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \@carcube.)

\@onlypreamble
\@preamblecmds

This macro adds its argument to the list of commands stored in \@preamblecmds to be disabled after \begin{document}. These commands are redefined to generate \@notprerr at this point.

```
\def\@preamblecmds{}
\def\@onlypreamble#1{%
  \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
       \@preamblecmds\do#1}}
\@onlypreamble\@onlypreamble
\@onlypreamble\@preamblecmds
```

(*End of definition for* \@onlypreamble *and* \@preamblecmds.)

\@star@or@long

Look ahead for a *. If present reset \l@ngrel@x so that the next definition, #1, will be non-long.

```
\def\@star@or@long#1{%
  \@ifstar
   {\let\l@ngrel@x\relax#1}%
   {\let\l@ngrel@x\long#1}}
```

(*End of definition for* \@star@or@long.)

\l@ngrel@x

This is either \relax or \long depending on whether the *-form of a definition command is being executed.

```
\let\l@ngrel@x\relax
```

(*End of definition for* \l@ngrel@x.)

\newcommand

User level \newcommand.

```
\def\newcommand{\@star@or@long\new@command}
```

\new@command

```
\def\new@command#1{%
  \@testopt{\@newcommand#1}0}
```

(*End of definition for* \newcommand *and* \new@command.)

\@newcommand
\@argdef
\@xargdef

Handling arguments for \newcommand.

```
\def\@newcommand#1[#2]{%
  \kernel@ifnextchar [{\@xargdef#1[#2]}%
                {\@argdef#1[#2]}}
```

Define #1 if it is definable.

Both here and in \@xargdef the replacement text is absorbed as an argument because if we are not allowed to make the definition we have to get rid of it completely.

```
\long\def\@argdef#1[#2]#3{%
   \@ifdefinable #1{\@yargdef#1\@ne{#2}{#3}}}
```

Handle the second optional argument.

```
\long\def\@xargdef#1[#2][#3]#4{%
  \@ifdefinable#1{%
```

Define the actual command to be:
\def\foo{\@protected@testopt\foo\\foo{default}}
where \\foo is a csname generated from applying \csname and \string to \foo, ie the actual name contains a backslash and therefore can't clash easily with existing command names. "Default" is the contents of the second optional argument of (re)newcommand.

```
     \expandafter\def\expandafter#1\expandafter{%
          \expandafter
          \@protected@testopt
          \expandafter
          #1%
          \csname\string#1\endcsname
          {#3}}%
```

Now we define the internal macro ie \\foo which is supposed to pick up all arguments (optional and mandatory).

```
        \expandafter\@yargdef
          \csname\string#1\endcsname
           \tw@
           {#2}%
           {#4}}}
```

(*End of definition for* \@newcommand*,* \@argdef*, and* \@xargdef.)

\@testopt

This macro encapsulates the most common call to \@ifnextchar, saving several tokens each time it is used in the definition of a command with an optional argument. #1 The code to execute in the case that there is a [need not be a single token but can be any sequence of commands that 'expects' to be followed by [. If this command were only used in \newcommand definitions then #1 would be a single token and the braces could be omitted from {#1} in the definition below, saving a bit of memory.

```
\long\def\@testopt#1#2{%
  \kernel@ifnextchar[{#1}{#1[{#2}]}}
```

(*End of definition for* \@testopt.)

\@protected@testopt Robust version of \@testopt. The extra argument (#1) must be a single token. If protection is needed the call expands to \protect applied to this token, and the 2nd and 3rd arguments are discarded (by \@x@protect). Otherwise \@testopt is called on the 2nd and 3rd arguments.

This method of making commands robust avoids the need for using up two csnames per command, the price is the extra expansion time for the \ifx test.

```
\def\@protected@testopt#1{%
  \ifx\protect\@typeset@protect
    \expandafter\@testopt
  \else
    \@x@protect#1%
  \fi}
```

(*End of definition for* \@protected@testopt.)

\@yargdef \@yargd@f These generate a primitive argument specification, from a LaTeX [⟨*digit*⟩] form; in fact ⟨*digit*⟩ can be anything such that \number ⟨*digit*⟩ is single digit.

Reorganised slightly so that \renewcommand{\reserved@a}[1]{foo} works. I am not sure this is worth it, as a following \newcommand would over-write the definition of \reserved@a.

Recall that LaTeX2.09 goes into an infinite loop with
\renewcommand[1]{\@tempa}{foo}
(DPC 6 October 93).

Reorganised again (DPC 1999). Rather than make a loop to construct the argument spec by counting, just extract the required argument spec by using a delimited argument (delimited by the digit). This is faster and uses less tokens. The coding is slightly odd to preserve the old interface (using #2 = \tw@ as the flag to surround the first argument with []). But the new method did not allow for the number of arguments #3 not being given as an explicit digit; hence (further expansion of this argument and use of) \number was added later in 1999.

It is not clear why these are still \long.

```
\long \def \@yargdef #1#2#3{%
  \ifx#2\tw@
    \def\reserved@b##11{[####1]}%
  \else
    \let\reserved@b\@gobble
  \fi
  \expandafter
    \@yargd@f \expandafter{\number #3}#1%
}
```

```
\long \def \@yargd@f#1#2{%
  \def \reserved@a ##1#1##2##{%
    \expandafter\def\expandafter#2\reserved@b ##1#1%
    }%
  \l@ngrel@x \reserved@a 0##1##2##3##4##5##6##7##8##9###1%
}
```

(*End of definition for* \@yargdef *and* \@yargd@f.)

\@reargdef

```
\long\def\@reargdef#1[#2]{%
  \@yargdef#1\@ne{#2}}
```

(*End of definition for* `\@reargdef`.)

`\renewcommand` Check the command name is already used. If not give an error message. Then temporarily disable `\@ifdefinable` then call `\newcommand`. (Previous version `\let#1=\relax` but this does not work too well if `#1` is `\@temp`*a–e*.)

```
\def\renewcommand{\@star@or@long\renew@command}
```

`\renew@command`

```
\def\renew@command#1{%
  \begingroup \escapechar\m@ne\xdef\@gtempa{{\string#1}}\endgroup
  \expandafter\@ifundefined\@gtempa
     {\@latex@error{Command \string#1 undefined}\@ehc}%
     \relax
  \let\@ifdefinable\@rc@ifdefinable
  \new@command#1}
```

(*End of definition for* `\renewcommand` *and* `\renew@command`.)

`\@ifdefinable` `\@@ifdefinable` `\@rc@ifdefinable` Test if user is allowed to define a command.

```
\long\def\@ifdefinable #1#2{%
      \edef\reserved@a{\expandafter\@gobble\string #1}%
     \@ifundefined\reserved@a
         {\edef\reserved@b{\expandafter\@carcube \reserved@a xxx\@nil}%
          \ifx \reserved@b\@qend \@notdefinable\else
            \ifx \reserved@a\@qrelax \@notdefinable\else
              #2%
            \fi
          \fi}%
         \@notdefinable}
```

Saved definition of `\@ifdefinable`.

```
\let\@@ifdefinable\@ifdefinable
```

Version of `\@ifdefinable` for use with `\renewcommand`. Does not do the check this time, but restores the normal definition.

```
\long\def\@rc@ifdefinable#1#2{%
  \let\@ifdefinable\@@ifdefinable
  #2}
```

(*End of definition for* `\@ifdefinable`, `\@@ifdefinable`, *and* `\@rc@ifdefinable`.)

`\newenvironment` Define a new user environment. `#1` is the environment name. `#2#` Grabs all the tokens up to the first `{`. These will be any optional arguments. They are not parsed at this point, but are just passed to `\@newenv` which will eventually call `\newcommand`. Any optional arguments will then be parsed by `\newcommand` as it defines the command that executes the 'begin code' of the environment.

This `#2#` trick removed with version 1.2i as it fails if a `{` occurs in the optional argument. Now use `\@ifnextchar` directly.

```
\def\newenvironment{\@star@or@long\new@environment}
```

`\new@environment`

```
\def\new@environment#1{%
  \@testopt{\@newenva#1}0}
```

\@newenva

```
\def\@newenva#1[#2]{%
   \kernel@ifnextchar [{\@newenvb#1[#2]}{\@newenv{#1}{[#2]}}}
```

\@newenvb

```
\def\@newenvb#1[#2][#3]{\@newenv{#1}{[#2][{#3}]}}
```

(*End of definition for* `\newenvironment` *and others.*)

\renewenvironment

Redefine an environment. For `\renewenvironment` disable `\@ifdefinable` and then call `\newenvironment`. It is OK to `\let` the argument to `\relax` here as there should not be a `@temp`... environment.

```
\def\renewenvironment{\@star@or@long\renew@environment}
```

\renew@environment

```
\def\renew@environment#1{%
  \@ifundefined{#1}%
     {\@latex@error{Environment #1 undefined}\@ehc
     }\relax
  \expandafter\let\csname#1\endcsname\relax
  \expandafter\let\csname end#1\endcsname\relax
  \new@environment{#1}}
```

(*End of definition for* `\renewenvironment` *and* `\renew@environment`.)

\@newenv

The internal version of `\newenvironment`.

Call `\newcommand` to define the ⟨*begin-code*⟩ for the environment. `\def` is used for the ⟨*end-code*⟩ as it does not take arguments. (but may contain `\par`s)

Make sure that an attempt to define a 'graf' or 'group' environment fails by temporarily letting the undefined `\...` (begin code) to the definition of `\end...` and as a result we get an error if that has a definition.

```
\long\def\@newenv#1#2#3#4{%
  \@ifundefined{#1}%
    {\expandafter\let\csname#1\expandafter\endcsname
                          \csname end#1\endcsname}%
    \relax
  \expandafter\new@command
     \csname #1\endcsname#2{#3}%
     \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}}
```

(*End of definition for* `\@newenv`.)

\newif

And here's a different sort of allocation: For example, `\newif\iffoo` creates `\footrue`, `\foofalse` to go with `\iffoo`.

```
\def\newif#1{%
  \count@\escapechar \escapechar\m@ne
    \let#1\iffalse
    \@if#1\iftrue
    \@if#1\iffalse
  \escapechar\count@}
```

\@if

```
\def\@if#1#2{%
  \expandafter\def\csname\expandafter\@gobbletwo\string#1%
                     \expandafter\@gobbletwo\string#2\endcsname
                        {\let#1#2}}
```

(*End of definition for* `\newif` *and* `\@if`.)

\providecommand

`\providecommand` takes the same arguments as `\newcommand`, but discards them if `#1` is already defined, Otherwise it just acts like `\newcommand`. This implementation currently leaves any discarded definition in `\reserved@a` (and possibly `\\reserved@a`) this wastes a bit of space, but it will be reclaimed as soon as these scratch macros are redefined.

```
\def\providecommand{\@star@or@long\provide@command}
```

\provide@command

```
\def\provide@command#1{%
  \begingroup
    \escapechar\m@ne\xdef\@gtempa{{\string#1}}%
  \endgroup
  \expandafter\@ifundefined\@gtempa
    {\def\reserved@a{\new@command#1}}%
    {\def\reserved@a{\renew@command\reserved@a}}%
   \reserved@a}%
```

(*End of definition for* `\providecommand` *and* `\provide@command`.)

\CheckCommand

`\CheckCommand` takes the same arguments as `\newcommand`. If the command already exists, with the same definition, then nothing happens, otherwise a warning is issued. Useful for checking the current state before a macro package starts redefining things. Currently two macros are considered to have the same definition if they are the same except for different default arguments. That is, if the old definition was: `\newcommand\xxx[2][a]{(#1)(#2)}` then `\CheckCommand\xxx[2][b]{(#1)(#2)}` would *not* generate a warning, but, for instance `\CheckCommand\xxx[2]{(#1)(#2)}` would.

```
\def\CheckCommand{\@star@or@long\check@command}
```

`\CheckCommand` is only available in the preamble part of the document.

```
\@onlypreamble\CheckCommand
```

\check@command

```
\def\check@command#1#2#{\@check@c#1{#2}}
\@onlypreamble\check@command
```

(*End of definition for* `\CheckCommand` *and* `\check@command`.)

\@check@c

`\CheckCommand` itself just grabs all the arguments we need, without actually looking for `[` optional argument forms. Now define `\reserved@a`. If `\\reserved@a` is then defined, compare it with the "`\#1`' otherwise compare `\reserved@a` with `#1`.

```
\long\def\@check@c#1#2#3{%
  \expandafter\let\csname\string\reserved@a\endcsname\relax
  \renew@command\reserved@a#2{#3}%
  \@ifundefined{\string\reserved@a}%
   {\@check@eq#1\reserved@a}%
```

```
  {\expandafter\@check@eq
          \csname\string#1\expandafter\endcsname
          \csname\string\reserved@a\endcsname}}
\@onlypreamble\@check@c
```

(*End of definition for* `\@check@c`.)

`\@check@eq` Complain if `#1` and `#2` are not `\ifx` equal.

```
\def\@check@eq#1#2{%
  \ifx#1#2\else
     \@latex@warning@no@line
               {Command \noexpand#1 has
                changed.\MessageBreak
                Check if current package is valid}%
  \fi}
\@onlypreamble\@check@eq
```

(*End of definition for* `\@check@eq`.)

`\@gobble` `\@gobbletwo` `\@gobblethree` `\@gobblefour` The `\@gobble` macro is used to get rid of its argument.

```
\long\def \@gobble #1{}
\long\def \@gobbletwo #1#2{}
\long\def \@gobblethree #1#2#3{}
\long\def \@gobblefour #1#2#3#4{}
```

(*End of definition for* `\@gobble` *and others.*)

`\@firstofone` `\@firstoftwo` `\@secondoftwo` Some argument-grabbers.

```
\long\def\@firstofone#1{#1}
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}
```

`\@iden` is another name for `\@firstofone` for compatibility reasons.

```
\let\@iden\@firstofone
```

(*End of definition for* `\@firstofone` *and others.*)

`\@iden`

`\@thirdofthree` Another grabber now used in the encoding specific section.

```
\long\def\@thirdofthree#1#2#3{#3}
```

(*End of definition for* `\@thirdofthree`.)

`\@expandtwoargs` A macro to totally expand two arguments to another macro

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2022/11/01}%
⟨latexrelease⟩         {\@expandtwoargs}{protected edef}%
⟨*2ekernel | latexrelease⟩
\def\@expandtwoargs#1#2#3{%
\protected@edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{00/00/00}%
⟨latexrelease⟩         {\@expandtwoargs}{protected edef}%
⟨latexrelease⟩\def\@expandtwoargs#1#2#3{%
⟨latexrelease⟩\edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@expandtwoargs`.)

`\@backslashchar` A category code 12 backslash.

```
\edef\@backslashchar{\expandafter\@gobble\string\\}
```

(*End of definition for* `\@backslashchar`.)

1.4 Robust commands and protect

Fragile and robust commands are one of the thornier issues in LaTeX's commands. Whilst typesetting documents, LaTeX makes use of many of TeX's features, such as arithmetic, defining macros, and setting variables. However, there are (at least) three different occasions when these commands are not safe. These are called 'moving arguments' by LaTeX, and consist of:

- writing information to a file, such as indexes or tables of contents.
- writing information to the screen.
- inside an `\edef`, `\message`, `\mark`, or other command which evaluates its argument fully.

The method LaTeX uses for making fragile commands robust is to precede them with `\protect`. This can have one of four possible values:

- `\relax`, for normal typesetting. So `\protect\foo` will execute `\foo`.
- `\string`, for writing to the screen. So `\protect\foo` will write `\foo`.
- `\noexpand`, for writing to a file. So `\protect\foo` will write `\foo` followed by a space.
- `\@unexpandable@protect`, for writing a moving argument to a file. So `\protect\foo` will write `\protect\foo` followed by a space. This value is also used inside `\edef`s, `\mark`s and other commands which evaluate their arguments fully. More precisely, whenever the content of an `\edef` or `\xdef` etc. can contain arbitrary user input not under the direct control of the programmer, one should use `\proetected@edef` instead of `\edef`, etc., so that `\protect` has a suitable definition and the user input will not break if it contains fragile commands.

`\@unexpandable@protect`

```
\def\@unexpandable@protect{\noexpand\protect\noexpand}
```

(*End of definition for* `\@unexpandable@protect`.)

`\DeclareRobustCommand` `\declare@robustcommand` This is a package-writers command, which has the same syntax as `\newcommand`, but which declares a protected command. It does this by having
`\DeclareRobustCommand\foo`
define `\foo` to be `\protect\foo<space>`,
and then use `\newcommand\foo<space>`.
Since the internal command is `\foo<space>`, when it is written to an auxiliary file, it will appear as `\foo`.

We have to be a bit cleverer if we're defining a short command, such as `\_`, in order to make sure that the auxiliary file does not include a space after the command, since `\_ a` and `\_a` aren't the same. In this case we define `\_` to be:

```
\x@protect\_\protect\_<space>
```

which expands to:

```
\ifx\protect\@typeset@protect\else
   \@x@protect@\_
\fi
\protect\_<space>
```

Then if `\protect` is `\@typeset@protect` (normally `\relax`) then we just perform `\_<space>`, and otherwise `\@x@protect@` gobbles everything up and expands to `\protect\_`.

Note: setting `\protect` to any value other than `\relax` whilst in 'typesetting' mode will cause commands to go into an infinite loop! In particular, setting `\protect` to `\@empty` will cause `\_` to loop forever. It will also break lots of other things, such as protected `\ifmmode`s inside `\halign`s. If you really have to do such a thing, then please set `\@typeset@protect` to be `\@empty` as well. (This is what the code for `\patterns` does, for example.)

More fun with `\expandafter` and `\csname`.

```
\def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
```

```
\def\declare@robustcommand#1{%
   \ifx#1\@undefined\else\ifx#1\relax\else
      \@latex@info{Redefining \string#1}%
   \fi\fi
   \edef\reserved@a{\string#1}%
   \def\reserved@b{#1}%
   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
   \edef#1{%
      \ifx\reserved@a\reserved@b
         \noexpand\x@protect
         \noexpand#1%
      \fi
      \noexpand\protect
      \expandafter\noexpand\csname
         \expandafter\@gobble\string#1 \endcsname
   }%
   \let\@ifdefinable\@rc@ifdefinable
   \expandafter\new@command\csname
      \expandafter\@gobble\string#1 \endcsname
}
```

(*End of definition for* `\DeclareRobustCommand` *and* `\declare@robustcommand`.)

`\@x@protect`
`\x@protect`

```
\def\x@protect#1{%
   \ifx\protect\@typeset@protect\else
      \@x@protect#1%
   \fi
}
\def\@x@protect#1\fi#2#3{%
   \fi\protect#1%
}
```

(*End of definition for* `\@x@protect` *and* `\x@protect`.)

`\@typeset@protect` We set `\@typeset@protect` to `\relax` rather than `\@empty` to make sure that the protection mechanism stops the look-ahead and expansion performed at the start of `\halign` cells.

```
\let\@typeset@protect\relax
```

(*End of definition for* `\@typeset@protect`.)

`\set@display@protect` `\set@typeset@protect` These macros set `\protect` appropriately for typesetting or displaying.

```
\def\set@display@protect{\let\protect\string}
\def\set@typeset@protect{\let\protect\@typeset@protect}
```

(*End of definition for* `\set@display@protect` *and* `\set@typeset@protect`.)

`\protected@edef` `\protected@xdef` `\unrestored@protected@xdef` `\restore@protect` The commands `\protected@edef` and `\protected@xdef` perform 'safe' `\edef`s and `\xdef`s, saving and restoring `\protect` appropriately. For cases where restoring `\protect` doesn't matter, there's an 'unsafe' `\unrestored@protected@xdef`, useful if you know what you're doing!

```
\def\protected@edef{%
   \let\@@protect\protect
   \let\protect\@unexpandable@protect
   \afterassignment\restore@protect
   \edef
}
\def\protected@xdef{%
   \let\@@protect\protect
   \let\protect\@unexpandable@protect
   \afterassignment\restore@protect
   \xdef
}
\def\unrestored@protected@xdef{%
   \let\protect\@unexpandable@protect
   \xdef
}
\def\restore@protect{\let\protect\@@protect}
```

(*End of definition for* `\protected@edef` *and others*.)

`\protect` The normal meaning of `\protect`

```
\set@typeset@protect
```

(*End of definition for* `\protect`.)

`\MakeRobust` This macro makes an existing fragile macro robust, but only if it hasn't been robust in the past, i.e., it checks for the existence of the macro `\<name>␣` and if that exists it assumes that `\<name>` is already robust. In that case either undefine the inner macro first or use `\DeclareRobustCommand` to define it in a robust way directly. We could probably test the top-level definition to have the right kind of structure, but this is somewhat problematical as we then have to distinguish between `\long` macros and others and also take into account that sometimes the top-level is deliberately done manually (like with `\begin`).

The macro firstly checks if the control sequence in question exists at all.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\MakeRobust}{\MakeRobust}%
```

```
⟨*2ekernel | latexrelease⟩
\def\MakeRobust#1{%
  \count@=\escapechar
  \escapechar=`\\
  \@ifundefined{\expandafter\@gobble\string#1}{%
    \@latex@error{Command `\string#1' undefined.%
      \MessageBreak There is nothing here to make robust}%
    \@eha
  }%
```

Then we check if the macro is already robust. We do this by testing if the internal name for a robust macro is defined, namely `\foo␣`. If it is already defined do nothing, otherwise set `\foo␣` equal to `\foo` and redefine `\foo` so that it acts like a macro defined with `\DeclareRobustCommand`. We use `\@kernel@rename@newcommand` to copy `\foo` over to `\foo␣`, including a possible default optional argument.

```
  {%
    \@ifundefined{\expandafter\@gobble\string#1\space}%
    {%
      \expandafter\@kernel@rename@newcommand
        \csname\expandafter\@gobble\string#1\space\endcsname
        #1%
      \edef\reserved@a{\string#1}%
      \def\reserved@b{#1}%
      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
      \xdef#1{%
        \ifx\reserved@a\reserved@b
          \noexpand\x@protect\noexpand#1%
        \fi
        \noexpand\protect\expandafter\noexpand
        \csname\expandafter\@gobble\string#1\space\endcsname}%
    }%
    {\@latex@info{Command `\string#1' is already robust}}%
  }%
  \escapechar=\count@
}%
```

`\@kernel@rename@newcommand`

This macro renames a command, possibly with an optional argument (defined with `\newcommand`) from `#2` to `#1`, by renaming the internal macro `\\#2` to `\\#1` and defining `\#1` appropriately, then undefining `\#2` and `\\#2`. The `\afterassignment` trick is to make both definitions in `\@copy@newcommand` global (which are local by default).

In case the macro was defined with `\newcommand` and an optional argument, to replicate exactly the behaviour of `\DeclareRobustCommand` we have to move also the internal `\\foo` to `\\foo␣`. In that case, `#1` will be a parameterless macro (`\robust@command@chk@safe` checks that), and `\@if@newcommand` will return true (both defined below in this file). If so, we can use `\@copy@newcommand` rather than plain `\let` to copy the command over. `\@kernel@rename@newcommand` does this test and carries out the renaming.

```
\def\@kernel@rename@newcommand#1#2{%
  \robust@command@chk@safe#2%
    {\@if@newcommand#2%
      {\afterassignment\global
       \global\@copy@newcommand#1#2%
       \global\let#2\@undefined
```

```
        \global\expandafter\let\csname\string#2\endcsname\@undefined}%
      {\global\let#1=#2}}%
    {\global\let#1=#2}}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{2019/10/01}{\MakeRobust}{\MakeRobust}%
⟨latexrelease⟩\def\MakeRobust#1{%
⟨latexrelease⟩  \@ifundefined{\expandafter\@gobble\string#1}{%
⟨latexrelease⟩    \@latex@error{The control sequence `\string#1' is undefined!%
⟨latexrelease⟩      \MessageBreak There is nothing here to make robust}%
⟨latexrelease⟩    \@eha
⟨latexrelease⟩  }%
⟨latexrelease⟩  {%
⟨latexrelease⟩    \@ifundefined{\expandafter\@gobble\string#1\space}%
⟨latexrelease⟩    {%
⟨latexrelease⟩      \global\expandafter\let\csname
⟨latexrelease⟩      \expandafter\@gobble\string#1\space\endcsname=#1%
⟨latexrelease⟩      \edef\reserved@a{\string#1}%
⟨latexrelease⟩      \def\reserved@b{#1}%
⟨latexrelease⟩      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
⟨latexrelease⟩      \xdef#1{%
⟨latexrelease⟩        \ifx\reserved@a\reserved@b
⟨latexrelease⟩          \noexpand\x@protect\noexpand#1%
⟨latexrelease⟩        \fi
⟨latexrelease⟩        \noexpand\protect\expandafter\noexpand
⟨latexrelease⟩        \csname\expandafter\@gobble\string#1\space\endcsname}%
⟨latexrelease⟩    }%
⟨latexrelease⟩    {\@latex@info{The control sequence `\string#1' is already robust}}%
⟨latexrelease⟩   }%
⟨latexrelease⟩}%
⟨latexrelease⟩\let\@kernel@rename@newcommand\@undefined
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\MakeRobust}{\MakeRobust}%
⟨latexrelease⟩\def\MakeRobust#1{%
⟨latexrelease⟩  \@ifundefined{\expandafter\@gobble\string#1}{%
⟨latexrelease⟩    \@latex@error{The control sequence `\string#1' is undefined!%
⟨latexrelease⟩      \MessageBreak There is nothing here to make robust}%
⟨latexrelease⟩    \@eha
⟨latexrelease⟩  }%
⟨latexrelease⟩  {%
⟨latexrelease⟩    \@ifundefined{\expandafter\@gobble\string#1\space}%
⟨latexrelease⟩    {%
⟨latexrelease⟩      \expandafter\let\csname
⟨latexrelease⟩      \expandafter\@gobble\string#1\space\endcsname=#1%
⟨latexrelease⟩      \edef\reserved@a{\string#1}%
⟨latexrelease⟩      \def\reserved@b{#1}%
⟨latexrelease⟩      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
⟨latexrelease⟩      \edef#1{%
⟨latexrelease⟩        \ifx\reserved@a\reserved@b
⟨latexrelease⟩          \noexpand\x@protect\noexpand#1%
⟨latexrelease⟩        \fi
```

```
⟨latexrelease⟩        \noexpand\protect\expandafter\noexpand
⟨latexrelease⟩        \csname\expandafter\@gobble\string#1\space\endcsname}%
⟨latexrelease⟩    }%
⟨latexrelease⟩    {\@latex@info{The control sequence ‘\string#1’ is already robust}}%
⟨latexrelease⟩   }%
⟨latexrelease⟩}%
⟨latexrelease⟩\let\@kernel@rename@newcommand\@undefined
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\MakeRobust}{\MakeRobust}%
⟨latexrelease⟩\let\MakeRobust\@undefined
⟨latexrelease⟩\let\@kernel@rename@newcommand\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\MakeRobust` *and* `\@kernel@rename@newcommand`.)

`\kernel@make@fragile` The opposite of `\MakeRobust` except that it doesn't do many checks as it is internal to the kernel. Why does one want such a thing? Only for compatibility reasons if `latexrelease` requests a rollback of the kernel. For this reason we pretend that this command existed in all earlier versions of LaTeX i.e., we are not rolling it back since we need it precisely then. But we have to get it into the `latexrelease` file so that a roll forward is possible too.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\kernel@make@fragile}{Undo robustness}%
\def\kernel@make@fragile#1{%
  \@ifundefined{\expandafter\@gobble\string#1\space}%
```

If not robust do nothing.

```
      {}%
```

Otherwise copy `\foo␣` back to `\foo`. Then use `\@kernel@rename@newcommand` to check and copy `\\foo␣` back to `\\foo` in case the command has an optional argument. If so, also undefine `\\foo␣`, and at the end undefine `\foo␣`.

```
      {%
       \global\expandafter\let\expandafter #1\csname
         \expandafter\@gobble\string#1\space\endcsname
       \expandafter\@kernel@rename@newcommand
         \csname\expandafter\@gobble\string#1\expandafter\endcsname
         \csname\expandafter\@gobble\string#1\space\endcsname
       \global\expandafter\let\csname
         \expandafter\@gobble\string#1\space\endcsname\@undefined
      }%
}
```

```
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\kernel@make@fragile}{Undo robustness}%
⟨latexrelease⟩\def\kernel@make@fragile#1{%
⟨latexrelease⟩   \@ifundefined{\expandafter\@gobble\string#1\space}%
⟨latexrelease⟩      {}%
⟨latexrelease⟩      {%
```

```
⟨latexrelease⟩        \global\expandafter\let\expandafter #1\csname
⟨latexrelease⟩        \expandafter\@gobble\string#1\space\endcsname
⟨latexrelease⟩        \global\expandafter\let\csname
⟨latexrelease⟩        \expandafter\@gobble\string#1\space\endcsname\@undefined
⟨latexrelease⟩       }%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
```

(*End of definition for* `\kernel@make@fragile`.)

1.5 Acting on robust commands

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020-10-01}{\robust@command@act}
⟨latexrelease⟩   {Add \robust@command@act}%
⟨*2ekernel | latexrelease⟩
```

With most document level commands being robust now there is more of a requirement to have a standard way of aliasing (or copying) a command to a new name, for example to save an original definition before changing a command. `\DeclareCommandCopy` is analogous to TEX's `\let`, except that it copes with the different types of robust commands defined by LATEX's mechanisms.

A couple of "types of robustness" are defined by the LATEX 2_ε kernel, namely robust commands defined with `\DeclareRobustCommand` and commands with optional arguments defined with `\newcommand`. However there are other types of robust commands that are frequently used, which are not defined in the LATEX 2_ε kernel, like commands defined with xparse's `\NewDocumentCommand` and etoolbox's `\newrobustcmd`.

In this section we will define a generic extensible machinery to act on robust commands. This code will then be used to test if a command is robust, considered the different types of robustness, and then either copy that definition, if `\DeclareCommandCopy` (or similar) is used, or show the definition of the command, if `\ShowCommand` is used.

`\robust@command@act` The looping machinery is generic and knows nothing about what is to be done for each case. The syntax of the main macro `\robust@command@act` is:

> `\robust@command@act`⟨*action-list*⟩⟨*robust-cmd*⟩
> ⟨*fallback-action*⟩⟨*act-arg*⟩

⟨*action-list*⟩ is a token list of the form:

> {⟨*if-type-1*⟩ ⟨*act-type-1*⟩}
> {⟨*if-type-2*⟩ ⟨*act-type-2*⟩}
> ...

`\robust@command@act` will iterate over the ⟨*action-list*⟩, evaluating each ⟨*if-type-n*⟩ ⟨*robust-cmd*⟩ {⟨*true*⟩}{⟨*false*⟩}. If the ⟨*if-type-n*⟩ conditional returns ⟨*true*⟩, then ⟨*act-type-n*⟩⟨*act-arg*⟩ is executed, and the loop ends. If the conditional returns ⟨*false*⟩, then ⟨*if-type-n + 1*⟩ is executed in the same way, until either one of the conditionals return ⟨*true*⟩, or the end of the ⟨*action-list*⟩ is reached. If the end is reached, then ⟨*fallback-action*⟩⟨*act-arg*⟩ is executed before `\robust@command@act` exits.

`\robust@command@act` will start by using `\robust@command@act@chk@args` to check if the ⟨*robust-cmd*⟩ (#2) is a parameterless (possibly `\protected`) macro. If it is not, the

command is not a robust command: these always start with a parameterless user-level macro; in that case, \robust@command@act@end is used to short-circuit the process and do the ⟨*fallback-action*⟩ (#3). This first test is necessary because later on we need to be able to expand the ⟨*robust-cmd*⟩ without the risk of it Breaking Badly, and as a bonus, this speeds up the process in case we used \NewCommandCopy in a "normal" macro.

```
\long\def\robust@command@act#1#2#3#4{%
  \robust@command@chk@safe#2%
    {\expandafter\robust@command@act@loop
       \expandafter#2%
         #1{\@nnil\@nnil}%
     \robust@command@act@end}%
    {\robust@command@act@end}%
      {#3}{#4}}%
```

\robust@command@act@loop
\robust@command@act@loop@aux
\robust@command@act@do

If \robust@command@act@chk@args branched to false, then \robust@command@act@loop will loop over the list of items in the ⟨*action-list*⟩ (#1), and process each item as described earlier. If the ⟨*if-type-n*⟩ command expands to ⟨*true*⟩ then \robust@command@act@do is used to execute ⟨*act-type-n*⟩ on the ⟨*act-arg*⟩, otherwise the loop resumes with the next item.

```
\long\def\robust@command@act@loop#1#2{\robust@command@act@loop@aux#1#2}
\long\def\robust@command@act@loop@aux#1#2#3{%
  \ifx\@nnil#2%
  \else
    #2{#1}%
      {\robust@command@act@do{#3}}%
      {\expandafter\robust@command@act@loop\expandafter#1}%
  \fi}
\long\def\robust@command@act@do#1%
  \fi#2%
  \robust@command@act@end#3#4{%
  \fi
  #1#4}
```

If the end is reached and no action was taken, then do ⟨*fallback-action*⟩⟨*act-arg*⟩.

\robust@command@act@end

```
\long\def\robust@command@act@end#1#2{#1#2}
```

\robust@command@chk@safe
\robust@command@act@chk@args

```
\long\def\robust@command@chk@safe#1{%
  \begingroup
    \escapechar=`\\
  \expandafter\endgroup\expandafter
  \robust@command@act@chk@args\meaning#1:->\@nil}
\def\robust@command@act@chk@args#1:->#2\@nil{%
  \@expl@str@if@eq@@nnTF{#1}{macro}%
    {\@firstoftwo}%
    {\@expl@str@if@eq@@nnTF{#1}{\protected macro}%
      {\@firstoftwo}%
      {\@secondoftwo}}}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\robust@command@act}
⟨latexrelease⟩  {Add \robust@command@act}%
```

```
⟨latexrelease⟩\let\robust@command@act\@undefined
⟨latexrelease⟩\let\robust@command@act@loop\@undefined
⟨latexrelease⟩\let\robust@command@act@loop@aux\@undefined
⟨latexrelease⟩\let\robust@command@act@do\@undefined
⟨latexrelease⟩\let\robust@command@act@end\@undefined
⟨latexrelease⟩\let\robust@command@chk@safe\@undefined
⟨latexrelease⟩\let\robust@command@act@chk@args\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\robust@command@act` *and others.*)

1.5.1 Copying robust commands

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020-10-01}{\DeclareCommandCopy}
⟨latexrelease⟩  {Add \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
⟨*2ekernel | latexrelease⟩
```

`\NewCommandCopy`
`\RenewCommandCopy`
`\DeclareCommandCopy`

`\NewCommandCopy` starts by checking if `#1` is already defined, and raises an error if so, otherwise the definition is carried out. `\RenewCommandCopy` does (almost) the opposite. If the command is *not* defined, then an error is raised. But the definition is carried out anyhow, so the behaviour is consistent with `\renewcommand`.

A `\ProvideCommandCopy` isn't defined because it's not reasonably useful. `\provide...` commands mean "define this if there's no other definition", but copying a command (usually) implies that the command being copied is defined, so `\ProvideCommandCopy` doesn't make a lot of sense. But more importantly, the most common use case of copying a command is to redefine it later, while preserving the old definition, as in:

```
\ProvideCommandCopy \A \B
\renewcommand \B { ... \A ... }
```

then, if `\A` is already defined the first line is skipped, an in this case `\B` won't work as expected.

The three versions call the internal `\declare@commandcopy` with the proper action. `\@firstofone` will carry out the copy. The only case when the copy is not made is the ⟨*false*⟩ case for `\NewCommandCopy`, in which the command already exists and the definition is aborted.

```
\def\NewCommandCopy{%
  \declare@commandcopy
    {\@firstofone}%
    {\@firstoftwo\@notdefinable}}
\def\RenewCommandCopy{%
  \declare@commandcopy
    {\@latex@error{Command \@backslashchar\reserved@a\space undefined}\@ehc
     \@firstofone}%
    {\@firstofone}}
\def\DeclareCommandCopy{%
  \declare@commandcopy
    {\@firstofone}%
    {\@firstofone}}
```

Start by checking if the command is already defined. The proper action is taken by each specific command above. If all's good, then `\robust@command@act` is called with

`\declare@commandcopy`
`\declare@commandcopy@do`

the proper arguments as described earlier, with `\@declarecommandcopylisthook` as the ⟨*action-list*⟩ and `\declare@commandcopy@let` as the ⟨*fallback-action*⟩.

```
\long\def\declare@commandcopy#1#2#3#4{%
  \edef\reserved@a{\@expl@cs@to@str@@N#3}%
  \@ifundefined\reserved@a{#1}{#2}%
    {\declare@commandcopy@do{#3}{#4}}}
\long\def\declare@commandcopy@do#1#2{%
  \robust@command@act
    \@declarecommandcopylisthook#2%
    \declare@commandcopy@let{#1#2}}
```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

`\@declarecommandcopylisthook`

```
\def\@declarecommandcopylisthook{%
  {\@if@DeclareRobustCommand \@copy@DeclareRobustCommand}%
  {\@if@newcommand \@copy@newcommand}}
```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

`\declare@commandcopy@let`

```
\long\def\declare@commandcopy@let#1#2{\let#1=#2\relax}
```

Now the rollback code.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\DeclareCommandCopy}
⟨latexrelease⟩  {Undefine \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
⟨latexrelease⟩\let\NewCommandCopy\@undefined
⟨latexrelease⟩\let\RenewCommandCopy\@undefined
⟨latexrelease⟩\let\DeclareCommandCopy\@undefined
⟨latexrelease⟩\let\declare@commandcopy\@undefined
⟨latexrelease⟩\let\@declarecommandcopylisthook\@undefined
⟨latexrelease⟩\let\declare@commandcopy@let\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\NewCommandCopy` *and others.*)

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2023-06-01}{\DeclareEnvironmentCopy}
⟨latexrelease⟩  {Add \NewEnvironmentCopy, \RenewEnvironmentCopy, and \DeclareEnvironmentCopy}%
⟨*2ekernel | latexrelease⟩
```

`\NewEnvironmentCopy`
`\RenewEnvironmentCopy`
`\DeclareEnvironmentCopy`

If `\#1` or `\end#1` already exist one gets an error message talking about the problematical command (not the environment). The remainder of the LaTeX run is probably badly broken and it is unlikely that continuing it gives reasonable results.

```
\def\NewEnvironmentCopy{%
  \declare@environmentcopy
    {\@firstofone}%
    {\@firstoftwo\@notdefinable}}
\def\RenewEnvironmentCopy{%
  \declare@environmentcopy
    {\@latex@error{Environment \reserved@a\space undefined}\@ehc
     \@firstofone}%
    {\@firstofone}}
```

```
\def\DeclareEnvironmentCopy{%
  \declare@environmentcopy
    {\@firstofone}%
    {\@firstofone}}
\long\def\declare@environmentcopy#1#2#3#4{%
  \edef\reserved@a{\@ifundefined{#3}{end#3}{#3}}%
  \@ifundefined\reserved@a
      {\def\reserved@a{#3}#1}%
      {\def\reserved@a{#3}#2}%
    {\ExpandArgs{cc}\declare@commandcopy@do{#3}{#4}%
     \ExpandArgs{cc}\declare@commandcopy@do{end#3}{end#4}}}
```

Now the rollback code.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\DeclareEnvironmentCopy}
⟨latexrelease⟩  {Undefine \NewEnvironmentCopy, \RenewEnvironmentCopy, and \DeclareEnvironmentCop
⟨latexrelease⟩\let\NewEnvironmentCopy\@undefined
⟨latexrelease⟩\let\RenewEnvironmentCopy\@undefined
⟨latexrelease⟩\let\DeclareEnvironmentCopy\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\NewEnvironmentCopy`*,* `\RenewEnvironmentCopy`*, and* `\DeclareEnvironmentCopy`*.*)

1.5.2 Showing robust commands

`\ShowCommand` Most of the machinery defined for `\NewCommandCopy` can be used to show the definition of a robust command, in a similar fashion to `texdef`. The difference is that after the command is detected to have a given type of robustness, rather than making a copy, we use a separate routine to show its definition.

With all the machinery in place, `\ShowCommand` itself is quite simple: we use `\robust@command@act` to iterate through the `\@showcommandlisthook` list, and if nothing is found, fallback to `\show`.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020-10-01}{\ShowCommand}%
⟨latexrelease⟩  {Add \ShowCommand}%
⟨*2ekernel | latexrelease⟩
```

```
\long\def\ShowCommand#1{%
  \robust@command@act
    \@showcommandlisthook#1%
    \show#1}
```

`\@showcommandlisthook` The initial definition of `\@showcommandlisthook` contains the same tests as used for copying, but `\@show@...` commands instead of `\@copy@...`. Same as before, it is initialized to cope with `\DeclareRobustCommand` and `\newcommand` with optional arguments.

```
\def\@showcommandlisthook{%
  {\@if@DeclareRobustCommand \@show@DeclareRobustCommand}%
  {\@if@newcommand \@show@newcommand}}
```

Now the rollback code.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\ShowCommand}
⟨latexrelease⟩  {Undefine \ShowCommand}%
⟨latexrelease⟩\let\ShowCommand\@undefined
⟨latexrelease⟩\let\@showcommandlisthook\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\ShowCommand` *and* `\@showcommandlisthook`.)

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020-10-01}{\@if@DeclareRobustCommand}
⟨latexrelease⟩  {Add \@if@DeclareRobustCommand, \@if@newcommand,
⟨latexrelease⟩       \@copy@DeclareRobustCommand, \@copy@newcommand,
⟨latexrelease⟩       \@show@DeclareRobustCommand, \@show@newcommand}%
⟨*2ekernel | latexrelease⟩
```

1.5.3 Commands defined with `\DeclareRobustCommand`

`\@if@DeclareRobustCommand`

Now that we provided a generic way to copy one macro to another, we need to define a way to check if a command is one of LaTeX 2_ε's robust types. These tests are heavily based on Heiko's `\LetLtxMacro`, but chopped into separate macros.

The command `\@if@DeclareRobustCommand` checks if a command `\cmd` was defined by `\DeclareRobustCommand`. The test returns true if the expansion of `\cmd` is exactly `\protect\cmd␣`.

```
\long\def\@if@DeclareRobustCommand#1{%
  \begingroup
    \escapechar=`\\
    \edef\reserved@a{\string#1}%
    \edef\reserved@b{\detokenize{#1}}%
    \xdef\@gtempa{%
      \ifx\reserved@a\reserved@b
         \noexpand\x@protect
         \noexpand#1%
      \fi
      \noexpand\protect
      \expandafter\noexpand\csname\@expl@cs@to@str@@N#1 \endcsname}%
  \endgroup
  \ifx\@gtempa#1\relax
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi}
```

`\@copy@DeclareRobustCommand`
`\copy@kernel@robust@command`

If a command was defined by `\DeclareRobustCommand` (that is, `\@if@DeclareRobustCommand` returns true), then to make a copy of `\cmd` into `\foo` we define the latter such that it expands to `\protect\foo␣`, then make `\foo␣` equal to `\cmd␣`.

There is one detail we need to take care of: if a command was defined with `\DeclareRobustCommand` it may still have an optional argument, in which case there is one more macro layer before the actual definition of the command. We use `\@if@newcommand` to check that and `\@copy@newcommand` to do the copying.

```
\long\def\@copy@DeclareRobustCommand#1#2{%
  \begingroup
    \escapechar=`\\
```

```
    \edef\reserved@a{\string#1}%
    \edef\reserved@b{\detokenize{#1}}%
    \edef\reserved@a{%
  \endgroup
  \def\noexpand#1{%
    \ifx\reserved@a\reserved@b
       \noexpand\x@protect
       \noexpand#1%
    \fi
    \noexpand\protect
    \expandafter\noexpand\csname\@expl@cs@to@str@@N#1 \endcsname}%
  \noexpand\copy@kernel@robust@command
    \expandafter\noexpand\csname\@expl@cs@to@str@@N#1 \endcsname
    \expandafter\noexpand\csname\@expl@cs@to@str@@N#2 \endcsname}%
  \reserved@a}
\long\def\copy@kernel@robust@command#1#2{%
  \robust@command@chk@safe#2%
    {\@if@newcommand#2%
       {\@copy@newcommand}%
       {\declare@commandcopy@let}}
    {\declare@commandcopy@let}%
  #1#2}
```

\@show@DeclareRobustCommand
\show@kernel@robust@command
\@show@macro

Showing the command is pretty simple. This command prints the top-level expansion as TEX's `\show` would, but with `robust macro:` rather than just `macro:`, then a blank line and then `\show` the inner command. For a macro defined with, say, `\DeclareRobustCommand\foo[1]{bar}`, it will print:

```
> \foo=robust macro:
->\protect \foo  .

> \foo =\long macro:
#1->bar.
```

If the inner command is defined with an optional argument, then `\@show@newcommand` is also used.

The value of `\escapechar` is deliberately not enforced, so `\ShowCommand` behaves more like `\show`.

```
\long\def\@show@DeclareRobustCommand#1{%
  \typeout{> \string#1=robust macro:}%
  \typeout{->\@expl@cs@replacement@spec@@N#1.^^J}%
  \expandafter\show@kernel@robust@command
    \csname\@expl@cs@to@str@@N#1 \endcsname}
\long\def\show@kernel@robust@command#1{%
  \robust@command@chk@safe#1%
    {\@if@newcommand#1%
       {\@show@newcommand}%
       {\@show@macro}}%
    {\@show@macro}%
  #1}
\let\@show@macro\show
```

(*End of definition for* `\@if@DeclareRobustCommand` *and others.*)

1.5.4 Commands defined with \newcommand (with optional argument)

\@if@newcommand

A command \cmd (or \cmd␣, if it was defined with \DeclareRobustCommand) with an optional argument will expand to \@protected@testopt\cmd\\cmd{<opt>}. To check that we look at the first three tokens in the expansion of \cmd, and return true or false accordingly.

This test *requires* that the command be a parameterless macro, otherwise it will not work (and probably break). This is ensured with \robust@command@chk@safe before calling \@if@newcommand.

```
\long\def\@if@newcommand#1{%
  \edef\reserved@a{%
    \noexpand\@protected@testopt
    \noexpand#1%
    \expandafter\noexpand\csname\@backslashchar\@expl@cs@to@str@@N#1\endcsname}%
  \edef\reserved@b{%
    \unexpanded\expandafter\expandafter\expandafter
      {\expandafter\@carcube#1{}{}{}\@nil}}%
  \ifx\reserved@a\reserved@b
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi}
```

\@copy@newcommand

Then, if a command \cmd takes an optional argument, we copy it to \foo by defining the latter to expand to \@protected@testopt\foo\\foo{<opt>}.

```
\long\def\@copy@newcommand#1#2{%
  \edef#1{\noexpand\@protected@testopt
    \noexpand#1%
    \expandafter\noexpand\csname\@backslashchar\@expl@cs@to@str@@N#1\endcsname
    \unexpanded\expandafter\expandafter\expandafter
      {\expandafter\@gobblethree#2}}%
  \expandafter
  \let\csname\@backslashchar\@expl@cs@to@str@@N#1\expandafter\endcsname
      \csname\@backslashchar\@expl@cs@to@str@@N#2\endcsname}
```

\@show@newcommand
\@show@newcommand@aux

A command being \shown here is guaranteed to have an optional argument. Start by showing the top-level expansion of the command (using \typeout to avoid TeX asking for interaction and extra context lines), then call \@show@newcommand@aux with the internal command, which contains the actual definition, and with the expansion of the command to extract the default value of the optional argument.

```
\long\def\@show@newcommand#1{%
  \typeout{> \string#1=robust macro:}%
  \typeout{->\@expl@cs@replacement@spec@@N#1.^^J}%
  \expandafter\@show@newcommand@aux
    \csname\@backslashchar\@expl@cs@to@str@@N#1\expandafter\endcsname
    \expandafter{#1}\@show@tokens}
```

For a macro defined with, say, \newcommand\foo[1][opt]{bar}, it will print:

```
> \foo=robust macro:
->\@protected@testopt \foo \\foo {opt}.

> \\foo=\long macro:
```

```
> default #1=opt.
[#1]->bar.
```

If the command was defined with `\DeclareRobustCommand`, then another pair of lines show the top-level expansion `\protect␣\foo␣␣`.

```
\long\def\@show@newcommand@aux#1#2#3{%
  \typeout{> \string#1=\@expl@cs@prefix@spec@@N#1macro:}%
  #3{default \string##1=\expandafter\detokenize\@gobblethree#2.^^J%
    \@expl@cs@parameter@spec@@N#1->\@expl@cs@replacement@spec@@N#1}}
```

This macro prints the contents of the token list (macro) `#1` using `\showtokens`. The `\expandafter` gymnastics ensures that `\showtokens` itself, and the internals of this macro aren't showed in the context lines.

`\@show@tokens`

```
\long\def\@show@tokens#1{%
  \edef\reserved@a{#1}%
  \showtokens\expandafter
    \expandafter\expandafter{\expandafter\reserved@a}}
```

Now the rollback code.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\@if@DeclareRobustCommand}
⟨latexrelease⟩  {Undefine \@if@DeclareRobustCommand, \@if@newcommand,
⟨latexrelease⟩            \@copy@DeclareRobustCommand, \@copy@newcommand,
⟨latexrelease⟩            \@show@DeclareRobustCommand, \@show@newcommand}%
⟨latexrelease⟩\let\@if@DeclareRobustCommand\@undefined
⟨latexrelease⟩\let\@copy@DeclareRobustCommand\@undefined
⟨latexrelease⟩\let\@show@DeclareRobustCommand\@undefined
⟨latexrelease⟩\let\@if@newcommand\@undefined
⟨latexrelease⟩\let\@copy@newcommand\@undefined
⟨latexrelease⟩\let\@show@newcommand\@undefined
%
⟨latexrelease⟩\let\copy@kernel@robust@command\@undefined
⟨latexrelease⟩\let\show@kernel@robust@command\@undefined
⟨latexrelease⟩\let\@show@newcommand@aux\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@if@newcommand` *and others.*)

1.5.5 Showing environments

`\ShowEnvironment`

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2023-06-01}{\ShowEnvironment}
⟨latexrelease⟩  {Add \ShowEnvironment}%
⟨*2ekernel | latexrelease⟩
```

`\ShowEnvironment` is quite similar to `\ShowCommand`. We will pass the environment ⟨*env*⟩ around as the macro `\env`, because `\robust@command@act` expects a single token.

```
\def\ShowEnvironment#1{%
  \expandafter\@show@environment\csname #1\endcsname}
\long\def\@show@environment#1{%
  \robust@command@act
```

```
    \@showenvironmentlisthook#1%
    \@show@normalenv#1}
```

\@showenvironmentlisthook

This is similar to \@showcommandlisthook, but uses the dedicated versions for environments.

```
\def\@showenvironmentlisthook{%
  {\@if@DeclareRobustCommand \@show@DeclareRobustCommand@env}%
  {\@if@newcommand \@show@newcommand@env}}
```

\@show@newcommand@env
\@show@DeclareRobustCommand@env

These are similar to the command versions below, except they say "environment" and call \@show@environment@end to print the \end part.

```
\long\def\@show@newcommand@env#1{%
  \@show@environment@begin#1%
  \expandafter\@show@newcommand@aux
    \csname\@backslashchar\@expl@cs@to@str@@N#1\expandafter\endcsname
    \expandafter{#1}\@show@typeout
  \@show@environment@end#1}
\long\def\@show@DeclareRobustCommand@env#1{%
  \@show@environment@begin#1%
  \begingroup
    \let\@show@tokens\@show@typeout
    \let\@show@macro\@show@nonstop
    \expandafter\show@kernel@robust@command
      \csname\@expl@cs@to@str@@N#1 \endcsname
  \endgroup
  \@show@environment@end#1}
\long\def\@show@environment@begin#1{%
  \typeout{> \string\begin{\@expl@cs@to@str@@N#1}=environment:}%
  \typeout{\@expl@cs@parameter@spec@@N#1->%
           \@expl@cs@replacement@spec@@N#1.^^J}}
```

\@show@normalenv
\@show@environment@end

A "normal" environment is straightforward. \@show@environment@end needs to check if the \end part is defined and show it accordingly, otherwise the output would show gibberish.

```
\long\def\@show@normalenv#1{%
  \@show@environment@begin#1%
  \@show@environment@end#1}
\long\def\@show@environment@end#1{%
  \expandafter\@show@environment@end@aux
    \csname end\@expl@cs@to@str@@N#1\endcsname#1}
\long\def\@show@environment@end@aux#1#2{%
  \@show@tokens{\string\end{\@expl@cs@to@str@@N#2}%
    \ifx\relax#1=undefined%
    \else:^^J\@expl@cs@parameter@spec@@N#1->%
             \@expl@cs@replacement@spec@@N#1%
    \fi}}
```

And here some auxiliaries:

\@show@nonstop
\@show@typeout

\@show@nonstop same output as \show, but doesn't stop for interaction;

\@show@typeout same output as \showtokens, but doesn't stop for interaction.

```
\def\@show@nonstop#1{%
  \typeout{> \string#1=\@expl@cs@prefix@spec@@N#1macro:^^J%
   \@expl@cs@parameter@spec@@N#1->\@expl@cs@replacement@spec@@N#1.}}
\def\@show@typeout#1{\typeout{> #1.^^J}}
```

Now the rollback code.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\ShowEnvironment}
⟨latexrelease⟩  {Undefine \ShowEnvironment}%
⟨latexrelease⟩\let\ShowEnvironment\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\ShowEnvironment` *and others.*)

1.6 Internal defining commands

These commands are used internally to define other LaTeX commands.

`\@ifundefined` Check if first arg is undefined or `\relax` and execute second or third arg depending,

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2018-04-01}{\@ifundefined}
⟨latexrelease⟩{Leave commands undefined in \@ifundefined}%
⟨*2ekernel | latexrelease⟩
```

Version using `\ifcsname` to avoid defining undefined tokens to `\relax`. Defined here to simplify using unmatched `\fi`.

```
\def\@ifundefined#1{%
  \ifcsname#1\endcsname\@ifundefin@d@i\else\@ifundefin@d@ii\fi{#1}}
```

```
\long\def\@ifundefin@d@i#1\fi#2{\fi
  \expandafter\ifx\csname #2\endcsname\relax
    \@ifundefin@d@ii
  \fi
  \@secondoftwo}
```

```
\long\def\@ifundefin@d@ii\fi#1#2#3{\fi #2}
```

Now test of engine.

```
\ifx\numexpr\@undefined
```

Classic version (should not be needed as etex is assumed).

```
\def\@ifundefined#1{%
  \expandafter\ifx\csname#1\endcsname\relax
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi}
\else\ifx\directlua\@undefined
```

Use the `\ifcsname` defined above.

```
\else
```

Optimised version for LuaTEX, using \lastnamedcs

```
\def\@ifundefined#1{%
  \ifcsname#1\endcsname
    \expandafter\ifx\lastnamedcs\relax\else\@ifundefin@d@i\fi
  \fi
  \@firstoftwo}
```

```
\long\def\@ifundefin@d@i#1#2#3#4#5{#1#2#5}
```

```
\fi
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\@ifundefined}
⟨latexrelease⟩{Leave commands undefined in \@ifundefined}%
⟨latexrelease⟩\def\@ifundefined#1{%
⟨latexrelease⟩  \expandafter\ifx\csname#1\endcsname\relax
⟨latexrelease⟩    \expandafter\@firstoftwo
⟨latexrelease⟩  \else
⟨latexrelease⟩    \expandafter\@secondoftwo
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \@ifundefined.)

\@qend
\@qrelax

The following define \@qend and \@qrelax to be the strings 'end' and 'relax' with the characters \catcoded 12.

```
\edef\@qend{\expandafter\@cdr\string\end\@nil}
\edef\@qrelax{\expandafter\@cdr\string\relax\@nil}
```

(*End of definition for* \@qend *and* \@qrelax.)

\@ifnextchar

\@ifnextchar peeks at the following character and compares it with its first argument. If both are the same it executes its second argument, otherwise its third.

```
\long\def\@ifnextchar#1#2#3{%
  \let\reserved@d=#1%
  \def\reserved@a{#2}%
  \def\reserved@b{#3}%
  \futurelet\@let@token\@ifnch}
```

(*End of definition for* \@ifnextchar.)

\kernel@ifnextchar

This macro is the kernel version of \@ifnextchar which is used in a couple of places to prevent the AMS variant from being used since in some places this produced chaos. For example, if an fd file is loaded in a random place then the optional argument to \ProvidesFile could get printed there instead of being written only in the log file. This happened when there was a space or a newline between the mandatory and optional arguments! It should really be fixed in the amsmath package one day, but...

Note that there may be other places in the kernel where this version should be used rather than the original, but variable, version.

```
\let\kernel@ifnextchar\@ifnextchar
```

(*End of definition for* \kernel@ifnextchar.)

\@ifnch

\@ifnch is a tricky macro to skip any space tokens that may appear before the character in question. If it encounters a space token, it calls xifnch.

```
\def\@ifnch{%
  \ifx\@let@token\@sptoken
    \let\reserved@c\@xifnch
  \else
    \ifx\@let@token\reserved@d
      \let\reserved@c\reserved@a
    \else
      \let\reserved@c\reserved@b
    \fi
  \fi
  \reserved@c}
```

(*End of definition for* \@ifnch.)

\@sptoken

The following code makes \@sptoken a space token. It is important here that the control sequence \: consists of a non-letter only, so that the following whitespace is significant. Together with the fact that the equal sign in a \let may be followed by only one optional space the desired effect is achieved. NOTE: the following hacking must precede the definition of \: as math medium space.

```
\def\:{\let\@sptoken= } \:  % this makes \@sptoken a space token
```

(*End of definition for* \@sptoken.)

\@xifnch

In the following definition of \@xifnch, \: is again used to get a space token as delimiter into the definition.

```
\def\:{\@xifnch} \expandafter\def\: {\futurelet\@let@token\@ifnch}
```

(*End of definition for* \@xifnch.)

\@ifstar

The new implementation below avoids passing the ⟨*true code*⟩ Through one more \def than the ⟨*false code*⟩, which previously meant that # had to be written as #### in one argument, but ## in the other. The * is gobbled by \@firstoftwo.

```
\def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}}
```

(*End of definition for* \@ifstar.)

\@dblarg
\@xdblarg

```
\long\def\@dblarg#1{\kernel@ifnextchar[{#1}{\@xdblarg{#1}}}
\long\def\@xdblarg#1#2{#1[{#2}]{#2}}
```

(*End of definition for* \@dblarg *and* \@xdblarg.)

\@sanitize

The command \@sanitize changes the catcode of all special characters except for braces to 'other'. It can be used for commands like \index that want to write their arguments verbatim. Needless to say, this command should only be executed within a group, or chaos will ensue.

```
\def\@sanitize{\@makeother\ \@makeother\\\@makeother\$\@makeother\&%
\@makeother\#\@makeother\^\@makeother\_\@makeother\%\@makeother\~}
```

(*End of definition for* \@sanitize.)

\@onelevel@sanitize

This makes the whole "meaning" of #1 (its one-level expansion) into catcode 12 tokens: it could be used in \DeclareRobustCommand.

If it is to be used on default float specifiers, this should be done when they are defined.

```
\def \@onelevel@sanitize #1{%
  \edef #1{\expandafter\strip@prefix
           \meaning #1}%
}
```

(*End of definition for* \@onelevel@sanitize.)

\string@makeletter
\@string@makeletter
\char@if@alph

Iterates through a string, turning each alphabetic character into a catcode-11 token (partly undoes a \detokenize). Useful for \ifx-based string comparisons where \detokenize-ing the other string would break too much code.

The macro uses expl3's \@expl@str@map@function@@NN to iterate on the string (without losing spaces) and applies \@string@makeletter on each character. The latter checks if character is between a–z or A–Z, and uses \@alph or \@Alph to get the corresponding catcode-11 token. Other tokens are passed through unchanged.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\string@makeletter}
⟨latexrelease⟩  {Add \string@makeletter}%
⟨*2ekernel | latexrelease⟩
\def\string@makeletter#1{%
  \@expl@str@map@function@@NN#1\@string@makeletter}
\def\@string@makeletter#1{%
  \char@if@alph{#1}%
    {\@expl@char@generate@@nn{`#1}{11}}%
    {#1}}
\def\char@if@alph#1{%
  \ifnum0\ifnum`#1<`A 1\fi\ifnum`#1>`z 1\fi
      \if\ifnum`#1>`Z @\fi\ifnum`#1<`a @\fi01\fi>0
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\string@makeletter}
⟨latexrelease⟩  {Undefine \string@makeletter}%
⟨latexrelease⟩\let\string@makeletter\@undefined
⟨latexrelease⟩\let\@string@makeletter\@undefined
⟨latexrelease⟩\let\char@if@alph\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \string@makeletter, \@string@makeletter, *and* \char@if@alph.)

\makeatletter
\makeatother

Make internal control sequences accessible or inaccessible.

```
\DeclareRobustCommand\makeatletter{\catcode`\@11\relax}
\DeclareRobustCommand\makeatother{\catcode`\@12\relax}
```

(*End of definition for* \makeatletter *and* \makeatother.)

2 Discretionary Hyphenation

\-
\@dischyph

Moved here to be after the definition of `\DeclareRobustCommand`.

The primitive `\-` command adds a discretionary hyphen using the current font's `\hyphenchar`. Monospace fonts are usually declared with `\hyphenchar` set to -1 to suppress hyphenation.

LaTeX, from LaTeX2.09 in 1986 defined `\-` by

```
\def\-{\discretionary{-}{}{}}
```

The following comment was added when these commands were first set up, 19 April 1986:

> the `\-` command is redefined to allow it to work in the `\ttfamily` type style, where automatic hyphenation is suppressed by setting `\hyphenchar` to -1. The original primitive TeX definition is saved as `\@@hyph` just in case anyone needs it.

LaTeX 2_ε, between 1993 and 2017, had a comment at this point saying that the definition "would probably change" because the definition always uses `-`. The definition used below was given in comments at this point during time.

In 2017 we finally enabled this definition by default, with the older LaTeX definition accessible via latexrelease as usual.

In LuaLaTeX the primitive definition of `\-` is used directly because it's use of extended hyphenation parameters means that `\-` works correctly even with `\hyphenchar` set to -1. This change makes `\-` under LuaLaTeX compatible with language specific hyphenation characters.

Temporary definition of `\@latex@info`, final definition is later.

```
\def\@latex@info#1{}
```

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\-}{Use primitive \- in Lua\LaTeX}%
⟨*2ekernel | latexrelease⟩
\ifx\directlua\@undefined
  \DeclareRobustCommand{\-}{%
    \discretionary{%
      \char \ifnum\hyphenchar\font<\z@
              \defaulthyphenchar
            \else
              \hyphenchar\font
            \fi
                 }{}{}%
  }
\else
  \let\-\@@hyph
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2017/04/15}{\-}{Use \hyphenchar in \-}%
⟨latexrelease⟩\DeclareRobustCommand{\-}{%
⟨latexrelease⟩  \discretionary{%
⟨latexrelease⟩    \char \ifnum\hyphenchar\font<\z@
⟨latexrelease⟩            \defaulthyphenchar
⟨latexrelease⟩          \else
```

```
⟨latexrelease⟩              \hyphenchar\font
⟨latexrelease⟩           \fi
⟨latexrelease⟩                }{}{}%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\-}{Use \hyphenchar in \-}%
⟨latexrelease⟩\def\-{\discretionary{-}{}{}}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨*2ekernel | latexrelease⟩
\let\@dischyph=\-
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
```

(*End of definition for* `\-` *and* `\@dischyph`.)

Delayed from `ltvers.dtx`

```
\newif\if@includeinrelease
\@includeinreleasefalse
```

Delayed from `ltplain.dtx`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\allowbreak}{Make various commands robust}%
\MakeRobust\allowbreak
\MakeRobust\bigbreak
\MakeRobust\break
\MakeRobust\dotfill
\MakeRobust\frenchspacing
\MakeRobust\goodbreak
\MakeRobust\hrulefill
\MakeRobust\medbreak
\MakeRobust\nobreak
\MakeRobust\nonfrenchspacing
\MakeRobust\obeylines
\MakeRobust\obeyspaces
\MakeRobust\slash
\MakeRobust\smallbreak
\MakeRobust\strut
\MakeRobust\underbar
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\allowbreak}{Make various commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\allowbreak
⟨latexrelease⟩\kernel@make@fragile\bigbreak
⟨latexrelease⟩\kernel@make@fragile\break
⟨latexrelease⟩\kernel@make@fragile\dotfill
⟨latexrelease⟩\kernel@make@fragile\frenchspacing
⟨latexrelease⟩\kernel@make@fragile\goodbreak
⟨latexrelease⟩\kernel@make@fragile\hrulefill
⟨latexrelease⟩\kernel@make@fragile\medbreak
⟨latexrelease⟩\kernel@make@fragile\nobreak
⟨latexrelease⟩\kernel@make@fragile\nonfrenchspacing
```

```
⟨latexrelease⟩\kernel@make@fragile\obeylines
⟨latexrelease⟩\kernel@make@fragile\obeyspaces
⟨latexrelease⟩\kernel@make@fragile\slash
⟨latexrelease⟩\kernel@make@fragile\smallbreak
⟨latexrelease⟩\kernel@make@fragile\strut
⟨latexrelease⟩\kernel@make@fragile\underbar
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\g@addto@macro` Globally add to the end of a macro. This macro is used by the kernel to add to its internal hooks.

```
\long\def\g@addto@macro#1#2{%
  \begingroup
    \toks@\expandafter{#1#2}%
    \xdef#1{\the\toks@}%
  \endgroup}
```

(*End of definition for* `\g@addto@macro`.)

```
⟨/2ekernel⟩
```

File 07
ltcmd.dtx

1 Creating document commands

Document commands should be created using the tools provided by this module: `\NewDocumentCommand`, etc., in almost all cases. This allows clean separation of document-level syntax from code-level interfaces. Users have a need to create new document commands, and as such a significant amount of documentation for ltcmd is provided as part of `usrguide3`. Here, additional material aimed at programmers is provided

```
⟨@@=cmd⟩
```

```
⟨*2ekernel⟩
\message{document commands,}
⟨/2ekernel⟩
```

ltcmd code contains an `^^@` character, which usually has catcode 15, so `\IncludeInRelease` will break when this code is being skipped, so we'll save the catcode of `^^@` to restore later:

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\edef\@latexrelease@catcode@null{\the\catcode`\^^@ }
⟨latexrelease⟩\catcode`\^^@=12
\ExplSyntaxOn
⟨latexrelease⟩\NewModuleRelease{2020/10/01}{ltcmd}
⟨latexrelease⟩                    {Document~command~parser}%
```

1.1 Variables and constants

`\l__cmd_arg_spec_tl` Holds the argument specification after normalization of shorthands.

```
\tl_new:N \l__cmd_arg_spec_tl
```

`\l__cmd_args_tl` Token list variable for grabbed arguments.

```
\tl_new:N \l__cmd_args_tl
```

`\l__cmd_args_i_tl` `\l__cmd_args_ii_tl` Hold the modified arguments when dealing with default values or processors.

```
\tl_new:N \l__cmd_args_i_tl
\tl_new:N \l__cmd_args_ii_tl
```

`\l__cmd_current_arg_int` The number of the current argument being set up: this is used to make sure there are at most 9 arguments, then for creating the expandable auxiliary functions and knowing how many arguments the code function should take.

```
\int_new:N \l__cmd_current_arg_int
```

`\l__cmd_defaults_bool`
`\l__cmd_defaults_tl`

The boolean indicates whether there are any argument with default value other than `-NoValue-`; the token list holds the code to determine these default values in terms of other arguments.

```
\bool_new:N \l__cmd_defaults_bool
\tl_new:N \l__cmd_defaults_tl
```

`\l__cmd_environment_bool`

Generating environments uses the same mechanism as generating functions. However, full processing of arguments is always needed for environments, and so the function-generating code needs to know this. This variable is also used at run time to give correct error messages.

```
\bool_new:N \l__cmd_environment_bool
```

`\l__cmd_environment_str`

Name of the environment, used at definition time and at run time.

```
\str_new:N \l__cmd_environment_str
```

`\l__cmd_expandable_bool`

Used to indicate if an expandable command is begin generated, as this affects both the acceptable argument types and how they are implemented.

```
\bool_new:N \l__cmd_expandable_bool
```

`\l__cmd_expandable_aux_name_tl`

Used to create pretty-printing names for the auxiliaries: although the immediate definition does not vary, the full expansion does and so it does not count as a constant.

```
\tl_new:N \l__cmd_expandable_aux_name_tl
\tl_set:Nn \l__cmd_expandable_aux_name_tl
  {
    \l__cmd_function_tl \c_space_tl
    ( arg~ \int_use:N \l__cmd_current_arg_int )
  }
```

`\g__cmd_grabber_int`

Used (in exceptional cases) to get unique names for grabbers used by expandable commands.

```
\int_new:N \g__cmd_grabber_int
```

`\l__cmd_fn_tl`

For passing the pre-formed name of the auxiliary to be used as the parsing function.

```
\tl_new:N \l__cmd_fn_tl
```

`\l__cmd_fn_code_tl`

For passing the pre-formed name of the auxiliary that contains the actual code.

```
\tl_new:N \l__cmd_fn_code_tl
```

`\l__cmd_function_tl` Holds the control sequence name of the function currently being defined: used to avoid passing this as an argument and to avoid repeated use of `\cs_to_str:N`.

```
\tl_new:N \l__cmd_function_tl
```

`\l__cmd_grab_expandably_bool`

When defining a non-expandable command, indicates whether the arguments can all safely be grabbed by expandable grabbers. This is to support abuses of xparse that use protected functions inside csname constructions.

```
\bool_new:N \l__cmd_grab_expandably_bool
```

`\l__cmd_obey_spaces_bool` For trailing optionals.

```
\bool_new:N \l__cmd_obey_spaces_bool
```

`\l__cmd_last_delimiters_tl` Holds the delimiters (first tokens) of all optional arguments since the previous mandatory argument, to warn about cases where it would be impossible to omit optional arguments completely because the following mandatory argument has the same delimiter as one of the optional arguments.

```
\tl_new:N \l__cmd_last_delimiters_tl
```

`\l__cmd_long_bool` Used to indicate that an argument is long, on a per-argument basis.

```
\bool_new:N \l__cmd_long_bool
```

`\l__cmd_suppress_strip_bool`

Used to indicate that an a pair of braces should not be stripped from an optional argument.

```
\bool_new:N \l__cmd_suppress_strip_bool
```

`\l__cmd_m_args_int` The number of `m` arguments: if this is the same as the total number of arguments, then a short-cut can be taken in the creation of the grabber code.

```
\int_new:N \l__cmd_m_args_int
```

`\l__cmd_prefixed_bool` When preparing the signature of non-expandable commands, indicates that the current argument is affected by a processor or by `+` (namely is long).

```
\bool_new:N \l__cmd_prefixed_bool
```

`\l__cmd_process_all_tl`
`\l__cmd_process_one_tl`
`\l__cmd_process_some_bool`

When preparing the signature, the processors that will be applied to a given argument are collected in `\l__cmd_process_one_tl`, while `\l__cmd_process_all_tl` contains processors for all arguments. The boolean indicates whether there are any processors (to bypass the whole endeavour otherwise).

```
\tl_new:N \l__cmd_process_all_tl
\tl_new:N \l__cmd_process_one_tl
\bool_new:N \l__cmd_process_some_bool
```

`\l__cmd_saved_args_tl`

Stores `\l__cmd_args_tl` to deal with space-trimming of `b`-type arguments.

```
\tl_new:N \l__cmd_saved_args_tl
```

`\l__cmd_signature_tl`

Used when constructing the signature (code for argument grabbing) to hold what will become the implementation of the main function. When arguments are grabbed (at point of use of the command/environment), it also stores the code for grabbing the remaining arguments.

```
\tl_new:N \l__cmd_signature_tl
```

`\l__cmd_some_obey_spaces_bool`
`\l__cmd_some_long_bool`
`\l__cmd_some_short_bool`

These flags are set while normalizing the argument specification. The `obey_spaces` one is used to detect when `!` is used on an argument that is not a trailing optional argument. The other two are used to check whether all short arguments appear before long arguments: this is needed to grab arguments expandably. As soon as the first long argument is seen (other than `t`-type, whose long status is ignored) the `some_long` flag is set. The `some_short` flag is used for expandable commands, to know whether to define a short auxiliary too.

```
\bool_new:N \l__cmd_some_obey_spaces_bool
\bool_new:N \l__cmd_some_long_bool
\bool_new:N \l__cmd_some_short_bool
```

`\q__cmd_recursion_tail`
`\q__cmd_recursion_stop`
`\__cmd_if_recursion_tail_stop_do:Nn`
`\__cmd_use_i_delimit_by_q_recursion_stop:nw`

Quarks and functions for internal processing.

```
\quark_new:N \q__cmd_recursion_tail
\quark_new:N \q__cmd_recursion_stop
\__kernel_quark_new_test:N \__cmd_if_recursion_tail_stop_do:Nn
```

(*End of definition for* `\__cmd_if_recursion_tail_stop_do:Nn` *and* `\__cmd_use_i_delimit_by_q_recursion_stop:nw`.)

\l__cmd_tmp_prop
\l__cmd_tmpa_tl
\l__cmd_tmpb_tl

Scratch space.

```
\prop_new:N \l__cmd_tmp_prop
\tl_new:N \l__cmd_tmpa_tl
\tl_new:N \l__cmd_tmpb_tl
\cs_new_eq:NN \__cmd_tmp:w ?
```

(*End of definition for* `\__cmd_tmp:w`.)

With xparse, information about commands being (re)defined was switched off by default, unless the `log-declarations` package option was used, so here we'll switch that off as well.

```
\msg_redirect_module:nnn { cmd } { info } { none }
```

Also add cmd to the LaTeX messages.

```
\prop_gput:Nnn \g_msg_module_type_prop { cmd } { LaTeX }
```

1.2 Declaring commands and environments

__cmd_declare_cmd:Nnn
__cmd_declare_expandable_cmd:Nnn
__cmd_declare_cmd_aux:Nnn
__cmd_declare_cmd_internal:Nnnn

The main functions for creating commands set the appropriate flag then use the same internal code to do the definition.

```
\cs_new_protected:Npn \__cmd_declare_cmd:Nnn
  {
    \bool_set_false:N \l__cmd_expandable_bool
    \__cmd_declare_cmd_aux:Nnn
  }
\cs_new_protected:Npn \__cmd_declare_expandable_cmd:Nnn
  {
    \bool_set_true:N \l__cmd_expandable_bool
    \__cmd_declare_cmd_aux:Nnn
  }
```

The first stage is to log information, both for the user in the log and for programmatic use in a property list of all declared commands.

```
\cs_new_protected:Npn \__cmd_declare_cmd_aux:Nnn #1#2#3
  {
    \cs_if_exist:NTF #1
      {
        \msg_info:nnxx { cmd } { redefine }
          { \token_to_str:N #1 } { \tl_to_str:n {#2} }
      }
      {
        \bool_lazy_or:nnT
          { \cs_if_exist_p:c { \cs_to_str:N #1 ~ code } }
          { \cs_if_exist_p:c { \cs_to_str:N #1 ~ defaults } }
          {
            \msg_warning:nnx { cmd } { unsupported-let }
              { \token_to_str:N #1 }
          }
        \msg_info:nnxx { cmd } { define-command }
          { \token_to_str:N #1 } { \tl_to_str:n {#2} }
      }
    \bool_set_false:N \l__cmd_environment_bool
    \__cmd_declare_cmd_internal:Nnnn #1 {#2} {#3} { }
```

```
}
```

At definition time, the variable `\l__cmd_fn_tl` is only used for error messages. The real business of defining a document command starts with setting up the appropriate name, then normalizing the argument specification to get rid of shorthands.

```
\cs_new_protected:Npn \__cmd_declare_cmd_internal:Nnnn #1#2#3#4
  {
    \tl_set:Nx \l__cmd_function_tl { \cs_to_str:N #1 }
    \__cmd_normalize_arg_spec:n {#2}
    \exp_args:No \__cmd_prepare_signature:n \l__cmd_arg_spec_tl
    \__cmd_declare_cmd_code:Nnn #1 {#2} {#3}
    #4
    \__cmd_break_point:n {#2}
  }
```

(*End of definition for* `\__cmd_declare_cmd:Nnn` *and others.*)

`\__cmd_break_point:n` A marker used to escape from creating a definition if necessary.

```
\cs_new_eq:NN \__cmd_break_point:n \use_none:n
```

(*End of definition for* `\__cmd_break_point:n`.)

`\__cmd_all_m_check:n` `\__cmd_all_m_check_aux:n` A quick loop to check for all `(+)m`-type arguments.

```
\cs_new:Npn \__cmd_all_m_check:n #1
  { \tl_map_function:nN {#1} \__cmd_all_m_check_aux:n }
\cs_new:Npn \__cmd_all_m_check_aux:n #1
  {
    \str_if_eq:nnF {#1} { m }
      {
        \str_if_eq:nnF {#1} { + }
          { X }
      }
  }
```

(*End of definition for* `\__cmd_all_m_check:n` *and* `\__cmd_all_m_check_aux:n`.)

`\__cmd_declare_cmd_code:Nnn` `\__cmd_declare_cmd_optimized:Nnn` `\__cmd_declare_cmd_code_aux:Nnn` `\__cmd_declare_cmd_code_expandable:Nnn` `\__cmd_start_optimized:` At this stage we can check for a short-cut possibility: if the argument specification is made up of just `(+)m` tokens, and if all arguments are either short or long, then we can produce an optimized document command. This only applies to document commands, not creation of environments (which are more complex).

```
\cs_new_protected:Npn \__cmd_declare_cmd_code:Nnn #1#2
  {
    \bool_lazy_any:nTF
      {
        { \l__cmd_environment_bool }
        {
          \bool_lazy_and_p:nn
            { \l__cmd_some_short_bool }
            { \l__cmd_some_long_bool }
        }
        { ! \tl_if_blank_p:e { \__cmd_all_m_check:n {#2} } }
      }
      {
        \tl_set:Nx \l__cmd_fn_tl
          { \exp_not:c { \l__cmd_function_tl \c_space_tl } }
```

```
        \bool_if:NTF \l__cmd_grab_expandably_bool
          { \__cmd_declare_cmd_code_expandable:Nnn }
          { \__cmd_declare_cmd_code_aux:Nnn }
      }
      { \__cmd_declare_cmd_optimized:Nnn }
        #1 {#2}
  }
```

The optimized version of commands just has to worry about whether to make them protected or long. The commands start with an expandable marker so that other parts of the kernel know these are set up by ltcmd. We need the two layers of redirection so that the code internal function has the same form as it would for any other document command. Optimization means that there is no `\group_align_safe_begin:` before grabbing the arguments, so anything involving & tokens will not work. However, this is really only intended for making optional argument processing safe anyway, so in practice should not be an issue.

```
\cs_new_protected:Npn \__cmd_declare_cmd_optimized:Nnn #1#2#3
  {
    \bool_if:NTF \l__cmd_expandable_bool
      { \cs_set_nopar:Npe }
      { \cs_set_protected_nopar:Npe }
        #1
        {
          \exp_not:N \__cmd_start_optimized:
          \exp_not:c { \l__cmd_function_tl \c_space_tl code }
        }
    \exp_args:Ncc \cs_generate_from_arg_count:NNnn
      { \l__cmd_function_tl \c_space_tl code }
      {
        cs_set
        \bool_if:NF \l__cmd_expandable_bool { _protected }
        \bool_if:NF \l__cmd_some_long_bool { _nopar }
        :Npn
      }
      \l__cmd_current_arg_int
      {#3}
  }
\cs_new:Npn \__cmd_start_optimized: { }
```

Standard functions call `\__cmd_start:nNNnnn`, which receives the argument specification, an auxiliary used for grabbing arguments, an auxiliary containing the code, and then the signature, default arguments, and processors.

```
\cs_new_protected:Npn \__cmd_declare_cmd_code_aux:Nnn #1#2#3
  {
    \cs_generate_from_arg_count:cNnn
      { \l__cmd_function_tl \c_space_tl code }
      \cs_set_protected:Npn \l__cmd_current_arg_int {#3}
    \cs_set_protected_nopar:Npx #1
      {
        \bool_if:NTF \l__cmd_environment_bool
          {
            \__cmd_start_env:nnnnn { \exp_not:n {#2} }
              { \l__cmd_environment_str }
          }
```

```
          {
            \__cmd_start:nNNnnn { \exp_not:n {#2} }
              \exp_not:c { \l__cmd_function_tl \c_space_tl }
              \exp_not:c { \l__cmd_function_tl \c_space_tl code }
          }
          { \exp_not:o \l__cmd_signature_tl }
          {
            \bool_if:NT \l__cmd_defaults_bool
              { \exp_not:o \l__cmd_defaults_tl }
          }
          {
            \bool_if:NT \l__cmd_process_some_bool
              { \exp_not:o \l__cmd_process_all_tl }
          }
      }
  }
```

Expandable functions and functions whose arguments can be grabbed expandably call `\__cmd_start_expandable:nNNNNn`, which receives the argument specification, four auxiliaries (two for grabbing arguments, one for the code, and one for default arguments), and finally the signature. Non-expandable functions that take this branch should nevertheless be protected, as well as their `code` function. They will only be expanded in contexts such as constructing a csname. The two grabbers (named after the function with one or two spaces) are needed when there are both short and long arguments; otherwise the same grabber is included twice in the definition. If all arguments are long or all are short the (only) grabber is defined correspondingly to be long/short. Otherwise two grabbers are defined, one long, one short.

```
\cs_new_protected:Npn \__cmd_declare_cmd_code_expandable:Nnn #1#2#3
  {
    \exp_args:Ncc \cs_generate_from_arg_count:NNnn
      { \l__cmd_function_tl \c_space_tl code }
      { cs_set \bool_if:NF \l__cmd_expandable_bool { _protected } :Npn }
      \l__cmd_current_arg_int {#3}
    \bool_if:NT \l__cmd_defaults_bool
      {
        \use:x
          {
            \cs_generate_from_arg_count:cNnn
              { \l__cmd_function_tl \c_space_tl defaults }
              \cs_set:Npn \l__cmd_current_arg_int
              { \exp_not:o \l__cmd_defaults_tl }
          }
      }
    \bool_if:NTF \l__cmd_expandable_bool
      { \cs_set_nopar:Npx } { \cs_set_protected_nopar:Npx } #1
      {
        \exp_not:N \__cmd_start_expandable:nNNNNn
          { \exp_not:n {#2} }
          \exp_not:c { \l__cmd_function_tl \c_space_tl }
          \exp_not:c
            {
              \l__cmd_function_tl \c_space_tl
              \bool_if:NT \l__cmd_some_short_bool
                { \bool_if:NT \l__cmd_some_long_bool { \c_space_tl } }
```

```
            }
          \exp_not:c { \l__cmd_function_tl \c_space_tl code }
          \bool_if:NTF \l__cmd_defaults_bool
            { \exp_not:c { \l__cmd_function_tl \c_space_tl defaults } }
            { ? }
          { \exp_not:o \l__cmd_signature_tl }
      }
    \bool_if:NTF \l__cmd_some_long_bool
      {
        \bool_if:NT \l__cmd_some_short_bool
          {
            \cs_set_nopar:cpx { \l__cmd_function_tl \c_space_tl \c_space_tl }
              ##1##2 { ##1 {##2} }
          }
        \cs_set:cpx
      }
      { \cs_set_nopar:cpx }
          { \l__cmd_function_tl \c_space_tl } ##1##2 { ##1 {##2} }
  }
```

(*End of definition for* `\__cmd_declare_cmd_code:Nnn` *and others.*)

`\__cmd_declare_env:nnnn`
`\__cmd_declare_env:ennn`
`\__cmd_declare_env_internal:nnnn`
`\__cmd_set_environment_end:n`

The lead-off to creating an environment is much the same as that for creating a command: issue the appropriate message, store the argument specification then hand off to an internal function.

```
⟨latexrelease⟩\IncludeInRelease{2024/11/01}{\__cmd_declare_env:nnnn}%
⟨latexrelease⟩                  {Use~space-trimmed~envname~directly}
\cs_new_protected:Npn \__cmd_declare_env:nnnn #1#2
  {
    \str_set:Nn \l__cmd_environment_str {#1}
    \cs_if_exist:cTF { #1 }
      { \msg_info:nnnn { cmd } { redefine-env } { #1 } { #2 } }
      { \msg_info:nnnn { cmd } { define-env } { #1 } { #2 } }
    \bool_set_false:N \l__cmd_expandable_bool
    \bool_set_true:N \l__cmd_environment_bool
    \__cmd_declare_env_internal:nnnn {#1} {#2}
  }
\cs_generate_variant:Nn \__cmd_declare_env:nnnn { e }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2024/06/01}{\__cmd_declare_env:nnnn}%
⟨latexrelease⟩                  {Use~space-trimmed~envname~directly}
⟨latexrelease⟩\cs_new_protected:Npn \__cmd_declare_env:nnnn #1#2
⟨latexrelease⟩  {
⟨latexrelease⟩    \str_set:Nx \l__cmd_environment_str {#1}
⟨latexrelease⟩    \str_set:Nx \l__cmd_environment_str
⟨latexrelease⟩      { \tl_trim_spaces:o { \l__cmd_environment_str } }
⟨latexrelease⟩    \cs_if_exist:cTF { \l__cmd_environment_str }
⟨latexrelease⟩      {
⟨latexrelease⟩        \msg_info:nnxx { cmd } { redefine-env }
⟨latexrelease⟩          { \l__cmd_environment_str } { \tl_to_str:n {#2} }
⟨latexrelease⟩      }
⟨latexrelease⟩      {
⟨latexrelease⟩        \msg_info:nnxx { cmd } { define-env }
⟨latexrelease⟩          { \l__cmd_environment_str } { \tl_to_str:n {#2} }
```

```
⟨latexrelease⟩        }
⟨latexrelease⟩      \bool_set_false:N \l__cmd_expandable_bool
⟨latexrelease⟩      \bool_set_true:N \l__cmd_environment_bool
⟨latexrelease⟩      \exp_args:NV \__cmd_declare_env_internal:nnnn
⟨latexrelease⟩        \l__cmd_environment_str {#2}
⟨latexrelease⟩    }
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

Creating a document environment requires a few more steps than creating a single command. In order to pass the arguments of the command to the end of the function, it is necessary to store the grabbed arguments. To do that, the function used at the end of the environment has to be redefined to contain the appropriate information. To minimize the amount of expansion at point of use, the code here is expanded now as well as when used. The last argument of `\__cmd_declare_cmd_internal:Nnnn` is only run if the definition succeeded. In package mode this ensures that the original definition of the environment is not changed if the definition fails for any reason. This also avoids an error when defining the `end␣aux␣` function when the user asks for more than 9 arguments.

```
\cs_new_protected:Npn \__cmd_declare_env_internal:nnnn #1#2#3#4
  {
    \exp_args:Nc \__cmd_declare_cmd_internal:Nnnn { environment~ #1 } {#2}
      {#3}
      {
        \cs_set_nopar:cpx { environment~ #1 ~end }
          { \exp_not:c { environment~ #1 ~end~aux } }
        \cs_generate_from_arg_count:cNnn
          { environment~ #1 ~end~aux~ } \cs_set:Npn
          \l__cmd_current_arg_int {#4}
        \cs_set_eq:cc {#1}       { environment~ #1 }
        \cs_set_eq:cc { end #1 } { environment~ #1 ~end }
      }
  }
\cs_new_protected:Npn \__cmd_set_environment_end:n #1
  {
    \cs_set_nopar:cpx { environment~ #1 ~end~aux }
      {
        \exp_not:c { environment~ #1 ~end~aux~ }
        \exp_not:o \l__cmd_args_tl
      }
  }
```

(*End of definition for* `\__cmd_declare_env:nnnn`, `\__cmd_declare_env_internal:nnnn`, *and* `\__cmd_set_environment_end:n`.)

1.3 Structure of xparse commands

`\__cmd_start_env:nnnnn`
`\__cmd_start:nNNnnn`

For error messages that occur during run-time when getting arguments of environments it is necessary to keep track of the environment name. We begin non-expandable commands with a token equal to `\scan_stop:`, whose name gives a reasonable error message if the command is used inside a csname and protects against `f`-expansion. This is useless for environments since `\begin` is already not expandable. Both the command and environment codes start with `\group_align_safe_begin:`, then `\__cmd_run_code:` (used

by both) does `\group_align_safe_end:`, so that delimited arguments may be grabbed in alignments if they contain and alignment tab token (see latex3/latex3/issues/839).

```
\cs_new_protected:Npn \__cmd_start_env:nnnnn #1#2
  {
    \conditionally@traceoff
    \group_align_safe_begin:
    \str_set:Nn \l__cmd_environment_str {#2}
    \bool_set_true:N \l__cmd_environment_bool
    \__cmd_start_aux:ccnnnn
      { environment~ \l__cmd_environment_str \c_space_tl }
      { environment~ \l__cmd_environment_str \c_space_tl code }
      {#1}
  }
\cs_new_protected:Npx \__cmd_start:nNNnnn #1#2#3
  {
    \exp_not:c { xparse~function~is~not~expandable }
    \exp_not:N \conditionally@traceoff
    \exp_not:N \group_align_safe_begin:
    \exp_not:n { \bool_set_false:N \l__cmd_environment_bool }
    \exp_not:N \__cmd_start_aux:NNnnnn
    #2 #3 {#1}
  }
```

(*End of definition for* `\__cmd_start_env:nnnnn` *and* `\__cmd_start:nNNnnn`.)

`\__cmd_start_aux:NNnnnn`
`\__cmd_start_aux:ccnnnn`

This sets up a few variables to minimize the boilerplate code included in all xparse-defined commands. It then runs the grabbers `#4`. Again, the argument specification `#1` is only for diagnostics.

```
\cs_new_protected:Npn \__cmd_start_aux:NNnnnn #1#2#3#4#5#6
  {
    \tl_clear:N \l__cmd_args_tl
    \tl_set:Nn \l__cmd_fn_tl {#1}
    \tl_set:Nn \l__cmd_fn_code_tl {#2}
    \tl_set:Nn \l__cmd_defaults_tl {#5}
    \tl_set:Nn \l__cmd_process_all_tl {#6}
    #4
    \__cmd_run_code:
  }
\cs_generate_variant:Nn \__cmd_start_aux:NNnnnn { cc }
```

(*End of definition for* `\__cmd_start_aux:NNnnnn`.)

`\__cmd_run_code:`

After arguments are grabbed, this function is responsible for inserting default values, running processors, and finally doing `\group_align_safe_end:` as promised, and running the code.

```
\cs_new_protected:Npn \__cmd_run_code:
  {
    \tl_if_empty:NF \l__cmd_defaults_tl { \__cmd_defaults: }
    \tl_if_empty:NF \l__cmd_process_all_tl { \__cmd_args_process: }
    \bool_if:NT \l__cmd_environment_bool
      { \exp_args:No \__cmd_set_environment_end:n \l__cmd_environment_str }
    \group_align_safe_end:
    \conditionally@traceon
    \exp_after:wN \l__cmd_fn_code_tl \l__cmd_args_tl
  }
```

(*End of definition for* `\__cmd_run_code:`.)

`\__cmd_defaults:`
`\__cmd_defaults_def:`
`\__cmd_defaults_def:nn`
`\__cmd_defaults_def:nnn`
`\__cmd_defaults_aux:`
`\__cmd_defaults_error:w`

First construct `\__cmd_tmp:w` (see below) that will receive the arguments found so far and determine default values for any missing argument. Then call it repeatedly until the set of arguments stabilizes. Since that could lead to an infinite loop we only call it up to nine times, the maximal number needed for stabilization if there is a chain of arguments that depend on each other. If that fails to stabilize raise an error.

```
\cs_new_protected:Npn \__cmd_defaults:
  {
    \__cmd_defaults_def:
    \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_tl
    \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
    \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
    \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
    \__cmd_defaults_error:w
    \q_recursion_stop
    \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_i_tl
  }
\cs_new_protected:Npn \__cmd_defaults_aux:
  {
    \tl_set:Nx \l__cmd_args_ii_tl
      { \exp_after:wN \__cmd_tmp:w \l__cmd_args_i_tl }
    \tl_if_eq:NNT \l__cmd_args_ii_tl \l__cmd_args_i_tl
      { \use_none_delimit_by_q_recursion_stop:w }
    \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_ii_tl
  }
\cs_new_protected:Npn \__cmd_defaults_error:w \q_recursion_stop
  {
    \msg_error:nnx { cmd } { default-loop }
      { \__cmd_environment_or_command: }
  }
```

To construct `\__cmd_tmp:w`, first go through the arguments found and the corresponding defaults, building a token list with `{#`⟨*arg number*⟩`}` for arguments found in the input (whose default will not be used) and otherwise `{\exp_not:n{`⟨*default*⟩`}}` for arguments whose default will be used.

```
\cs_new_protected:Npn \__cmd_defaults_def:
  {
    \tl_clear:N \l__cmd_tmpa_tl
    \int_zero:N \l__cmd_current_arg_int
    \__cmd_tl_mapthread_function:NNN \l__cmd_args_tl \l__cmd_defaults_tl
      \__cmd_defaults_def:nn
    \cs_generate_from_arg_count:NNno \__cmd_tmp:w \cs_set:Npn
      \l__cmd_current_arg_int \l__cmd_tmpa_tl
  }
\cs_generate_variant:Nn \cs_generate_from_arg_count:NNnn { NNno }
\cs_new_protected:Npn \__cmd_defaults_def:nn
  {
    \int_incr:N \l__cmd_current_arg_int
    \exp_args:NV \__cmd_defaults_def:nnn \l__cmd_current_arg_int
  }
\cs_new_protected:Npn \__cmd_defaults_def:nnn #1#2#3
  {
    \tl_put_right:Nx \l__cmd_tmpa_tl
```

```
      {
        {
          \exp_not:N \exp_not:n
            {
              \tl_if_novalue:nTF {#2}
                { \exp_not:o {#3} }
                { \exp_not:n { ## #1 } }
            }
        }
      }
  }
```

(*End of definition for* `\__cmd_defaults:` *and others.*)

`\__cmd_args_process:`
`\__cmd_args_process_loop:nn`
`\__cmd_args_process_aux:n`

Loop through arguments (stored in `\l__cmd_args_tl`) and the corresponding processors (in `\l__cmd_process_all_tl`) simultaneously, apply all processors for each argument and store the result back into `\l__cmd_args_tl`. To allow processors to depend on other arguments, for every processor define a temporary auxiliary that receives all arguments `\l__cmd_args_tl`.

```
\cs_new_protected:Npn \__cmd_args_process:
  {
    \tl_clear:N \l__cmd_args_ii_tl
    \__cmd_tl_mapthread_function:NNN
      \l__cmd_args_tl
      \l__cmd_process_all_tl
      \__cmd_args_process_loop:nn
    \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_ii_tl
  }
\cs_new_protected:Npn \__cmd_args_process_loop:nn #1#2
  {
    \tl_set:Nn \ProcessedArgument {#1}
    \tl_if_novalue:nF {#1}
      { \tl_map_function:nN {#2} \__cmd_args_process_aux:n }
    \tl_put_right:No \l__cmd_args_ii_tl
      { \exp_after:wN { \ProcessedArgument } }
  }
\cs_new_protected:Npn \__cmd_args_process_aux:n #1
  {
    \cs_generate_from_arg_count:NNnn \__cmd_tmp:w \cs_set:Npn
      { \tl_count:N \l__cmd_args_tl } {#1}
    \exp_args:NNNo \exp_after:wN \__cmd_tmp:w \l__cmd_args_tl
      { \ProcessedArgument }
  }
```

(*End of definition for* `\__cmd_args_process:`*,* `\__cmd_args_process_loop:nn`*, and* `\__cmd_args_process_aux:n`*.*)

`\__cmd_start_expandable:nNNNNn`

This is called for all expandable commands. `#6` is the signature, responsible for grabbing arguments. `#5` is used to determine default values (or is `?` if there are none). `#4` is the code to run. `#2` and `#3` are functions (named after the command) that grab a single argument in the input stream (`#3` is short). The argument specification `#1` is only used by diagnostic functions. Same as for the non-expandable version, this starts with `\group_-align_safe_begin:`, which expands to nothing, so may be safely used in an expandable context.

```
\cs_new:Npn \__cmd_start_expandable:nNNNNn #1#2#3#4#5#6
  {
    \group_align_safe_begin:
    #6 \__cmd_end_expandable:NNw #5 #4 \q__cmd #2#3
  }
```

(*End of definition for* `\__cmd_start_expandable:nNNNNn`.)

`\__cmd_end_expandable:NNw`
`\__cmd_end_expandable_aux:w`
`\__cmd_end_expandable_aux:nNNNN`
`\__cmd_end_expandable_defaults:nnnNNn`
`\__cmd_end_expandable_defaults:nnw`
`\__cmd_end_expandable_defaults:nw`

Followed by a function `#1` to determine default values (or `?` if there are no defaults), the code `#2`, arguments that have been grabbed, then `\q__cmd` and two generic grabbers. The idea to find default values is similar to the non-expandable case but we cannot define an auxiliary function, so at every step in the loop we need to go through all arguments searching for which ones started out as `-NoValue-` and replacing these by the newly computed values. In fact we need to keep track of three versions of all arguments: the original version, the previous version with default values, and the currently built version (first argument of `\__cmd_end_expandable_defaults:nnnNNn`).

```
\cs_new:Npn \__cmd_end_expandable:NNw #1#2
  { \__cmd_end_expandable_aux:w #1#2 \prg_do_nothing: }
\cs_new:Npn \__cmd_end_expandable_aux:w #1#2#3 \q__cmd
  { \exp_args:No \__cmd_end_expandable_aux:nNNNN {#3} #1 #2 }
\cs_new:Npn \__cmd_end_expandable_aux:nNNNN #1#2#3#4#5
  {
    \token_if_eq_charcode:NNT ? #2 { \exp_after:wN \use_iv:nnnn }
    \__cmd_end_expandable_defaults:nnnNNn {#1} { } {#1} #2#3
      { } { } { } { } { } { } { } { } { } { }
      {
        \msg_expandable_error:nnf { cmd } { default-loop }
          { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N #4 } }
        \use_iv:nnnn
      }
    \q_stop
  }
\cs_new:Npn \__cmd_end_expandable_defaults:nnnNNn #1#2#3#4#5#6
  {
    #6
    \str_if_eq:nnTF {#1} {#2}
      { \use_i_delimit_by_q_stop:nw { \group_align_safe_end: #5 #1 } }
      {
        \exp_args:No \__cmd_tl_mapthread_function:nnN
          { #4 #1 } {#3}
          \__cmd_end_expandable_defaults:nnw
        \__cmd_end_expandable_defaults:nnnNNn { } {#1} {#3} #4 #5
      }
  }
\cs_new:Npn \__cmd_end_expandable_defaults:nnw #1#2
  {
    \tl_if_novalue:nTF {#2}
      { \exp_args:No \__cmd_end_expandable_defaults:nw {#1} }
      { \__cmd_end_expandable_defaults:nw {#2} }
  }
\cs_new:Npn \__cmd_end_expandable_defaults:nw
    #1#2 \__cmd_end_expandable_defaults:nnnNNn #3
  { #2 \__cmd_end_expandable_defaults:nnnNNn { #3 {#1} } }
```

(*End of definition for* `\__cmd_end_expandable:NNw` *and others.*)

1.4 Normalizing the argument specifications

The goal here is to expand aliases and check that the argument specification is valid before the main parsing run. If it is not valid the entire set up is abandoned to avoid any strange internal errors. A function is provided for each argument type that will grab any extra data items and call the loop function after performing the following checks and tasks.

- Check that each argument has the correct number of data items associated with it, and that where a single character is required, one has actually been supplied.
- Check that processors and the markers `+`, `!` and `=` are followed by an argument for which they make sense, and are not redundant.
- Check the absence of forbidden types for expandable commands, namely `G`/`v` always, and `l`/`u` after optional arguments (`xparse` may have inserted braces due to a failed search for an optional argument).
- Check that no optional argument is followed by a mandatory argument with the same delimiter, as otherwise the optional argument could never be omitted.
- Keep track in `\l__cmd_some_long_bool` and `\l__cmd_some_short_bool` of whether the command has some long/short arguments.
- Keep track in `\l__cmd_grab_expandably_bool` of whether all arguments are `m`/`l`/`u` type and short arguments appear before long ones, in which case they can be grabbed expandably just as safely as they could be grabbed nonexpandably. Regardless of that, arguments of expandable commands will be grabbed expandably and arguments of environments will not (because the list of arguments built by non-expandable grabbing is used to pass them to the end-environment code).

Further checks happen at the end of the loop:

- that there are at most 9 arguments;
- that an expandable command does not end with an optional argument (this case is detected by using the fact that `\l__cmd_last_delimiters_tl` is cleared by every mandatory argument and filled by every optional argument).

`\__cmd_normalize_arg_spec:n`
`\__cmd_normalize_arg_spec_loop:n`

Loop through the argument specification, calling an auxiliary specific to each argument type. If any argument is unknown stop the definition.

```
\cs_new_protected:Npn \__cmd_normalize_arg_spec:n #1
  {
    \int_zero:N \l__cmd_current_arg_int
    \tl_clear:N \l__cmd_last_delimiters_tl
    \tl_clear:N \l__cmd_arg_spec_tl
    \bool_set_true:N \l__cmd_grab_expandably_bool
    \bool_set_false:N \l__cmd_obey_spaces_bool
    \bool_set_false:N \l__cmd_long_bool
    \bool_set_false:N \l__cmd_suppress_strip_bool
    \bool_set_false:N \l__cmd_some_obey_spaces_bool
    \bool_set_false:N \l__cmd_some_long_bool
    \bool_set_false:N \l__cmd_some_short_bool
    \__cmd_normalize_arg_spec_loop:n #1
      \q_recursion_tail \q_recursion_tail \q_recursion_tail \q_recursion_stop
```

```
    \int_compare:nNnT \l__cmd_current_arg_int > 9
      {
        \msg_error:nnxx { cmd } { too-many-args }
          { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
        \__cmd_bad_def:wn
      }
    \bool_if:NT \l__cmd_expandable_bool
      {
        \tl_if_empty:NF \l__cmd_last_delimiters_tl
          {
            \msg_error:nnxx { cmd } { expandable-ending-optional }
              { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
            \__cmd_bad_def:wn
          }
      }
    \bool_if:NT \l__cmd_expandable_bool
      { \bool_set_true:N \l__cmd_grab_expandably_bool }
    \bool_if:NT \l__cmd_environment_bool
      { \bool_set_false:N \l__cmd_grab_expandably_bool }
  }
\cs_new_protected:Npn \__cmd_normalize_arg_spec_loop:n #1
  {
    \quark_if_recursion_tail_stop:n {#1}
    \int_incr:N \l__cmd_current_arg_int
    \cs_if_exist_use:cF { __cmd_normalize_type_ \tl_to_str:n {#1} :w }
      {
        \bool_lazy_any:nTF
          {
            { \str_if_eq_p:nn {#1} { G } }
            { \str_if_eq_p:nn {#1} { g } }
            { \str_if_eq_p:nn {#1} { l } }
            { \str_if_eq_p:nn {#1} { u } }
          }
          {
            \msg_error:nnxx { cmd } { xparse-arg-type }
              { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
          }
          {
            \msg_error:nnxx { cmd } { unknown-argument-type }
              { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
          }
        \__cmd_bad_def:wn
      }
  }
```

(*End of definition for* `\__cmd_normalize_arg_spec:n` *and* `\__cmd_normalize_arg_spec_loop:n`.)

`\__cmd_normalize_type_d:w`
`\__cmd_normalize_type_e:w`
`\__cmd_normalize_type_o:w`
`\__cmd_normalize_type_O:w`
`\__cmd_normalize_type_r:w`
`\__cmd_normalize_type_s:w`

These argument types are aliases of more general ones, for example with the default argument `-NoValue-`. To easily insert that marker expanded in the definitions we call `\__cmd_tmp:w` with the argument `-NoValue-`. For argument types that need additional data, check that the data is present (not `\q_recursion_tail`) before proceeding.

```
\cs_set_protected:Npn \__cmd_tmp:w #1
  {
    \cs_new_protected:Npn \__cmd_normalize_type_d:w ##1##2
```

```
      {
        \quark_if_recursion_tail_stop_do:nn {##2} { \__cmd_bad_arg_spec:wn }
        \__cmd_normalize_type_D:w {##1} {##2} {#1}
      }
    \cs_new_protected:Npn \__cmd_normalize_type_e:w ##1
      {
        \quark_if_recursion_tail_stop_do:nn {##1} { \__cmd_bad_arg_spec:wn }
        \__cmd_normalize_type_E:w {##1} { }
      }
    \cs_new_protected:Npn \__cmd_normalize_type_o:w
      { \__cmd_normalize_type_D:w [ ] {#1} }
    \cs_new_protected:Npn \__cmd_normalize_type_O:w
      { \__cmd_normalize_type_D:w [ ] }
    \cs_new_protected:Npn \__cmd_normalize_type_r:w ##1##2
      {
        \quark_if_recursion_tail_stop_do:nn {##2} { \__cmd_bad_arg_spec:wn }
        \__cmd_normalize_type_R:w {##1} {##2} {#1}
      }
    \cs_new_protected:Npn \__cmd_normalize_type_s:w
      { \__cmd_normalize_type_t:w * }
  }
\exp_args:No \__cmd_tmp:w { \c_novalue_tl }
```

(*End of definition for* `\__cmd_normalize_type_d:w` *and others.*)

`\__cmd_normalize_type_>:w`
`\__cmd_normalize_type_+:w`
`\__cmd_normalize_type_!:w`
`\__cmd_normalize_type_=:w`
`\__cmd_normalize_type_aux:NnNn`

Check that these prefixes have arguments, namely that the next token is not `\q_recursion_tail`, and remember to leave it after the looping macro. Processors are forbidden in expandable commands. If all is good, store the prefix in the cleaned up `\l__cmd_arg_spec_tl`, and decrement the argument number as prefixes do not correspond to arguments.

```
\cs_new_protected:cpn { __cmd_normalize_type_>:w } #1#2
  {
    \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
    \bool_if:NT \l__cmd_expandable_bool
      {
        \msg_error:nnxx { cmd } { processor-in-expandable }
          { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
        \__cmd_bad_def:wn
      }
    \tl_put_right:Nx \l__cmd_arg_spec_tl { > { \tl_trim_spaces:n {#1} } }
    \int_decr:N \l__cmd_current_arg_int
    \bool_set_false:N \l__cmd_grab_expandably_bool
    \__cmd_normalize_arg_spec_loop:n {#2}
  }
\cs_new_protected:cpn { __cmd_normalize_type_+:w } #1
  {
    \__cmd_normalize_type_aux:NnNn + {#1}
      \l__cmd_long_bool
      { \bool_set_true:N \l__cmd_long_bool }
  }
\cs_new_protected:cpn { __cmd_normalize_type_!:w } #1
  {
    \__cmd_normalize_type_aux:NnNn ! {#1}
      \l__cmd_obey_spaces_bool
```

```
      {
        \bool_set_true:N \l__cmd_obey_spaces_bool
        \bool_set_true:N \l__cmd_some_obey_spaces_bool
      }
  }
\cs_new_protected:cpn { __cmd_normalize_type_=:w } #1#2
  {
    \__cmd_normalize_type_aux:NnNn = {#2}
      \l__cmd_suppress_strip_bool
      {
        \bool_if:NT \l__cmd_expandable_bool
          {
            \msg_error:nnxx { cmd } { keyval-in-expandable }
              { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
            \__cmd_bad_def:wn
          }
        \bool_set_true:N \l__cmd_suppress_strip_bool
        \bool_set_false:N \l__cmd_grab_expandably_bool
        \tl_put_right:Nx \l__cmd_arg_spec_tl
          { = { \tl_trim_spaces:n {#1} } }
      }
  }
\cs_new_protected:Npn \__cmd_normalize_type_aux:NnNn #1#2#3#4
  {
    \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
    \bool_if:NT #3
      {
        \msg_error:nnxx { cmd } { two-markers }
          { \__cmd_environment_or_command: } { #1 }
        \__cmd_bad_def:wn
      }
    #4
    \int_decr:N \l__cmd_current_arg_int
    \__cmd_normalize_arg_spec_loop:n {#2}
  }
```

(*End of definition for* `\__cmd_normalize_type_>:w` *and others.*)

`\__cmd_normalize_type_D:w`
`\__cmd_normalize_type_E:w`
`\__cmd_normalize_type_t:w`
`\__cmd_normalize_E_unique_check:w`

Optional argument types. Check that all required data is present (and consists of single characters if applicable) and check for forbidden types for expandable commands. For E-type require that there is at least one embellishment, that each one is a single character, and that there aren't more optional arguments than embellishments; also remember that each embellishment counts as one argument for `\l__cmd_current_arg_int`. Then in each case store the data in `\l__cmd_arg_spec_tl`, and for later checks store in `\l__cmd_last_delimiters_tl` the tokens whose presence determines whether there is an optional argument (for braces store `{}`, seen later as an empty delimiter).

```
\cs_new_protected:Npn \__cmd_normalize_type_D:w #1#2#3
  {
    \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
    \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
    \__cmd_single_token_check:n {#2}
    \__cmd_add_arg_spec:n { D #1 #2 {#3} }
    \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
    \bool_set_false:N \l__cmd_grab_expandably_bool
```

```
    \__cmd_normalize_arg_spec_loop:n
  }
\cs_new_protected:Npn \__cmd_normalize_type_E:w #1#2
  {
    \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
    \tl_if_blank:nT {#1} { \__cmd_bad_arg_spec:wn }
    \tl_map_function:nN {#1} \__cmd_single_token_check:n
    \tl_map_function:nN {#1} \__cmd_allowed_token_check:N
    \__cmd_normalize_E_unique_check:w #1 \q_nil \q_stop
    \int_compare:nNnT { \tl_count:n {#2} } > { \tl_count:n {#1} }
      { \__cmd_bad_arg_spec:wn }
    \__cmd_add_arg_spec:n { E {#1} {#2} }
    \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
    \bool_set_false:N \l__cmd_grab_expandably_bool
    \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#1} - 1 }
    \__cmd_normalize_arg_spec_loop:n
  }
\cs_new_protected:Npn \__cmd_normalize_E_unique_check:w #1#2 \q_stop
  {
    \quark_if_nil:NF #1
      {
        \tl_if_in:nnT {#2} {#1} { \__cmd_bad_arg_spec:wn }
        \__cmd_normalize_E_unique_check:w #2 \q_stop
      }
  }
\cs_new_protected:Npn \__cmd_normalize_type_t:w #1
  {
    \quark_if_recursion_tail_stop_do:Nn #1 { \__cmd_bad_arg_spec:wn }
    \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
    \tl_put_right:Nx \l__cmd_arg_spec_tl
      {
        \bool_if:NT \l__cmd_obey_spaces_bool { ! }
        t \exp_not:n {#1}
      }
    \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
    \bool_set_false:N \l__cmd_grab_expandably_bool
    \bool_set_false:N \l__cmd_obey_spaces_bool
    \bool_set_false:N \l__cmd_long_bool
    \__cmd_normalize_arg_spec_loop:n
  }
```

(*End of definition for* `\__cmd_normalize_type_D:w` *and others.*)

`\__cmd_normalize_type_m:w`
`\__cmd_normalize_type_R:w`
`\__cmd_normalize_type_v:w`

Mandatory arguments. First check the required data is present, consists of single characters where applicable, and that the argument type is allowed for expandable commands if applicable. For the `m` and `R` argument types check that they do not follow some optional argument with that delimiter as otherwise the optional argument could not be omitted. Then save data in `\l__cmd_arg_spec_tl`, count the mandatory argument, and empty the list of last delimiters.

```
\cs_new_protected:Npn \__cmd_normalize_type_m:w
  {
    \__cmd_delimiter_check:nnn { } { m } { \iow_char:N \{ }
    \__cmd_add_arg_spec_mandatory:n { m }
    \__cmd_normalize_arg_spec_loop:n
```

```
  }
\cs_new_protected:Npn \__cmd_normalize_type_R:w #1#2#3
  {
    \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
    \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
    \__cmd_single_token_check:n {#2}
    \__cmd_delimiter_check:nnn {#1} { R/r } { \tl_to_str:n {#1} }
    \bool_set_false:N \l__cmd_grab_expandably_bool
    \__cmd_add_arg_spec_mandatory:n { R #1 #2 {#3} }
    \__cmd_normalize_arg_spec_loop:n
  }
\cs_new_protected:Npn \__cmd_normalize_type_v:w
  {
    \__cmd_normalize_check_gv:N v
    \__cmd_add_arg_spec_mandatory:n { v }
    \__cmd_normalize_arg_spec_loop:n
  }
```

(*End of definition for* `\__cmd_normalize_type_m:w`, `\__cmd_normalize_type_R:w`, *and* `\__cmd_normalize_type_v:w`.)

`\__cmd_normalize_type_b:w` This argument type is not allowed for commands. This is only allowed at the end of the argument specification, hence we check that `#1` is the end.

```
\cs_new_protected:Npn \__cmd_normalize_type_b:w #1
  {
    \bool_if:NF \l__cmd_environment_bool
      {
        \msg_error:nnxx { cmd } { invalid-command-arg }
          { \__cmd_environment_or_command: } { b }
        \__cmd_bad_def:wn
      }
    \tl_clear:N \l__cmd_last_delimiters_tl
    \__cmd_add_arg_spec:n { b }
    \quark_if_recursion_tail_stop:n {#1}
    \msg_error:nnxx { cmd } { arg-after-body }
      { \__cmd_environment_or_command: }
      { \tl_to_str:n {#1} }
    \__cmd_bad_def:wn
  }
```

(*End of definition for* `\__cmd_normalize_type_b:w`.)

`\__cmd_single_token_check:n` Checks that the argument is a single (non-space) token (possibly surrounded by spaces), and aborts the definition otherwise.

```
\cs_new_protected:Npn \__cmd_single_token_check:n #1
  {
    \tl_trim_spaces_apply:nN {#1} \tl_if_single_token:nF
      {
        \msg_error:nnxx { cmd } { not-single-token }
          { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
        \__cmd_bad_def:wn
      }
  }
```

(*End of definition for* `\__cmd_single_token_check:n`.)

__cmd_allowed_token_check:N

Some tokens are not allowed as delimiters for some argument types, notably implicit begin/end-group tokens (\bgroup/\egroup). The major problem with these tokens is that for \peek_... functions, a literal $\{_1$ is virtually indistinguishable from a \bgroup or other token which was \let to a $\{_1$, and the same goes for $\}_2$. All other tokens can be easily distinguished from their implicit counterparts by grabbing them and looking at the string length (see __cmd_token_if_cs:NTF), but for begin/end group tokens that is not possible without the risk of mistakenly grabbing the entire brace group (potentially leading to a ! Runaway argument error) or trying to grab a $\}_2$, leading to an ! Argument of \dots has an extra } error.

```
\cs_new_protected:Npn \__cmd_allowed_token_check:N #1
  {
    \token_if_eq_meaning:NNTF #1 \c_group_begin_token
        { \use:n }
        {
          \token_if_eq_meaning:NNTF #1 \c_group_end_token
            { \use:n }
            { \use_none:n }
        }
      {
        \msg_error:nnxxx { cmd } { forbidden-group-token }
          { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
          {
            \token_if_eq_meaning:NNTF #1 \c_group_begin_token
              { begin } { end }
          }
        \__cmd_bad_def:wn
      }
  }
```

(*End of definition for* __cmd_allowed_token_check:N.)

__cmd_normalize_check_gv:N
__cmd_normalize_check_lu:N

Called for arguments that are always forbidden, or forbidden after an optional argument, for expandable commands.

```
\cs_new_protected:Npn \__cmd_normalize_check_gv:N #1
  {
    \bool_if:NT \l__cmd_expandable_bool
      {
        \msg_error:nnxx { cmd } { invalid-expandable-arg }
          { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
        \__cmd_bad_def:wn
      }
    \bool_set_false:N \l__cmd_grab_expandably_bool
  }
\cs_new_protected:Npn \__cmd_normalize_check_lu:N #1
  {
    \bool_if:NT \l__cmd_expandable_bool
      {
        \tl_if_empty:NF \l__cmd_last_delimiters_tl
          {
            \msg_error:nnxx { cmd } { invalid-after-optional-expandably }
              { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
            \__cmd_bad_def:wn
          }
```

```
      }
  }
```

(*End of definition for* `\__cmd_normalize_check_gv:N` *and* `\__cmd_normalize_check_lu:N`.)

`\__cmd_delimiter_check:nnn` Called for `m` and `R` arguments. Checks that the leading token does not coincide with the token denoting the presence of a previous optional argument. Instead of dealing with braces for the `m`-type we use an empty delimiter to denote that case.

```
\cs_new_protected:Npn \__cmd_delimiter_check:nnn #1#2#3
  {
    \tl_map_inline:Nn \l__cmd_last_delimiters_tl
      {
        \tl_if_eq:nnT {##1} {#1}
          {
            \msg_warning:nnxx { cmd } { optional-mandatory }
              {#2} {#3}
          }
      }
  }
```

(*End of definition for* `\__cmd_delimiter_check:nnn`.)

`\__cmd_bad_arg_spec:wn` `\__cmd_bad_def:wn` If the argument specification is wrong, this provides an escape from the entire definition process.

```
\cs_new_protected:Npn \__cmd_bad_arg_spec:wn #1 \__cmd_break_point:n #2
  {
    \msg_error:nnxx { cmd } { bad-arg-spec }
      { \__cmd_environment_or_command: } { \tl_to_str:n {#2} }
  }
\cs_new_protected:Npn \__cmd_bad_def:wn #1 \__cmd_break_point:n #2 { }
```

(*End of definition for* `\__cmd_bad_arg_spec:wn` *and* `\__cmd_bad_def:wn`.)

`\__cmd_add_arg_spec:n` `\__cmd_add_arg_spec_mandatory:n` When adding an argument to the argument specification, set the `some_long` or `some_short` booleans as appropriate and clear the booleans keeping track of `+`, `!` and `=` markers. Before that, test for a short argument following some long arguments: this is forbidden for expandable commands and prevents grabbing arguments expandably.

For mandatory arguments do some more work, in particular complain if they were preceded by `!`.

```
\cs_new_protected:Npn \__cmd_add_arg_spec:n #1
  {
    \bool_lazy_and:nnT
      { ! \l__cmd_long_bool }
      { \l__cmd_some_long_bool }
      {
        \bool_if:NT \l__cmd_expandable_bool
          {
            \msg_error:nnx { cmd } { long-short-mix }
              { \iow_char:N \\ \l__cmd_function_tl }
            \__cmd_bad_def:wn
          }
        \bool_set_false:N \l__cmd_grab_expandably_bool
      }
    \bool_if:NTF \l__cmd_long_bool
```

```
      { \bool_set_true:N \l__cmd_some_long_bool }
      { \bool_set_true:N \l__cmd_some_short_bool }
    \tl_put_right:Nx \l__cmd_arg_spec_tl
      {
        \bool_if:NT \l__cmd_long_bool { + }
        \bool_if:NT \l__cmd_obey_spaces_bool { ! }
        \exp_not:n {#1}
      }
    \bool_set_false:N \l__cmd_long_bool
    \bool_set_false:N \l__cmd_obey_spaces_bool
  }
\cs_new_protected:Npn \__cmd_add_arg_spec_mandatory:n #1
  {
    \bool_if:NT \l__cmd_some_obey_spaces_bool
      {
        \msg_error:nnxx { cmd } { invalid-bang }
          { \__cmd_environment_or_command: }
          {
            \bool_if:NTF \l__cmd_obey_spaces_bool
              { \tl_to_str:n {'#1'} }
              { an~optional~argument~before~mandatory~ \tl_to_str:n {'#1'} }
          }
        \__cmd_bad_def:wn
      }
    \tl_clear:N \l__cmd_last_delimiters_tl
    \__cmd_add_arg_spec:n {#1}
  }
```

(*End of definition for* `\__cmd_add_arg_spec:n` *and* `\__cmd_add_arg_spec_mandatory:n`.)

1.5 Preparing the signature: general mechanism

`\__cmd_prepare_signature:n`
`\__cmd_prepare_signature:N`
`\__cmd_prepare_signature_bypass:N`

Actually creating the signature uses the same loop approach as normalizing the signature. There are first a number of variables which need to be set to track what is going on. Many of these variables are unused when defining expandable commands.

```
\cs_new_protected:Npn \__cmd_prepare_signature:n #1
  {
    \int_zero:N \l__cmd_current_arg_int
    \bool_set_false:N \l__cmd_long_bool
    \bool_set_false:N \l__cmd_obey_spaces_bool
    \bool_set_false:N \l__cmd_suppress_strip_bool
    \int_zero:N \l__cmd_m_args_int
    \bool_set_false:N \l__cmd_defaults_bool
    \tl_clear:N \l__cmd_defaults_tl
    \tl_clear:N \l__cmd_process_all_tl
    \tl_clear:N \l__cmd_process_one_tl
    \bool_set_false:N \l__cmd_process_some_bool
    \tl_clear:N \l__cmd_signature_tl
    \__cmd_prepare_signature:N #1 \q_recursion_tail \q_recursion_stop
    \bool_if:NF \l__cmd_expandable_bool { \__cmd_flush_m_args: }
  }
```

The main looping function does not take an argument, but carries out the reset on the processor boolean. This is split off from the rest of the process so that when actually setting up processors the flag-reset can be bypassed.

For each known argument type there is an appropriate function to actually do the addition to the signature. These are separate for expandable and standard functions, as the approaches are different.

```
\cs_new_protected:Npn \__cmd_prepare_signature:N
  {
    \bool_set_false:N \l__cmd_prefixed_bool
    \__cmd_prepare_signature_bypass:N
  }
\cs_new_protected:Npn \__cmd_prepare_signature_bypass:N #1
  {
    \quark_if_recursion_tail_stop:N #1
    \use:c
      {
         __cmd_add
         \bool_if:NT \l__cmd_grab_expandably_bool { _expandable }
         _type_  \token_to_str:N #1 :w
      }
  }
```

(*End of definition for* `\__cmd_prepare_signature:n`, `\__cmd_prepare_signature:N`, *and* `\__cmd_prepare_signature_bypass:N`.)

1.6 Setting up a standard signature

Each argument-adding function appends to the signature a grabber (and for some types, the delimiters or default value), except the one for `m` arguments. These are collected and added to the signature all at once by `\__cmd_flush_m_args:`, called for every other argument type. All of the functions then call the loop function `\__cmd_prepare_signature:N`. Default values of arguments are collected by `\__cmd_add_default:n` rather than being stored with the argument; this function and `\__cmd_add_default:` are also responsible for keeping track of `\l__cmd_current_arg_int`.

`\__cmd_add_type_+:w` Making the next argument long means setting the flag. The `m` arguments are recorded here as this has to be done for every case where there is then a long argument.

```
\cs_new_protected:cpn { __cmd_add_type_+:w }
  {
    \__cmd_flush_m_args:
    \bool_set_true:N \l__cmd_long_bool
    \bool_set_true:N \l__cmd_prefixed_bool
    \__cmd_prepare_signature_bypass:N
  }
```

(*End of definition for* `\__cmd_add_type_+:w`.)

`\__cmd_add_type_!:w` Much the same for controlling trailing optional arguments.

```
\cs_new_protected:cpn { __cmd_add_type_!:w }
  {
    \__cmd_flush_m_args:
    \bool_set_true:N \l__cmd_obey_spaces_bool
    \bool_set_true:N \l__cmd_prefixed_bool
    \__cmd_prepare_signature_bypass:N
  }
```

(*End of definition for* `\__cmd_add_type_!:w`.)

__cmd_add_type_>:w When a processor is found, the processor code is stored. It will be used by __cmd_-args_process: once arguments are all found. Here too the loop calls __cmd_prepare_-signature_bypass:N rather than __cmd_prepare_signature:N so that the flag is not reset.

```
\cs_new_protected:cpn { __cmd_add_type_>:w } #1
  {
    \__cmd_flush_m_args:
    \bool_set_true:N \l__cmd_prefixed_bool
    \bool_set_true:N \l__cmd_process_some_bool
    \tl_put_left:Nn \l__cmd_process_one_tl { {#1} }
    \__cmd_prepare_signature_bypass:N
  }
```

(*End of definition for* __cmd_add_type_>:w.)

__cmd_add_type_=: A mix of the ideas from above: set a flag and add a processor.

```
\cs_new_protected:cpn { __cmd_add_type_=:w } #1
  {
    \__cmd_flush_m_args:
    \bool_set_true:N \l__cmd_prefixed_bool
    \bool_set_true:N \l__cmd_suppress_strip_bool
    \bool_set_true:N \l__cmd_process_some_bool
    \tl_put_left:Nn \l__cmd_process_one_tl
      { { \__cmd_arg_to_keyvalue:nn {#1} } }
    \__cmd_prepare_signature_bypass:N
  }
```

(*End of definition for* __cmd_add_type_=:.)

__cmd_add_type_b:w

```
\cs_new_protected:Npn \__cmd_add_type_b:w
  {
    \__cmd_flush_m_args:
    \__cmd_add_default:
    \__cmd_add_grabber:N b
    \__cmd_prepare_signature:N
  }
```

(*End of definition for* __cmd_add_type_b:w.)

__cmd_add_type_D:w

```
\cs_new_protected:Npn \__cmd_add_type_D:w #1#2#3
  {
    \__cmd_flush_m_args:
    \__cmd_add_default:n {#3}
    \__cmd_add_grabber:N D
    \tl_put_right:Nn \l__cmd_signature_tl { #1 #2 }
    \__cmd_prepare_signature:N
  }
```

(*End of definition for* __cmd_add_type_D:w.)

__cmd_add_type_E:w The E-type argument needs a special handling of default values. Since each embellishment is a separate argument, it also needs to replicate the argument processors for each embellishment argument so that the numbers of arguments and processors remain in sync.

```
\cs_new_protected:Npn \__cmd_add_type_E:w #1#2
  {
    \__cmd_flush_m_args:
    \__cmd_add_default_E:nn {#1} {#2}
    \use:x
      {
        \__cmd_replicate_processor:nn { \tl_count:n {#1} }
          { \exp_not:o \l__cmd_process_one_tl }
      }
    \__cmd_add_grabber:N E
    \tl_put_right:Nn \l__cmd_signature_tl { {#1} }
    \__cmd_prepare_signature:N
  }
```

(*End of definition for* `\__cmd_add_type_E:w`.)

__cmd_replicate_processor:nn In the command's argument processor signature (the final argument of `\__cmd_start:nNNnnn`) there is one braced item for each formal argument (up to nine), and in each of these items there is one braced item for each processor (as many as there were processors declared for a given argument). Something like this:

```
{ % argument processors
  { % argument 1
    { processor 1 } { processor 2 } ... { processor n }
  } % end argument 1
  { ... } % argument 2
      ⋮
  { ... } % argument n
} % end argument processors
```

The function `\__cmd_add_grabber:N` adds one single grabber for an argument, and adds the braced item for that one argument. However, in an E-type argument each embellishment requires its own formal argument, so we need to break out of one layer of braces in `\l__cmd_process_one_tl`, add copies of the processor as necessary, and then return the removed brace. The function below does just that: it defines `\l__cmd_process_one_tl` starting with a $\}_2$ and ending with a $\{_1$, so that it adds as many processors as needed when x-expanded.

```
\cs_new_protected:Npn \__cmd_replicate_processor:nn #1 #2
  {
    \int_compare:nNnF {#1} > { 1 } { \use_none:nnn }
    \tl_set:Nx \l__cmd_process_one_tl
      {
        \exp_not:n { \exp_not:n {#2} \if_false: { \fi: } }
        \prg_replicate:nn { #1 - 2 }
          { \exp_not:n { \exp_not:n { {#2} } } }
        \exp_not:n { { \if_false: } \fi: \exp_not:n {#2} }
```

```
      }
  }
```

(*End of definition for* `\__cmd_replicate_processor:nn`.)

`\__cmd_add_type_m:w` The `m` type is special as short arguments which are not post-processed are simply counted at this stage. Thus there is a check to see if either of these cases apply. If so, a one-argument grabber is added to the signature. On the other hand, if a standard short argument is required it is simply counted at this stage, to be added later using `\__cmd_flush_m_args:`.

```
\cs_new_protected:Npn \__cmd_add_type_m:w
  {
    \__cmd_add_default:
    \bool_if:NTF \l__cmd_prefixed_bool
      { \__cmd_add_grabber:N m }
      { \int_incr:N \l__cmd_m_args_int }
    \__cmd_prepare_signature:N
  }
```

(*End of definition for* `\__cmd_add_type_m:w`.)

`\__cmd_add_type_R:w` The `R`-type argument is very similar to the `D`-type.

```
\cs_new_protected:Npn \__cmd_add_type_R:w #1#2#3
  {
    \__cmd_flush_m_args:
    \__cmd_add_default:n {#3}
    \__cmd_add_grabber:N R
    \tl_put_right:Nn \l__cmd_signature_tl { #1 #2 }
    \__cmd_prepare_signature:N
  }
```

(*End of definition for* `\__cmd_add_type_R:w`.)

`\__cmd_add_type_t:w` Setting up a `t` argument means collecting one token for the test, and adding it along with the grabber to the signature.

```
\cs_new_protected:Npn \__cmd_add_type_t:w #1
  {
    \__cmd_flush_m_args:
    \__cmd_add_default:
    \__cmd_add_grabber:N t
    \tl_put_right:Nn \l__cmd_signature_tl {#1}
    \__cmd_prepare_signature:N
  }
```

(*End of definition for* `\__cmd_add_type_t:w`.)

`\__cmd_add_type_v:w` At this stage, the `v` argument is identical to `l` except that since the grabber may fail to read a verbatim argument we need a default value.

```
\cs_new_protected:Npn \__cmd_add_type_v:w
  {
    \__cmd_flush_m_args:
    \exp_args:No \__cmd_add_default:n \c_novalue_tl
    \__cmd_add_grabber:N v
    \__cmd_prepare_signature:N
  }
```

(*End of definition for* `\__cmd_add_type_v:w`.)

`\__cmd_flush_m_args:` As `m` arguments are simply counted, there is a need to add them to the token register in a block. As this function can only be called if something other than `m` turns up, the flag can be switched here.

```
\cs_new_protected:Npn \__cmd_flush_m_args:
  {
    \int_compare:nNnT \l__cmd_m_args_int > 0
      {
        \tl_put_right:Nx \l__cmd_signature_tl
          { \exp_not:c { __cmd_grab_m_ \int_use:N \l__cmd_m_args_int :w } }
        \tl_put_right:Nx \l__cmd_process_all_tl
          { \prg_replicate:nn { \l__cmd_m_args_int } { { } } }
      }
    \int_zero:N \l__cmd_m_args_int
  }
```

(*End of definition for* `\__cmd_flush_m_args:`.)

`\__cmd_add_grabber:N` To keep the various checks needed in one place, adding the grabber to the signature is done here. The only questions are whether the grabber should be long or not, and whether to obey spaces. The `\l__cmd_obey_spaces_bool` boolean can only be `true` for trailing optional arguments. In that case spaces will not be ignored when looking for that optional argument.

```
\cs_new_protected:Npn \__cmd_add_grabber:N #1
  {
    \tl_put_right:Nx \l__cmd_signature_tl
      {
        \exp_not:c
          {
            __cmd_grab_ #1
            \bool_if:NT \l__cmd_long_bool { _long }
            \bool_if:NT \l__cmd_obey_spaces_bool { _obey_spaces }
            \bool_lazy_and:nnT
              { \l__cmd_suppress_strip_bool }
              { \str_if_eq_p:nn {#1} { D } }
              { _no_strip }
            :w
          }
      }
    \bool_set_false:N \l__cmd_long_bool
    \bool_set_false:N \l__cmd_obey_spaces_bool
    \bool_set_false:N \l__cmd_suppress_strip_bool
    \tl_put_right:Nx \l__cmd_process_all_tl
      {
        {
          \if_charcode:w E #1 \use_i:nn \fi:
          \exp_not:o \l__cmd_process_one_tl
        }
      }
    \tl_clear:N \l__cmd_process_one_tl
  }
```

(*End of definition for* `\__cmd_add_grabber:N`.)

__cmd_add_default:n
__cmd_add_default:
__cmd_add_default_E:nn

Store the default value of an argument, or rather code that gives that default value (it may involve other arguments). This is \c_novalue_tl for arguments with no actual default or with default -NoValue-; and (in a brace group) \prg_do_nothing: followed by a default value for others. For E-type arguments, pad the defaults #2 with some \c_novalue_tl until there are as many as embellishments #1. These functions are also used when defining expandable commands.

```
\cs_new_protected:Npn \__cmd_add_default:n #1
  {
    \tl_if_novalue:nTF {#1}
      { \__cmd_add_default: }
      {
        \int_incr:N \l__cmd_current_arg_int
        \bool_set_true:N \l__cmd_defaults_bool
        \tl_put_right:Nn \l__cmd_defaults_tl { { \prg_do_nothing: #1 } }
      }
  }
\cs_new_protected:Npn \__cmd_add_default:
  {
    \int_incr:N \l__cmd_current_arg_int
    \tl_put_right:Nn \l__cmd_defaults_tl { \c_novalue_tl }
  }
\cs_new_protected:Npn \__cmd_add_default_E:nn #1#2
  {
    \tl_map_function:nN {#2} \__cmd_add_default:n
    \prg_replicate:nn
      { \tl_count:n {#1} - \tl_count:n {#2} }
      { \__cmd_add_default: }
  }
```

(*End of definition for* __cmd_add_default:n, __cmd_add_default:, *and* __cmd_add_default_E:nn.)

1.7 Setting up expandable types

The approach here is not dissimilar to that for standard types, but fewer types are supported. There is also a need to define the per-function auxiliaries: this is done here, while the general grabbers are dealt with later.

__cmd_add_expandable_type_+:w

We have already checked that short arguments are before long arguments, so \l__cmd_long_bool only changes from false to true once (and there is no need to reset it after each argument). Continue the loop.

```
\cs_new_protected:cpn { __cmd_add_expandable_type_+:w }
  {
    \bool_set_true:N \l__cmd_long_bool
    \__cmd_prepare_signature:N
  }
```

(*End of definition for* __cmd_add_expandable_type_+:w.)

__cmd_add_expandable_type_D:w
__cmd_add_expandable_type_D_aux:NNNn
__cmd_add_expandable_type_D_aux:NNN
__cmd_add_expandable_type_D_aux:NN

The set up for D-type arguments involves constructing a rather complex auxiliary which is used repeatedly when grabbing. There is an auxiliary here so that the R-type can share code readily: #1 is D or R. The _aux:NN auxiliary is needed if the two delimiting tokens are identical: in contrast to the non-expandable route, the grabber here has to act differently for this case.

```
\cs_new_protected:Npn \__cmd_add_expandable_type_D:w
  { \__cmd_add_expandable_type_D_aux:NNNn D }
\cs_new_protected:Npn \__cmd_add_expandable_type_D_aux:NNNn #1#2#3#4
  {
    \__cmd_add_default:n {#4}
    \tl_if_eq:nnTF {#2} {#3}
      { \__cmd_add_expandable_type_D_aux:NN #1 #2 }
      { \__cmd_add_expandable_type_D_aux:NNN #1 #2 #3 }
    \__cmd_prepare_signature:N
  }
\cs_new_protected:Npn \__cmd_add_expandable_type_D_aux:NNN #1#2#3
  {
    \bool_if:NTF \l__cmd_long_bool
      { \cs_set:cpx }
      { \cs_set_nopar:cpx }
      { \l__cmd_expandable_aux_name_tl } ##1 ##2 #2 ##3 \q__cmd ##4 #3
      { ##1 {##2} {##3} {##4} }
    \__cmd_add_expandable_grabber:nn {#1}
      {
        \exp_not:c  { \l__cmd_expandable_aux_name_tl }
        \exp_not:n { #2 #3 }
      }
  }
\cs_new_protected:Npn \__cmd_add_expandable_type_D_aux:NN #1#2
  {
    \bool_if:NTF \l__cmd_long_bool
      { \cs_set:cpx }
      { \cs_set_nopar:cpx }
      { \l__cmd_expandable_aux_name_tl } ##1 #2 ##2 #2
      { ##1 {##2} }
    \__cmd_add_expandable_grabber:nn { #1_alt }
      {
        \exp_not:c  { \l__cmd_expandable_aux_name_tl }
        \exp_not:n {#2}
      }
  }
```

(*End of definition for* `\__cmd_add_expandable_type_D:w` *and others.*)

`\__cmd_add_expandable_type_E:w`
`\__cmd_add_expandable_type_E_aux:n`

For each embellishment, use `\__cmd_get_grabber:NN` to obtain an auxiliary delimited by that token and store a pair constituted of the auxiliary and the token in `\l__cmd_tmpb_tl`, before appending the whole set of these pairs to the signature, and an equal number of `-NoValue-` markers (regardless of the default values of arguments). Set the current argument appropriately.

```
\cs_new_protected:Npn \__cmd_add_expandable_type_E:w #1#2
  {
    \__cmd_add_default_E:nn {#1} {#2}
    \tl_clear:N \l__cmd_tmpb_tl
    \tl_map_function:nN {#1} \__cmd_add_expandable_type_E_aux:n
    \__cmd_add_expandable_grabber:nn
      { E \bool_if:NT \l__cmd_long_bool { _long } }
      {
        { \exp_not:o \l__cmd_tmpb_tl }
        {
```

```
          \prg_replicate:nn { \tl_count:n {#1} }
            { { \c_novalue_tl } }
        }
      }
    \__cmd_prepare_signature:N
  }
\cs_new_protected:Npn \__cmd_add_expandable_type_E_aux:n #1
  {
    \__cmd_get_grabber:NN #1 \l__cmd_tmpa_tl
    \tl_put_right:Nx \l__cmd_tmpb_tl
      { \exp_not:o \l__cmd_tmpa_tl \exp_not:N #1 }
  }
```

(*End of definition for* `\__cmd_add_expandable_type_E:w` *and* `\__cmd_add_expandable_type_E_aux:n`.)

`\__cmd_add_expandable_type_m:w` Unlike the standard case, when working expandably each argument is always grabbed separately.

```
\cs_new_protected:Npn \__cmd_add_expandable_type_m:w
  {
    \__cmd_add_default:
    \__cmd_add_expandable_grabber:nn
      { m \bool_if:NT \l__cmd_long_bool { _long } } { }
    \__cmd_prepare_signature:N
  }
```

(*End of definition for* `\__cmd_add_expandable_type_m:w`.)

`\__cmd_add_expandable_type_R:w` The `R`-type is very similar to the `D`-type argument, and so the same internals are used.

```
\cs_new_protected:Npn \__cmd_add_expandable_type_R:w
  { \__cmd_add_expandable_type_D_aux:NNNn R }
```

(*End of definition for* `\__cmd_add_expandable_type_R:w`.)

`\__cmd_add_expandable_type_t:w` An auxiliary delimited by `#1` is built now. It will be used to test for the presence of that token.

```
\cs_new_protected:Npn \__cmd_add_expandable_type_t:w #1
  {
    \__cmd_add_default:
    \__cmd_get_grabber:NN #1 \l__cmd_tmpa_tl
    \__cmd_add_expandable_grabber:nn { t }
      {
        \exp_not:o \l__cmd_tmpa_tl
        \exp_not:N #1
      }
    \__cmd_prepare_signature:N
  }
```

(*End of definition for* `\__cmd_add_expandable_type_t:w`.)

`\__cmd_add_expandable_grabber:nn` This is called for all arguments to place the right grabber in the signature.

```
\cs_new_protected:Npn \__cmd_add_expandable_grabber:nn #1#2
  {
    \tl_put_right:Nx \l__cmd_signature_tl
      { \exp_not:c { __cmd_expandable_grab_ #1 :w } #2 }
  }
```

(*End of definition for* `\__cmd_add_expandable_grabber:nn`.)

`\__cmd_get_grabber:NN`
`\__cmd_get_grabber_auxi:NN`
`\__cmd_get_grabber_auxii:NN`

Given a token `#1`, defines an expandable function delimited by that token and stores it in the token list `#2`. The function is named after the token, unless that function name is already taken by some other grabber (this can happen in the rare case where delimiters with different category codes are used in the same document): in that case use a global counter to get a unique name. Since the grabbers are not named after xparse commands they should not be used to get material from the input stream.

```
\cs_new_protected:Npn \__cmd_get_grabber:NN #1#2
  {
    \cs_set:Npn \__cmd_tmp:w ##1 #1 {##1}
    \exp_args:Nc \__cmd_get_grabber_auxi:NN
      { __cmd_grabber_ \token_to_str:N #1 :w } #2
  }
\cs_new_protected:Npn \__cmd_get_grabber_auxi:NN #1#2
  {
    \cs_if_eq:NNTF \__cmd_tmp:w #1
      { \tl_set:Nn #2 {#1} }
      {
        \cs_if_exist:NTF #1
          {
            \int_gincr:N \g__cmd_grabber_int
            \exp_args:Nc \__cmd_get_grabber_auxi:NN
              {
                __cmd_grabber_
                - \int_use:N \g__cmd_grabber_int :w
              }
              #2
          }
          { \__cmd_get_grabber_auxii:NN #1 #2 }
      }
  }
\cs_new_protected:Npn \__cmd_get_grabber_auxii:NN #1#2
  {
    \cs_set_eq:NN #1 \__cmd_tmp:w
    \tl_set:Nn #2 {#1}
  }
```

(*End of definition for* `\__cmd_get_grabber:NN`, `\__cmd_get_grabber_auxi:NN`, *and* `\__cmd_get_grabber_auxii:NN`.)

1.7.1 Copying a command and its internal structure

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__cmd_copy:NN}%
⟨latexrelease⟩   {Support~\NewCommandCopy~in~ltcmd}
```

Since the 2020-10-01 LaTeX 2_ε release, support for copying, and showing the definition of, robust commands has been available, but the specifics of each command are implemented separately. Here we'll add support for copying and showing ltcmd definitions.

To fully support copying, we need two commands: a conditional to test if a command is in fact a ltcmd command, and another command to actually copy the command. The conditional is defined later as `\__kernel_cmd_if_xparse:NTF`, so now to the copying:

__cmd_copy:NN
__cmd_set_eq_if_exist:NN
__cmd_set_eq_if_exist:cc

This macro just branches to the proper copying command by using __cmd_cmd_type_cases:NnnnnnF. The copying command takes the names of the commands to be copied to and from, and the actual commands as its four arguments.

```
\cs_new_protected:Npn \__cmd_copy:NN #1 #2
  {
    \use:x
      {
        \int_set:Nn \tex_escapechar:D { 92 }
        \exp_not:N \__cmd_cmd_type_cases:NnnnnnF \exp_not:N #2
          { \__cmd_copy_command:nnNN }
          { \__cmd_copy_expandable:nnNN }
          { \__cmd_copy_optimized:nnNN }
          { \__cmd_copy_environment:nnNN }
          { \__cmd_copy_environment_end:nnNN }
          { \__cmd_cant_copy:nwn { non-ltcmd } }
            { \cs_to_str:N #1 } { \cs_to_str:N #2 }
            \exp_not:N #1 \exp_not:N #2
        \exp_not:N \__cmd_break_point:n { \cs_to_str:N #2 }
        \int_set:Nn \tex_escapechar:D { \int_use:N \tex_escapechar:D }
      }
  }
\cs_new_protected:Npn \__cmd_set_eq_if_exist:NN #1 #2
  { \cs_if_exist:NTF #2 { \cs_set_eq:NN } { \use_none:nn } #1 #2 }
\cs_generate_variant:Nn \__cmd_set_eq_if_exist:NN { cc }
```

__cmd_cant_copy:nwn

An utility macro similar to __cmd_bad_def:wn to abort a command copy. Contrary to __cmd_bad_def:wn though, when this happens the issue is most likely internal, because the command was already (supposedly) correcly defined so it should be copyable. Hopefully this macro will never be used ever, but if it does, apologise and give the reason for the failure so the user can report.

```
\cs_new_protected:Npn \__cmd_cant_copy:nwn #1 #2 \__cmd_break_point:n #3
  { \msg_error:nnnn { cmd } { copy-bug } {#1} {#3} }
\msg_new:nnn { cmd } { copy-bug }
  {
    Error~while~copying~command~\iow_char:N\\#2:\\
    \str_case:nn {#1}
      {
        { non-ltcmd } { Command~is~not~a~valid~ltcmd~command. }
        { unknown-type } { Found~an~unknown~argument~type. }
        { invalid-end }
          { Target~command~is~not~named~\iow_char:N \\end<name>. }
      }
  }
```

And, of course, add __kernel_cmd_if_xparse:NTF and __cmd_copy:NN to \@declarecommandcopylisthook:

```
\tl_gput_right:Nn \@declarecommandcopylisthook
  { { \__kernel_cmd_if_xparse:NTF \__cmd_copy:NN } }
```

(*End of definition for* __cmd_copy:NN, __cmd_set_eq_if_exist:NN, *and* __cmd_cant_copy:nwn.)

__cmd_copy_command:nnNN
__cmd_copy_command:NnNNnnnn

A normal (non-expandable) command has a pretty straightforward structure. Its definition is stored in \⟨*cmd*⟩␣code, its defaults (if any) are stored in \⟨*cmd*⟩␣defaults, and its

top-level definition contains its signature, which can just be copied over. `\__cmd_copy_command:nnNN` copies the command code and defaults, and then defines the top-level command using the auxiliary `\__cmd_copy_command:NnNNnnnn`. This macro takes the signature of the command being copied from its top-level definition, and replaces the named bits with the new name.

```
\cs_new_protected:Npn \__cmd_copy_command:nnNN #1 #2 #3 #4
  {
    \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
    \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
    \cs_set_protected_nopar:Npx #3
      { \exp_after:wN \__cmd_copy_command:NnNNnnnn #4 {#1} }
  }
\cs_new:Npn \__cmd_copy_command:NnNNnnnn #1 #2 #3 #4 #5 #6 #7 #8
  {
    #1 \exp_not:n { {#2} }
    \exp_not:c { #8 ~ } \exp_not:c { #8 ~ code }
    \exp_not:n { {#5} {#6} {#7} }
  }
```

(*End of definition for* `\__cmd_copy_command:nnNN` *and* `\__cmd_copy_command:NnNNnnnn`.)

`\__cmd_copy_expandable:nnNN`
`\__cmd_copy_expandable:NnNNNNnnn`

An expandable command is slightly more complicated. Besides the `\`⟨*cmd*⟩`␣code`, and `\`⟨*cmd*⟩`␣defaults`, it also has an auxiliary `\`⟨*cmd*⟩`␣` for grabbing delimited arguments, and possibly another auxiliary `\`⟨*cmd*⟩`␣␣`, if the command has both long and short arguments. Then, its signature also has several specific bits that are unique to that command; this is in contrast to non-expandable commands, which use a common set of parsing functions.

We start by copying the basics, then call `\__cmd_copy_expandable_signature:NnNNNNnnn` to parse the signature of the command and build up the modified copy in a temporary token list, then we call `\__cmd_copy_expandable:NnNNNNnnn` that will copy the top-level definition of the command, with the proper internal renames.

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__cmd_copy:NN}%
⟨latexrelease⟩   {Support~\NewCommandCopy~in~ltcmd}
⟨latexrelease⟩\EndIncludeInRelease
```

There's one variant: a command begins with `\__cmd_start_expandable:nNNNNn` may still be un-expandable/protected if it's defined by `\NewDocumentCommand` and friends, with empty or only m-type arguments.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__cmd_copy_expandable:nnNN}%
⟨latexrelease⟩   {Distinguish~non-expandable~document~commands}
\cs_new_protected:Npn \__cmd_copy_expandable:nnNN #1 #2 #3 #4
  {
    \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
    \__cmd_set_eq_if_exist:cc { #1 ~ } { #2 ~ }
    \__cmd_set_eq_if_exist:cc { #1 ~ \c_space_tl } { #2 ~ \c_space_tl }
    \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
    \exp_after:wN \__cmd_copy_expandable_signature:NnNNNNnnn #4 {#1} {#2}
    \token_if_protected_macro:NTF #4
      { \cs_set_protected_nopar:Npx }{ \cs_set_nopar:Npx }
      #3
      { \exp_after:wN \__cmd_copy_expandable:NnNNNNnnn #4 {#1} {#2} }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__cmd_copy_expandable:nnNN}%
⟨latexrelease⟩  {Support~\NewCommandCopy~in~ltcmd}
⟨latexrelease⟩\cs_new_protected:Npn \__cmd_copy_expandable:nnNN #1 #2 #3 #4
⟨latexrelease⟩  {
⟨latexrelease⟩    \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
⟨latexrelease⟩    \__cmd_set_eq_if_exist:cc { #1 ~ } { #2 ~ }
⟨latexrelease⟩    \__cmd_set_eq_if_exist:cc { #1 ~ \c_space_tl } { #2 ~ \c_space_tl }
⟨latexrelease⟩    \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
⟨latexrelease⟩    \exp_after:wN \__cmd_copy_expandable_signature:NnNNNNnnn #4 {#1} {#2}
⟨latexrelease⟩    \cs_set_nopar:Npx #3
⟨latexrelease⟩      { \exp_after:wN \__cmd_copy_expandable:NnNNNNnnn #4 {#1} {#2} }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__cmd_copy_expandable:nnNN}%
⟨latexrelease⟩  {Support~\NewCommandCopy~in~ltcmd}
⟨latexrelease⟩\EndIncludeInRelease
```

Copy the code, simply define the wrapper.

__cmd_copy_optimized:nnNN

```
\cs_new_protected:Npn \__cmd_copy_optimized:nnNN #1#2#3#4
  {
    \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
    \token_if_protected_macro:NTF #4
      { \cs_set_protected_nopar:Npe }
      { \cs_set_nopar:Npe }
        #3
        {
          \exp_not:N \__cmd_start_optimized:
          \exp_not:c { #1 ~ code }
        }
  }
```

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__cmd_copy:NN (part 2)}%
⟨latexrelease⟩  {Support~\NewCommandCopy~in~ltcmd}
```

```
\cs_new:Npn \__cmd_copy_expandable:NnNNNNnnn #1 #2 #3 #4 #5 #6 #7 #8 #9
  {
    \exp_not:N #1 \exp_not:n { {#2} }
    \exp_not:c { #8 ~ }
    \exp_not:c
      {
        #8 ~
        \str_if_eq:eeT
            { \exp_not:c { #9 ~ \c_space_tl } } { \exp_not:N #4 }
          { \c_space_tl }
      }
    \exp_not:c { #8 ~ code }
    \str_if_eq:eeTF { \exp_not:N #6 } { ? }
      { ? }
      { \exp_not:c { #8 ~ defaults } }
    { \exp_not:V \l__cmd_tmpa_tl }
  }
```

A signature for an expandable command contains as many `\expandable_grab_⟨type⟩:w` as there are arguments, and what follows this macro depends on the ⟨*type*⟩. We'll start a loop through the signature, and at each argument grabber, we'll step the argument

__cmd_copy_expandable_signature:NnNNNNnnn
__cmd_copy_expandable:nnN
__cmd_copy_parse_grabber:w

count, and look for the ⟨*type*⟩ with `\__cmd_copy_parse_grabber:w` so that we know which `\__cmd_copy_grabber_`⟨*type*⟩`:w` to call next.

```
\cs_new_protected:Npn \__cmd_copy_expandable_signature:NnNNNNnnn
    #1 #2 #3 #4 #5 #6 #7 #8 #9
  {
    \int_zero:N \l__cmd_current_arg_int
    \tl_clear:N \l__cmd_tmpa_tl
    \__cmd_copy_expandable:nnN {#8} {#9} #7
      \q_recursion_tail \q_recursion_stop
  }
\cs_new_protected:Npn \__cmd_copy_expandable:nnN #1 #2 #3
  {
    \quark_if_recursion_tail_stop:n {#3}
    \int_incr:N \l__cmd_current_arg_int
    \exp_after:wN \__cmd_copy_parse_grabber:w \token_to_str:N #3 {#1} {#2}
  }
\use:x
  {
    \cs_new_protected:Npn \exp_not:N \__cmd_copy_parse_grabber:w ##1
        \tl_to_str:n { expandable_grab_ } ##2 \tl_to_str:n { :w }
      {
        \tl_put_right:Nx \exp_not:N \l__cmd_tmpa_tl
          { \exp_not:N \exp_not:c { __cmd_expandable_grab_##2:w } }
        \exp_not:N \cs_if_exist_use:cF { __cmd_copy_grabber_##2:w }
          { \__cmd_cant_copy:nwn { unknown-type } }
      }
  }
```

The most complicated is the `Delimited` argument: each argument has a dedicated grabbing function named after the command that has to be copied over (of the form `\`⟨*cmd*⟩`␣(arg␣`⟨*num*⟩`)`).

```
\__cmd_copy_grabber_D:w
\__cmd_copy_grabber_D_alt:w
\__cmd_copy_grabber_R:w
\__cmd_copy_grabber_R_alt:w
\__cmd_copy_grabber_E:w
\__cmd_copy_grabber_E_long:w
\__cmd_copy_grabber_t:w
\__cmd_copy_grabber_m:w
\__cmd_copy_grabber_m_long:w
```

```
\cs_new_protected:Npn \__cmd_copy_grabber_D:w #1 #2 #3 #4 #5
  {
    \tl_put_right:Nx \l__cmd_tmpa_tl
      {
        \exp_not:c { #1 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
        \exp_not:n { #4 #5 }
      }
    \cs_set_eq:cc
      { #1 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
      { #2 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
    \__cmd_copy_expandable:nnN {#1} {#2}
  }
```

`D_alt` is just a special case of `D` that uses a single delimiter (used when both delimiters of the argument are identical):

```
\cs_new_protected:Npn \__cmd_copy_grabber_D_alt:w #1 #2 #3 #4
  { \__cmd_copy_grabber_D:w {#1} {#2} {#3} {#4} { } }
```

As far as copying is concerned, `R` is identical to `D`:

```
\cs_new_eq:NN \__cmd_copy_grabber_R:w \__cmd_copy_grabber_D:w
\cs_new_eq:NN \__cmd_copy_grabber_R_alt:w \__cmd_copy_grabber_D_alt:w
```

E is straightforward: we just copy the embellishments over, and increase the current argument number \l__cmd_current_arg_int by the number of embellishments (minus one because there is a \int_incr:N down the line).

```
\cs_new_protected:Npn \__cmd_copy_grabber_E:w #1 #2 #3 #4
  {
    \tl_put_right:Nn \l__cmd_tmpa_tl { {#3} {#4} }
    \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#4} - 1 }
    \__cmd_copy_expandable:nnN {#1} {#2}
  }
\cs_new_eq:NN \__cmd_copy_grabber_E_long:w \__cmd_copy_grabber_E:w
```

t just needs copying the token to be tested for:

```
\cs_new_protected:Npn \__cmd_copy_grabber_t:w #1 #2 #3 #4
  {
    \tl_put_right:Nn \l__cmd_tmpa_tl { #3 #4 }
    \__cmd_copy_expandable:nnN {#1} {#2}
  }
```

And last but not least, m is the simplest; the grabber is just __cmd_expandable_grab_m:w, which is already added to the new command so here we just resume the loop:

```
\cs_new_protected:Npn \__cmd_copy_grabber_m:w { \__cmd_copy_expandable:nnN }
\cs_new_eq:NN \__cmd_copy_grabber_m_long:w \__cmd_copy_grabber_m:w
```

(*End of definition for* __cmd_copy_expandable:nnNN *and others.*)

__cmd_copy_environment:nnNN
__cmd_copy_environment:Nnnnnnn

Copying an environment's \begin part is pretty much like copying a command, except it has a longer name, and at the end we have to copy \environment ⟨*name*⟩ into \⟨*name*⟩.

```
\cs_new_protected:Npn \__cmd_copy_environment:nnNN #1 #2 #3 #4
  {
    \cs_set_eq:cc { environment~ #1 ~ code } { environment~ #2 ~ code }
    \__cmd_set_eq_if_exist:cc
      { environment~ #1 ~ defaults } { environment~ #2 ~ defaults }
    \cs_set_protected_nopar:cpx { environment~ #1 }
      { \exp_after:wN \__cmd_copy_environment:Nnnnnnn #4 {#1} }
    \cs_set_eq:cc {#1} { environment~ #1 }
  }
\cs_new:Npn \__cmd_copy_environment:Nnnnnnn #1 #2 #3 #4 #5 #6 #7
  { #1 \exp_not:n { {#2} } {#7} \exp_not:n { {#4} {#5} {#6} } }
```

(*End of definition for* __cmd_copy_environment:nnNN *and* __cmd_copy_environment:Nnnnnnn.)

__cmd_copy_environment_end:nnNN
__cmd_copy_environment_end_aux:nnNN

Copying an environment's \end part is a bit trickier. We first have to make sure that both parts are named \end⟨*name*⟩ (that's actually not a hard requirement, but an environment \end command makes no sense without the end in its name), and strip the leading end from the strings. After that, copying is straightforward.

```
\cs_new_protected:Npn \__cmd_copy_environment_end:nnNN #1 #2
  {
    \__cmd_check_end:Nn \l__cmd_tmpa_tl {#1}
    \__cmd_check_end:Nn \l__cmd_tmpb_tl {#2}
    \exp_args:Noo \__cmd_copy_environment_end_aux:nnNN
      { \l__cmd_tmpa_tl } { \l__cmd_tmpb_tl }
  }
\cs_new_protected:Npn \__cmd_copy_environment_end_aux:nnNN #1 #2 #3 #4
  {
```

```
    \cs_set_nopar:cpx { environment~ #1 ~end }
      { \exp_not:c { environment~ #1 ~end~aux } }
    \cs_set_eq:cc
      { environment~ #1 ~end~aux~ } { environment~ #2 ~end~aux~ }
    \cs_set_eq:cc { end #1 } { environment~ #1 ~end }
  }
```

To check whether an `\end` command is valid, we look for the string `end` at the beginning of the command name, and if not found, raise an error:

__cmd_check_end:Nn
__cmd_check_end:n
__cmd_check_end:w

```
\cs_new_protected:Npn \__cmd_check_end:Nn #1 #2
  {
    \tl_set:Nx #1 { \__cmd_check_end:n {#2} }
    \token_if_eq_meaning:NNT #1 \q_nil
      { \__cmd_cant_copy:nwn { invalid-end } }
  }
\cs_set_protected:Npn \__cmd_tmp:w #1
  {
    \cs_new:Npn \__cmd_check_end:n ##1
      {
        \exp_after:wN \__cmd_check_end:w \tl_to_str:n {##1}
          #1 \q_mark #1 \q_stop
      }
    \cs_new:Npn \__cmd_check_end:w ##1 #1 ##2 #1 ##3 \q_stop
      { \if_meaning:w ##2 \q_mark \exp_not:N \q_nil \else: ##2 \fi: }
  }
\exp_args:No \__cmd_tmp:w { \tl_to_str:n { end } }
```

(*End of definition for* `\__cmd_copy_environment_end:nnNN` *and others.*)

Not much to do regarding latexrelease: we could remove the entries from `\@declarecommandcopylist` but it doesn't seem worth it.

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__cmd_copy:NN (part 2)}%
⟨latexrelease⟩   {Support~\NewCommandCopy~in~ltcmd}
⟨latexrelease⟩\EndIncludeInRelease
```

1.7.2 Showing the definition of a command

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__cmd_show:N}%
⟨latexrelease⟩   {Support~\ShowCommand~in~ltcmd}
```

To show the definition of a command we need more or less the same building blocks as for copying, except that instead of making a copy, we'll just print stuff to the terminal.

__cmd_show:N

This macro just branches to the proper showing command by using `\__cmd_cmd_type_cases:NnnnnnF`. The showing command takes the command to be shown as argument.

```
\cs_new_protected:Npn \__cmd_show:N #1
  {
    \use:x
      {
        \int_set:Nn \tex_escapechar:D { 92 }
        \exp_not:N \__cmd_cmd_type_cases:NnnnnnF \exp_not:N #1
          { \__cmd_show_command:N }
          { \__cmd_show_expandable:N }
          { \__cmd_show_optimized:N }
          { \__cmd_show_environment:N }
```

```
          { \__cmd_show_environment_end:N }
          { \__cmd_cant_copy:nwn { non-ltcmd } }
            \exp_not:N #1
        \exp_not:N \__cmd_break_point:n { \cs_to_str:N #1 }
        \int_set:Nn \tex_escapechar:D { \int_use:N \tex_escapechar:D }
      }
  }
```

(*End of definition for* `\__cmd_show:N`.)

`\__cmd_show_command:N`
`\__cmd_show_command:NnNNwN`
`\__cmd_show_expandable:N`
`\__cmd_show_expandable:NnNNNNnN`
`\__cmd_show_optimized:N`
`\__cmd_show_command_aux:NnNNn`
`\__cmd_show_environment:N`
`\__cmd_show:x`

These commands just expand the command once to reveal its innards, then pass the type of command, the control sequence, the signature, and the code macro to `\__cmd_show_command_aux:NnNNn`.

```
\cs_new_protected:Npn \__cmd_show_command:N #1
  { \exp_after:wN \__cmd_show_command:NnNNwN #1 \q__cmd #1 }
\cs_new_protected:Npn \__cmd_show_command:NnNNwN #1 #2 #3 #4 #5 \q__cmd #6
  {
    \__cmd_show_command_aux:NnNNn \tl_show:x
      { document~command } #6 #4 {#2}
  }
\cs_new_protected:Npn \__cmd_show_expandable:N #1
  { \exp_after:wN \__cmd_show_expandable:NnNNNNnN #1 #1 }
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__cmd_show:N}%
⟨latexrelease⟩   {Support~\ShowCommand~in~ltcmd}
⟨latexrelease⟩\EndIncludeInRelease
```

There's one variant: a command begins with `\__cmd_start_expandable:nNNNNn` may still be un-expandable/protected if it's defined by `\NewDocumentCommand` and friends, with empty or only m-type arguments.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__cmd_show_expandable:NnNNNNnN}%
⟨latexrelease⟩   {Distinguish~non-expandable~document~commands}
\cs_new_protected:Npn \__cmd_show_expandable:NnNNNNnN #1 #2 #3 #4 #5 #6 #7 #8
  {
    \exp_args:NNe \__cmd_show_command_aux:NnNNn \tl_show:x
      { \token_if_protected_macro:NF #8 { expandable~ } document~command }
      #8 #5 {#2}
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__cmd_show_expandable:NnNNNNnN}%
⟨latexrelease⟩   {Support~\ShowCommand~in~ltcmd}
⟨latexrelease⟩\cs_new_protected:Npn \__cmd_show_expandable:NnNNNNnN #1 #2 #3 #4 #5 #6 #7 #8
⟨latexrelease⟩   {
⟨latexrelease⟩     \__cmd_show_command_aux:NnNNn \tl_show:x
⟨latexrelease⟩       { expandable~document~command } #8 #5 {#2}
⟨latexrelease⟩   }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__cmd_show_expandable:NnNNNNnN}%
⟨latexrelease⟩   {Support~\ShowCommand~in~ltcmd}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__cmd_show:N (part 2)}%
⟨latexrelease⟩   {Support~\ShowCommand~in~ltcmd}
```

Now just print everything in the required format. The auxiliary `\__cmd_split_signature:n` stores a ready-to-print token list in `\l__cmd_tmpa_tl`, so we ust use that here:

```
\cs_new_protected:Npn \__cmd_show_command_aux:NnNNn #1 #2 #3 #4 #5
  {
    \__cmd_split_signature:n {#5}
    #1
      {
        \token_to_str:N #3 = #2: \iow_newline:
        \tl_use:N \l__cmd_tmpa_tl
        -> \cs_replacement_spec:N #4
      }
  }
```

Optimized functions need things done a bit differently as we need to reconstruct the argument spec.

```
\cs_new_protected:Npn \__cmd_show_optimized:N #1
  {
    \exp_args:Nc \__cmd_show_optimized:NN
      { \cs_to_str:N #1 \c_space_tl code }
      #1
  }
\cs_new_protected:Npn \__cmd_show_optimized:NN #1#2
  {
    \cs_set:Npe \__cmd_show_optimized_aux:N ##1
      {
        \c_space_tl \c_space_tl \c_hash_str ##1 :
        \bool_lazy_or:nnT
          { \token_if_long_macro_p:N #1 }
          { \token_if_protected_long_macro_p:N #1 }
            { + } m
         \iow_newline:
      }
    \tl_show:e
      {
        \token_to_str:N #2 =
          \bool_lazy_or:nnF
            { \token_if_protected_macro_p:N #1 }
            { \token_if_protected_long_macro_p:N #1 }
              { expandable ~ } document~command:
        \iow_newline:
        \int_step_function:nN
          {
            \int_div_truncate:nn
              { \tl_count:e { \cs_parameter_spec:N #1 } }
              { 2 }
          }
          \__cmd_show_optimized_aux:N
        ->
        \cs_replacement_spec:N #1
      }
  }
\cs_generate_variant:Nn \tl_count:n { e }
```

We can reuse most of the above to show an environment, except that we need to ensure that the proper `\environment ...` are passed to `\__cmd_show_command_aux:NnNNn`. Additionally, when `\ShowCommand\foo` is used (if `foo` is an environment), we show `\endfoo` as well, and when `\ShowCommand\endfoo` is used, change that to `\ShowCommand\foo` and do the same.

```
\cs_new_protected:Npn \__cmd_show_environment:N #1
  {
    \exp_after:wN \__cmd_show_environment:Nnnw #1 \q__cmd
    \tl_show:x
      {
        \token_to_str:N \end { \cs_to_str:N #1 } : \iow_newline:
        -> \exp_args:Nc \cs_replacement_spec:N
          { environment~ \cs_to_str:N #1 ~end~aux~ }
      }
  }
\cs_new_protected:Npn \__cmd_show_environment:Nnnw #1 #2 #3 #4 \q__cmd
  {
    \use:x
      {
        \__cmd_show_command_aux:NnNNn \__cmd_show:x { document~environment }
          { \exp_not:N \begin {#3} }
          \exp_not:c { environment~ #3 ~ code }
          {#2}
      }
  }
\cs_new_protected:Npn \__cmd_show:x #1
  { \iow_term:x { > ~ #1 . \iow_newline: } }
\cs_new_protected:Npn \__cmd_show_environment_end:N #1
  {
    \exp_args:NNx \__cmd_check_end:Nn \l__cmd_tmpa_tl { \cs_to_str:N #1 }
    \exp_args:Nc \__cmd_show_environment:N { \l__cmd_tmpa_tl }
  }
```

And, of course, add `\__kernel_cmd_if_xparse:NTF` and `\__cmd_show:N` to `\@showcommandlisthoo` and to `\@showenvironmentlisthook` (`\__cmd_show:N` takes care of the environment case as well, so both entries are identical):

```
\tl_gput_right:Nn \@showcommandlisthook
  { { \__kernel_cmd_if_xparse:NTF \__cmd_show:N } }
\tl_gput_right:Nn \@showenvironmentlisthook
  { { \__kernel_cmd_if_xparse:NTF \__cmd_show:N } }
```

(*End of definition for* `\__cmd_show_command:N` *and others.*)

`\__cmd_split_signature:n` Now we'll try a least-effort adventure into splitting the symbolic user-provided signature for a command into individual parameters for pretty-printing. A counter is used to keep track of the current argument number, and two token lists are used: `\l__cmd_tmpa_tl` holds the final token list to be printed, and `\l__cmd_tmpb_tl` holds just the current item, so that we can make changes to an individual item without having to dissect the whole thing (this is used for `e`- and `E`-types).

```
\cs_new_protected:Npn \__cmd_split_signature:n #1
  {
    \int_set:Nn \l__cmd_current_arg_int { 1 }
    \tl_clear:N \l__cmd_tmpa_tl
```

```
    \tl_clear:N \l__cmd_tmpb_tl
    \__cmd_split_signature_loop:Nw #1 \q_recursion_tail \q_recursion_stop
  }
```

This is the main chunk of the loop: it starts an item with `\__cmd_split_start_item:` (this adds indentation and the argument number to `\l__cmd_tmpb_tl`), then checks if a special token list `\c__cmd_show_type_`⟨*type*⟩`_tl` exists. If it doesn't, the current argument is a "simple" type which needs no extra processing. Otherwise, call a specific function depending on the value of said token list.

__cmd_split_signature_loop:Nw

```
\cs_new_protected:Npn \__cmd_split_signature_loop:Nw #1
  {
    \quark_if_recursion_tail_stop:N #1
    \tl_if_empty:NT \l__cmd_tmpb_tl { \__cmd_split_start_item: }
    \tl_if_exist:cTF { c__cmd_show_type_#1_tl }
      {
        \use:c
          {
            __cmd_show_
            \if_case:w \tl_use:c { c__cmd_show_type_#1_tl } \exp_stop_f:
            delim \or: delims \or: delims_opt \or: opt \or:
            e \or: E \or: prefix \or: processor \fi: :Nw
          } #1
      }
      { \__cmd_split_end_item:n {#1} \__cmd_split_signature_loop:Nw }
  }
```

The token lists `\c__cmd_show_type_`⟨*type*⟩`_tl` exist for nontrivial (for printing) ⟨*types*⟩ that require special parsing (like delimiters or optional arguments). Values from 0 to 7 are assigned to each type:

\c__cmd_show_type_t_tl
\c__cmd_show_type_r_tl
\c__cmd_show_type_d_tl
\c__cmd_show_type_R_tl
\c__cmd_show_type_D_tl
\c__cmd_show_type_O_tl
\c__cmd_show_type_e_tl
\c__cmd_show_type_E_tl
\c__cmd_show_type_+_tl
\c__cmd_show_type_!_tl
\c__cmd_show_type_>_tl

1. a single delimiter token;
2. two delimiter tokens;
3. two delimiter tokens plus a default value;
4. a default value;
5. a list of embellishments (exclusive for `e`-type);
6. embellishments plus defaults (exclusive for `E`-type);
7. simple prefixes;
8. prefixes with arguments (argument processors);

```
\cs_set_protected:Npn \__cmd_tmp:w #1 #2
  {
    \quark_if_nil:nF {#1}
      { \tl_const:cn { c__cmd_show_type_#1_tl } {#2} \__cmd_tmp:w }
  }
\__cmd_tmp:w t0 r1 d1 R2 D2 O3 e4 E5 +6 !6 >7 =7 \q_nil \q_nil
```

Now, based on each type we know how to act. In most cases it is just a matter of feeding in the grabbed arguments and resuming the loop. The embellishments require a bit more attention: the e-type loops through the list of embellishments and adds each to the token list as a separate argument. The E-type does more or less the same, but uses __cmd_tl_mapthread_function:nnN to map over two lists simultaneously, adding each token and default to the token list for printing.

```
\cs_new_protected:Npn \__cmd_show_delim:Nw #1 #2
  { \__cmd_split_end_item:n { #1 #2 } \__cmd_split_signature_loop:Nw }
\cs_new_protected:Npn \__cmd_show_delims:Nw #1 #2 #3
  { \__cmd_split_end_item:n { #1 #2 #3 } \__cmd_split_signature_loop:Nw }
\cs_new_protected:Npn \__cmd_show_delims_opt:Nw #1 #2 #3 #4
  { \__cmd_split_end_item:n { #1 #2 #3 {#4} } \__cmd_split_signature_loop:Nw }
\cs_new_protected:Npn \__cmd_show_opt:Nw #1 #2
  { \__cmd_split_end_item:n { #1 {#2} } \__cmd_split_signature_loop:Nw }
\cs_new_protected:Npn \__cmd_show_e:Nw #1 #2
  {
    \tl_map_inline:nn {#2}
      {
        \__cmd_split_start_item:
        \__cmd_split_end_item:n { #1 ##1 }
      }
    \__cmd_split_signature_loop:Nw
  }
\cs_set_protected:Npn \__cmd_tmp:w #1
  {
    \cs_new_protected:Npn \__cmd_show_E:Nw ##1 ##2 ##3
      {
        \cs_set_protected:Npn \__cmd_tmp:w ####1 ####2
          {
            \__cmd_split_start_item:
            \__cmd_split_end_item:n { ##1 ####1 {####2} }
          }
        \__cmd_tl_mapthread_function:nnN {##2}
          { ##3 {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} } \__cmd_tmp:w
        \__cmd_split_signature_loop:Nw
      }
  }
\exp_args:NV \__cmd_tmp:w \c_novalue_tl
```

__cmd_show_delim:Nw
__cmd_show_delims:Nw
__cmd_show_delims_opt:Nw
__cmd_show_opt:Nw
__cmd_show_e:Nw
__cmd_show_E:Nw
__cmd_show_prefix:Nw
__cmd_show_processor:Nw

Minor wrinkle with the prefixes: they use __cmd_split_add_item:n instead of __cmd_split_end_item:n (add *vs.* end) because they are followed by an argument, so they can't end the item.

```
\cs_new_protected:Npn \__cmd_show_prefix:Nw #1
  { \__cmd_split_add_item:n {#1} \__cmd_split_signature_loop:Nw }
\cs_new_protected:Npn \__cmd_show_processor:Nw #1 #2
  { \__cmd_split_add_item:n { #1 {#2} } \__cmd_split_signature_loop:Nw }
```

And now the auxiliaries that store the strings to be printed. __cmd_split_start_item: starts an item from scratch, __cmd_split_add_item:n adds tokens to an item without adding a newline, and __cmd_split_end_item:n adds tokens, terminates the item with a newline, and steps the argument count.

```
\cs_new_protected:Npn \__cmd_split_start_item:
  {
```

```
    \tl_set:Nx \l__cmd_tmpb_tl
      { ~ \c_space_tl \c_hash_str \int_use:N \l__cmd_current_arg_int : }
  }
\cs_new_protected:Npn \__cmd_split_add_item:n #1
  { \tl_put_right:Nx \l__cmd_tmpb_tl { \tl_to_str:n {#1} } }
\cs_new_protected:Npn \__cmd_split_end_item:n #1
  {
    \tl_put_right:Nx \l__cmd_tmpa_tl
      { \l__cmd_tmpb_tl \tl_to_str:n {#1} \iow_newline: }
    \tl_clear:N \l__cmd_tmpb_tl
    \int_incr:N \l__cmd_current_arg_int
  }
```

(*End of definition for* `\__cmd_split_signature:n` *and others.*)

Not much to do regarding latexrelease: we could remove the entries from `\@showcommandlisthook`, but it doesn't seem worth it.

```
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__cmd_show:N (part 2)}%
⟨latexrelease⟩  {Support~\ShowCommand~in~ltcmd}
⟨latexrelease⟩\EndIncludeInRelease
```

1.8 Grabbing arguments

All of the grabbers follow the same basic pattern. The initial function stores in `\l__cmd_signature_tl` the code to grab further arguments, defines (the function in) `\l__cmd_fn_tl` that will grab the argument, and calls it.

Defining `\l__cmd_fn_tl` means determining whether to use `\cs_set:Npn` or `\cs_set_nopar:Npn`, and for optional arguments whether to skip spaces. Once the argument is found, `\l__cmd_fn_tl` calls `\__cmd_add_arg:n`, responsible for calling processors and grabbing further arguments.

`\__cmd_grab_b:w`
`\__cmd_grab_b_long:w`
`\__cmd_grab_b_obey_spaces:w`
`\__cmd_grab_b_long_obey_spaces:w`
`\__cmd_grab_b_aux:NNw`
`\__cmd_grab_b_end:Nw`

This uses the well-tested code of `D`-type arguments, skipping the peeking step because the `b`-type argument is always present, and adding a cleanup stage at the end by hijacking the signature. The clean-up consists of properly dealing with `\l__cmd_args_tl` and also putting back the `\end` that served as an end-delimiter: this `\end` receives the environment name as its argument and is run normally. The `D`-type code stores the argument found (body of the environment) as a brace group in `\l__cmd_args_tl` and depending on the presence of a prefix `!` we trim spaces or not before adding this braced argument into the saved `\l__cmd_args_tl`. The strange `\begin␣` control sequence is there for display purposes only: it has to look like `\begin` in the terminal but not to delimited arguments.

```
\cs_new_protected:Npn \__cmd_grab_b:w
  { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \tl_trim_spaces:n }
\cs_new_protected:Npn \__cmd_grab_b_long:w
  { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \tl_trim_spaces:n }
\cs_new_protected:Npn \__cmd_grab_b_obey_spaces:w
  { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \exp_not:n }
\cs_new_protected:Npn \__cmd_grab_b_long_obey_spaces:w
  { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \exp_not:n }
\cs_new_protected:Npn \__cmd_grab_b_aux:NNw #1#2#3 \__cmd_run_code:
  {
    \__cmd_grab_D_aux:NNnNN \begin \end {#3} #1 \use_ii:nn
```

```
    \tl_put_left:Nn \l__cmd_signature_tl { \__cmd_grab_b_end:Nw #2 }
    \tl_set_eq:NN \l__cmd_saved_args_tl \l__cmd_args_tl
    \tl_clear:N \l__cmd_args_tl
    \exp_args:Nc \l__cmd_fn_tl { begin ~ }
  }
\cs_new_protected:Npn \__cmd_grab_b_end:Nw #1#2 \__cmd_run_code:
  {
    \tl_set:Nx \l__cmd_args_tl
      {
        \exp_not:V \l__cmd_saved_args_tl
        { \exp_after:wN #1 \l__cmd_args_tl }
      }
    #2
    \__cmd_run_code:
    \end
  }
```

(*End of definition for* `\__cmd_grab_b:w` *and others.*)

`\__cmd_grab_D:w`
`\__cmd_grab_D_long:w`
`\__cmd_grab_D_obey_spaces:w`
`\__cmd_grab_D_long_obey_spaces:w`
`\__cmd_grab_D_no_strip:w`
`\__cmd_grab_D_long_no_strip:w`
`\__cmd_grab_D_obey_spaces_no_strip:w`
`\__cmd_grab_D_long_obey_spaces_no_strip:w`

The generic delimited argument grabber. The auxiliary function does a peek test before calling `\__cmd_grab_D_call:Nw`, so that the optional nature of the argument works as expected.

```
\cs_new_protected:Npn \__cmd_grab_D:w #1#2#3 \__cmd_run_code:
  {
    \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected_nopar:Npn
      \__cmd_peek_nonspace_remove:NTF \use_ii:nn
  }
\cs_new_protected:Npn \__cmd_grab_D_long:w #1#2#3 \__cmd_run_code:
  {
    \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected:Npn
      \__cmd_peek_nonspace_remove:NTF \use_ii:nn
  }
\cs_new_protected:Npn \__cmd_grab_D_obey_spaces:w #1#2#3 \__cmd_run_code:
  {
    \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected_nopar:Npn
      \__cmd_peek_meaning_remove:NTF \use_ii:nn
  }
\cs_new_protected:Npn \__cmd_grab_D_long_obey_spaces:w #1#2#3 \__cmd_run_code:
  {
    \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected:Npn
      \__cmd_peek_meaning_remove:NTF \use_ii:nn
  }
\cs_new_protected:Npn \__cmd_grab_D_no_strip:w
  #1#2#3 \__cmd_run_code:
  {
    \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected_nopar:Npn
      \__cmd_peek_nonspace_remove:NTF \use_none:n
  }
\cs_new_protected:Npn \__cmd_grab_D_long_no_strip:w
  #1#2#3 \__cmd_run_code:
  {
    \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected:Npn
      \__cmd_peek_nonspace_remove:NTF \use_none:n
  }
```

```
\cs_new_protected:Npn \__cmd_grab_D_obey_spaces_no_strip:w
  #1#2#3 \__cmd_run_code:
  {
    \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected_nopar:Npn
      \__cmd_peek_meaning_remove:NTF \use_none:n
  }
\cs_new_protected:Npn \__cmd_grab_D_long_obey_spaces_no_strip:w
  #1#2#3 \__cmd_run_code:
  {
    \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected:Npn
      \__cmd_peek_meaning_remove:NTF \use_none:n
  }
```

This is a bit complicated. The idea is that, in order to check for nested optional argument tokens (`[[...]]` and so on) the argument needs to be grabbed without removing any braces at all. If this is not done, then cases like `[{[}]` fail. So after testing for an optional argument, it is collected piece-wise. Inserting a quark prevents loss of braces, and there is then a test to see if there are nested delimiters to handle.

```
\cs_new_protected:Npn \__cmd_grab_D_aux:NNnNNN #1#2#3#4#5#6
  {
    \__cmd_grab_D_aux:NNnNN #1#2 {#3} #4 #6
    #5 #1
      { \__cmd_grab_D_call:Nw #1 }
      { \__cmd_add_arg:o \c_novalue_tl }
  }
```

`\__cmd_grab_D_aux:NNnNNN`
`\__cmd_grab_D_aux:NNnNN`

Inside the "standard" grabber, there is a test to see if the grabbed argument is entirely enclosed by braces. There are a couple of extra factors to allow for: the argument might be entirely empty, and spaces at the start and end of the input must be retained around a brace group. Also notice that a *blank* argument might still contain spaces. To allow for suppression of brace stripping, the business end is passed here as `#5`.

```
\cs_new_protected:Npn \__cmd_grab_D_aux:NNnNN #1#2#3#4#5
  {
    \tl_set:Nn \l__cmd_signature_tl {#3}
    \exp_after:wN #4 \l__cmd_fn_tl ##1 #2
      {
        \tl_if_in:nnTF {##1} {#1}
          { \__cmd_grab_D_nested:NNnN #1 #2 {##1} #4 }
          {
            \tl_if_blank:oTF { \use_none:n ##1 }
              { \__cmd_add_arg:o { \use_none:n ##1 } }
              {
                \str_if_eq:eeTF
                  { \exp_not:o { \use_none:n ##1 } }
                  { { \exp_not:o { \use_ii:nnn ##1 \q_nil } } }
                  { \__cmd_add_arg:o { #5 ##1 } }
                  { \__cmd_add_arg:o { \use_none:n ##1 } }
              }
          }
      }
  }
```

(*End of definition for* `\__cmd_grab_D:w` *and others.*)

__cmd_grab_D_nested:NNnN
__cmd_grab_D_nested:w
\l__cmd_nesting_a_tl
\l__cmd_nesting_b_tl
\q__cmd

Catching nested optional arguments means more work. The aim here is to collect up each pair of optional tokens without TEX helping out, and without counting anything. The code above will already have removed the leading opening token and a closing token, but the wrong one. The aim is then to work through the material grabbed so far and divide it up on each opening token, grabbing a closing token to match (thus working in pairs). Once there are no opening tokens, then there is a second check to see if there are any opening tokens in the second part of the argument (for things like `[][]`). Once everything has been found, the entire collected material is added to the output as a single argument. The only tricky part here is ensuring that any grabbing function that might run away is named after the function currently being parsed and not after xparse. That leads to some rather complex nesting! There is also a need to prevent the loss of any braces, hence the insertion and removal of quarks along the way.

```
\tl_new:N \l__cmd_nesting_a_tl
\tl_new:N \l__cmd_nesting_b_tl
\quark_new:N \q__cmd
\cs_new_protected:Npn \__cmd_grab_D_nested:NNnN #1#2#3#4
  {
    \tl_clear:N \l__cmd_nesting_a_tl
    \tl_clear:N \l__cmd_nesting_b_tl
    \exp_after:wN #4 \l__cmd_fn_tl ##1 #1 ##2 \q__cmd ##3 #2
      {
        \tl_put_right:No \l__cmd_nesting_a_tl { \use_none:n ##1 #1 }
        \tl_put_right:No \l__cmd_nesting_b_tl { \use_i:nn #2 ##3 }
        \tl_if_in:nnTF {##2} {#1}
          {
            \l__cmd_fn_tl
              \q_nil ##2 \q__cmd \ERROR
          }
          {
            \tl_put_right:Nx \l__cmd_nesting_a_tl
              { \__cmd_grab_D_nested:w \q_nil ##2 \q_stop }
            \tl_if_in:NnTF \l__cmd_nesting_b_tl {#1}
              {
                \tl_set_eq:NN \l__cmd_tmpa_tl \l__cmd_nesting_b_tl
                \tl_clear:N \l__cmd_nesting_b_tl
                \exp_after:wN \l__cmd_fn_tl \exp_after:wN
                  \q_nil \l__cmd_tmpa_tl \q_nil \q__cmd \ERROR
              }
              {
                \tl_put_right:No \l__cmd_nesting_a_tl
                  \l__cmd_nesting_b_tl
                \__cmd_add_arg:V \l__cmd_nesting_a_tl
              }
          }
      }
    \l__cmd_fn_tl #3 \q_nil \q__cmd \ERROR
  }
\cs_new:Npn \__cmd_grab_D_nested:w #1 \q_nil \q_stop
  { \exp_not:o { \use_none:n #1 } }
```

(*End of definition for* `\__cmd_grab_D_nested:NNnN` *and others.*)

__cmd_grab_D_call:Nw

For D and R-type arguments, to avoid losing any braces, a token needs to be inserted before the argument to be grabbed. If the argument runs away because the closing token

is missing then this inserted token shows up in the terminal. Ideally, #1 would therefore be used directly, but that is no good as it will mess up the rest of the grabber. Instead, a copy of #1 with an altered category code is used, as this will look right in the terminal but will not mess up the grabber. The only issue then is that the category code of #1 is unknown. So there is a quick test to ensure that the inserted token can never be matched by the grabber. (This assumes that the open and close delimiters are not the same character with different category codes, but that really should not happen in any sensible document-level syntax.) An exception is when #1 is a control sequence token, in which case the character-token treatment is no good because if hit with \token_to_str:N it would add sputios tokens to the argument. In this case a different branch is taken. The token inserted is then the same ⟨*csname*⟩ as #1, but with a space appended, so that the grabber don't see it as another of the same delimiter.

```
\cs_new_protected_nopar:Npn \__cmd_grab_D_call:Nw #1
  {
    \token_if_eq_catcode:NNTF + #1
      {
        \exp_after:wN \exp_after:wN \exp_after:wN
          \l__cmd_fn_tl \char_generate:nn { `#1 } { 11 }
      }
      {
        \__cmd_token_if_cs:NTF #1
          {
            \exp_after:wN \l__cmd_fn_tl
            \cs:w \cs_to_str:N #1 ~ \cs_end:
          }
          {
            \exp_after:wN \l__cmd_fn_tl
            \token_to_str:N #1
          }
      }
  }
```

(*End of definition for* __cmd_grab_D_call:Nw.)

__cmd_grab_E:w
__cmd_grab_E_long:w
__cmd_grab_E_obey_spaces:w
__cmd_grab_E_long_obey_spaces:w
__cmd_grab_E:nnNN
__cmd_grab_E_loop:NnN
__cmd_grab_E_finalise:

Everything here needs to point to a loop.

```
\cs_new_protected:Npn \__cmd_grab_E:w #1#2 \__cmd_run_code:
  {
    \__cmd_grab_E:nnNN {#1} {#2}
      \cs_set_protected_nopar:Npn
      \__cmd_peek_nonspace_remove:NTF
  }
\cs_new_protected:Npn \__cmd_grab_E_long:w #1#2 \__cmd_run_code:
  {
    \__cmd_grab_E:nnNN {#1} {#2}
      \cs_set_protected:Npn
      \__cmd_peek_nonspace_remove:NTF
  }
\cs_new_protected:Npn \__cmd_grab_E_obey_spaces:w #1#2 \__cmd_run_code:
  {
    \__cmd_grab_E:nnNN {#1} {#2}
      \cs_set_protected_nopar:Npn
      \__cmd_peek_meaning_remove:NTF
  }
```

```
\cs_new_protected:Npn \__cmd_grab_E_long_obey_spaces:w #1#2 \__cmd_run_code:
  {
    \__cmd_grab_E:nnNN {#1} {#2}
      \cs_set_protected:Npn
      \__cmd_peek_meaning_remove:NTF
  }
```

A loop is needed here to allow a random ordering of keys. These are searched for one at a time, with any not found needing to be tracked: they can appear later. The grabbed values are held in a property list which is then turned into an ordered list to be passed back to the user.

```
\cs_new_protected:Npn \__cmd_grab_E:nnNN #1#2#3#4
  {
    \exp_after:wN #3 \l__cmd_fn_tl ##1##2##3
      {
        \prop_put:Nnn \l__cmd_tmp_prop {##1} {##3}
        \__cmd_grab_E_loop:NnN #4 { } ##2 \q_recursion_stop
      }
    \prop_clear:N \l__cmd_tmp_prop
    \tl_set:Nn \l__cmd_signature_tl {#2}
    \cs_set_protected:Npn \__cmd_grab_E_finalise:
      {
        \tl_map_inline:nn {#1}
          {
            \prop_get:NnNF \l__cmd_tmp_prop {####1} \l__cmd_tmpb_tl
              { \tl_set_eq:NN \l__cmd_tmpb_tl \c_novalue_tl }
            \tl_put_right:Nx \l__cmd_args_tl
              { { \exp_not:V \l__cmd_tmpb_tl } }
          }
        \l__cmd_signature_tl \__cmd_run_code:
      }
    \__cmd_grab_E_loop:NnN #4 { } #1 \q_recursion_tail \q_recursion_stop
  }
\cs_new_protected:Npn \__cmd_grab_E_loop:NnN #1#2#3#4 \q_recursion_stop
  {
    \cs_if_eq:NNTF #3 \q_recursion_tail
      { \__cmd_grab_E_finalise: }
      {
        #1 #3
          { \l__cmd_fn_tl #3 {#2#4} }
          { \__cmd_grab_E_loop:NnN #1 {#2#3} #4 \q_recursion_stop }
      }
  }
\cs_new_protected:Npn \__cmd_grab_E_finalise: { }
```

(*End of definition for* `\__cmd_grab_E:w` *and others.*)

`\__cmd_grab_m:w`
`\__cmd_grab_m_long:w`

Collecting a single mandatory argument is quite easy.

```
\cs_new_protected:Npn \__cmd_grab_m:w #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \exp_after:wN \cs_set_protected_nopar:Npn \l__cmd_fn_tl ##1
      { \__cmd_add_arg:n {##1} }
    \l__cmd_fn_tl
```

```
  }
\cs_new_protected:Npn \__cmd_grab_m_long:w #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl ##1
      { \__cmd_add_arg:n {##1} }
    \l__cmd_fn_tl
  }
```

(*End of definition for* `\__cmd_grab_m:w` *and* `\__cmd_grab_m_long:w`.)

`\__cmd_grab_m_1:w`
`\__cmd_grab_m_2:w`
`\__cmd_grab_m_3:w`
`\__cmd_grab_m_4:w`
`\__cmd_grab_m_5:w`
`\__cmd_grab_m_6:w`
`\__cmd_grab_m_7:w`
`\__cmd_grab_m_8:w`
`\__cmd_grab_m_9:w`
`\__cmd_grab_m_aux:Nnnnnnnnn`

Grabbing 1–8 mandatory arguments is done by giving 8–1 known arguments to a 9-argument function that stores them in `\l__cmd_args_tl`. For simplicity, grabbing 9 mandatory arguments is done by grabbing 5 then 4 arguments.

```
\cs_new_protected_nopar:Npn \__cmd_grab_m_aux:Nnnnnnnnn #1#2#3#4#5#6#7#8#9
  {
    \tl_put_right:No \l__cmd_args_tl
      { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }
    \l__cmd_signature_tl \__cmd_run_code:
  }
\cs_new_protected:cpn { __cmd_grab_m_1:w } #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
    \l__cmd_fn_tl \use_none:nnnnnnn { } { } { } { } { } { } { }
  }
\cs_new_protected:cpn { __cmd_grab_m_2:w } #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
    \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { }
  }
\cs_new_protected:cpn { __cmd_grab_m_3:w } #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
    \l__cmd_fn_tl \use_none:nnnnn { } { } { } { } { }
  }
\cs_new_protected:cpn { __cmd_grab_m_4:w } #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
    \l__cmd_fn_tl \use_none:nnnn { } { } { } { }
  }
\cs_new_protected:cpn { __cmd_grab_m_5:w } #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
    \l__cmd_fn_tl \use_none:nnn { } { } { }
  }
\cs_new_protected:cpn { __cmd_grab_m_6:w } #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
```

```
    \l__cmd_fn_tl \use_none:nn { } { }
  }
\cs_new_protected:cpn { __cmd_grab_m_7:w } #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
    \l__cmd_fn_tl \use_none:n { }
  }
\cs_new_protected:cpn { __cmd_grab_m_8:w } #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
    \l__cmd_fn_tl \prg_do_nothing:
  }
\cs_new_protected:cpx { __cmd_grab_m_9:w }
  {
    \exp_not:c { __cmd_grab_m_5:w }
    \exp_not:c { __cmd_grab_m_4:w }
  }
```

(*End of definition for* `\__cmd_grab_m_1:w` *and others.*)

`\__cmd_grab_R:w`
`\__cmd_grab_R_long:w`
`\__cmd_grab_R_aux:NNnN`

The grabber for R-type arguments is basically the same as that for D-type ones, but always skips spaces (as it is mandatory) and has a hard-coded error message.

```
\cs_new_protected:Npn \__cmd_grab_R:w #1#2#3 \__cmd_run_code:
  { \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected_nopar:Npn }
\cs_new_protected:Npn \__cmd_grab_R_long:w #1#2#3 \__cmd_run_code:
  { \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected:Npn }
\cs_new_protected:Npn \__cmd_grab_R_aux:NNnN #1#2#3#4
  {
    \__cmd_grab_D_aux:NNnNN #1 #2 {#3} #4 \use_ii:nn
    \__cmd_peek_nonspace_remove:NTF #1
      { \__cmd_grab_D_call:Nw #1 }
      {
        \msg_error:nnxx { cmd } { missing-required }
          { \__cmd_environment_or_command: }
          { \token_to_str:N #1 }
        \__cmd_add_arg:o \c_novalue_tl
      }
  }
```

(*End of definition for* `\__cmd_grab_R:w`, `\__cmd_grab_R_long:w`, *and* `\__cmd_grab_R_aux:NNnN`.)

`\__cmd_grab_t:w`
`\__cmd_grab_t_obey_spaces:w`
`\__cmd_grab_t_aux:NNw`

Dealing with a token is quite easy. Check the match, remove the token if needed and add a flag to the output.

```
\cs_new_protected:Npn \__cmd_grab_t:w
  { \__cmd_grab_t_aux:NNw \__cmd_peek_nonspace_remove:NTF }
\cs_new_protected:Npn \__cmd_grab_t_obey_spaces:w
  { \__cmd_grab_t_aux:NNw \__cmd_peek_meaning_remove:NTF }
\cs_new_protected:Npn \__cmd_grab_t_aux:NNw #1#2#3 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#3}
    \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl
      {
```

```
        #1 #2
          { \__cmd_add_arg:n { \BooleanTrue } }
          { \__cmd_add_arg:n { \BooleanFalse } }
      }
    \l__cmd_fn_tl
  }
```

(*End of definition for* `\__cmd_grab_t:w`, `\__cmd_grab_t_obey_spaces:w`, *and* `\__cmd_grab_t_aux:NNw`.)

`\l__cmd_v_arg_tl`

```
\tl_new:N \l__cmd_v_arg_tl
```

`\__cmd_grab_v:w`
`\__cmd_grab_v_long:w`
`\__cmd_grab_v_aux:w`
`\__cmd_grab_v_group_end:`

Firstly, it is necessary to change `\tex_endlinechar:D` so that newlines in different catcode regimes (e.g., `\ExplSyntaxOn`) are not misinterpreted as spaces. The opening delimiter is the first non-space token, and is never read verbatim. This is required by consistency with the case where the preceding argument was optional and absent: then TEX has already read and tokenized that token when looking for the optional argument. The first thing is thus to check is that this delimiter is a character, and to distinguish the case of a left brace (in that case, `\group_align_safe_end:` is needed to compensate for the begin-group character that was just seen). Then set verbatim catcodes with `\__cmd_grab_v_aux_catcodes:`.

The group keep catcode changes local, and `\group_align_safe_begin/end:` allow to use a character with category code 4 (normally `&`) as the delimiter (all commands do `\group_align_safe_begin/end:`, so there's no need to do that again here). It is ended by `\__cmd_grab_v_group_end:`, which smuggles the collected argument out of the group.

```
\cs_new_protected:Npn \__cmd_grab_v:w
  {
    \bool_set_false:N \l__cmd_long_bool
    \__cmd_grab_v_aux:w
  }
\cs_new_protected:Npn \__cmd_grab_v_long:w
  {
    \bool_set_true:N \l__cmd_long_bool
    \__cmd_grab_v_aux:w
  }
\cs_new_protected:Npn \__cmd_grab_v_aux:w #1 \__cmd_run_code:
  {
    \tl_set:Nn \l__cmd_signature_tl {#1}
    \group_begin:
      \tex_escapechar:D = 92 \scan_stop:
      \tex_endlinechar:D = `\^^M \scan_stop:
      \tl_clear:N \l__cmd_v_arg_tl
      \peek_remove_spaces:n
        {
          \peek_meaning_remove:NTF \c_group_begin_token
            {
              \group_align_safe_end:
              \__cmd_grab_v_bgroup:
            }
            {
              \peek_N_type:TF
```

```
                { \__cmd_grab_v_aux_test:N }
                { \__cmd_grab_v_aux_abort:n { } }
            }
        }
  }
\cs_new_protected:Npn \__cmd_grab_v_group_end:
  {
      \exp_args:NNNo
    \group_end:
    \tl_set:Nn \l__cmd_v_arg_tl { \l__cmd_v_arg_tl }
  }
```

(*End of definition for* `\__cmd_grab_v:w` *and others.*)

`\__cmd_grab_v_aux_test:N`
`\__cmd_grab_v_aux_loop:N`
`\__cmd_grab_v_aux_loop:NN`
`\__cmd_grab_v_aux_loop_end:`

Check that the opening delimiter is a character, setup category codes, then start reading tokens one by one, keeping the delimiter as an argument. If the verbatim was not nested, we will be grabbing one character at each step. Unfortunately, it can happen that what follows the verbatim argument is already tokenized. Thus, we check at each step that the next token is indeed a "nice" character, *i.e.*, is not a character with category code 1 (begin-group), 2 (end-group) or 6 (macro parameter), nor the space character, with category code 10 and character code 32, nor a control sequence. The partially built argument is stored in `\l__cmd_v_arg_tl`. If we ever meet a token which we cannot grab (non-N-type), or which is not a character according to `\__cmd_grab_v_token_if_char:NTF`, then we bail out with `\__cmd_grab_v_aux_abort:n`. Otherwise, we stop at the first character matching the delimiter.

```
\cs_new_protected:Npn \__cmd_grab_v_aux_test:N #1
  {
    \__cmd_grab_v_token_if_char:NTF #1
      {
        \__cmd_grab_v_aux_put:N #1
        \__cmd_grab_v_aux_catcodes:
        \__cmd_grab_v_aux_loop:N #1
      }
      { \__cmd_grab_v_aux_abort:n {#1} #1 }
  }
\cs_new_protected:Npn \__cmd_grab_v_aux_loop:N #1
  {
    \peek_N_type:TF
      { \__cmd_grab_v_aux_loop:NN #1 }
      { \__cmd_grab_v_aux_abort:n { } }
  }
\cs_new_protected:Npn \__cmd_grab_v_aux_loop:NN #1#2
  {
    \__cmd_grab_v_token_if_char:NTF #2
      {
        \token_if_eq_charcode:NNTF #1 #2
          { \__cmd_grab_v_aux_loop_end: }
          {
            \__cmd_grab_v_aux_put:N #2
            \__cmd_grab_v_aux_loop:N #1
          }
      }
      { \__cmd_grab_v_aux_abort:n {#2} #2 }
```

```
  }
\cs_new_protected:Npn \__cmd_grab_v_aux_loop_end:
  {
    \__cmd_grab_v_group_end:
    \__cmd_add_arg:x { \tl_tail:N \l__cmd_v_arg_tl }
  }
```

(*End of definition for* `\__cmd_grab_v_aux_test:N` *and others.*)

`\l__cmd_v_nesting_int`

```
\int_new:N \l__cmd_v_nesting_int
```

`\__cmd_grab_v_bgroup:`
`\__cmd_grab_v_bgroup_loop:`
`\__cmd_grab_v_bgroup_loop:N`

If the opening delimiter is a left brace, we keep track of how many left and right braces were encountered so far in `\l__cmd_v_nesting_int` (the methods used for optional arguments cannot apply here), and stop as soon as it reaches 0.

Some care was needed when removing the opening delimiter, which has already been assigned category code 1: using `\peek_meaning_remove:NTF` in the `\__cmd_grab_v_aux:w` function would break within alignments. Instead, we first convert that token to a string, and remove the result as a normal undelimited argument.

```
\cs_new_protected:Npx \__cmd_grab_v_bgroup:
  {
    \exp_not:N \__cmd_grab_v_aux_catcodes:
    \exp_not:n { \int_set:Nn \l__cmd_v_nesting_int { 1 } }
    \exp_not:N \__cmd_grab_v_aux_put:N \iow_char:N \{
    \exp_not:N \__cmd_grab_v_bgroup_loop:
  }
\cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:
  {
    \peek_N_type:TF
      { \__cmd_grab_v_bgroup_loop:N }
      { \__cmd_grab_v_aux_abort:n { } }
  }
\cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:N #1
  {
    \__cmd_grab_v_token_if_char:NTF #1
      {
        \token_if_eq_charcode:NNTF \c_group_end_token #1
          {
            \int_decr:N \l__cmd_v_nesting_int
            \int_compare:nNnTF \l__cmd_v_nesting_int > 0
              {
                \__cmd_grab_v_aux_put:N #1
                \__cmd_grab_v_bgroup_loop:
              }
              { \__cmd_grab_v_aux_loop_end: }
          }
          {
            \token_if_eq_charcode:NNT \c_group_begin_token #1
              { \int_incr:N \l__cmd_v_nesting_int }
            \__cmd_grab_v_aux_put:N #1
            \__cmd_grab_v_bgroup_loop:
          }
      }
```

```
      { \__cmd_grab_v_aux_abort:n {#1} #1 }
  }
```

(*End of definition for* `\__cmd_grab_v_bgroup:` , `\__cmd_grab_v_bgroup_loop:` , *and* `\__cmd_grab_v_bgroup_loop:N`.)

`\__cmd_grab_v_aux_catcodes:`
`\__cmd_grab_v_aux_abort:n`

The approach for short verbatim arguments is to make the end-line character a macro parameter character: this is forbidden by the rest of the code. Then the error branch can check what caused the bail out and give the appropriate error message.

```
\cs_new_protected:Npn \__cmd_grab_v_aux_catcodes:
  {
    \cs_set_eq:NN \do \char_set_catcode_other:N
    \dospecials
    \bool_if:NTF \l__cmd_long_bool
      { \char_set_catcode_other:n { \tex_endlinechar:D } }
      { \char_set_catcode_parameter:n { \tex_endlinechar:D } }
  }
\cs_new_protected:Npn \__cmd_grab_v_aux_abort:n #1
  {
    \__cmd_grab_v_group_end:
    \exp_after:wN \exp_after:wN \exp_after:wN
      \peek_meaning_remove:NTF \char_generate:nn { \tex_endlinechar:D } { 6 }
      {
        \msg_error:nnxxx { cmd } { verbatim-nl }
          { \__cmd_environment_or_command: }
          { \tl_to_str:N \l__cmd_v_arg_tl }
          { \tl_to_str:n {#1} }
        \__cmd_add_arg:o \c_novalue_tl
      }
      {
        \msg_error:nnxxx { cmd } { verbatim-tokenized }
          { \__cmd_environment_or_command: }
          { \tl_to_str:N \l__cmd_v_arg_tl }
          { \tl_to_str:n {#1} }
        \__cmd_add_arg:o \c_novalue_tl
      }
  }
```

(*End of definition for* `\__cmd_grab_v_aux_catcodes:` *and* `\__cmd_grab_v_aux_abort:n`.)

`\__cmd_grab_v_aux_put:N`

Storing one token in the collected argument. Most tokens are converted to category code 12, with the exception of active characters, and spaces (not sure what should be done for those).

```
⟨latexrelease⟩\IncludeInRelease{2024/06/01}{\__cmd_grab_v_aux_put:N}%
⟨latexrelease⟩   {Endlines~as~\obeyedline}
\cs_new_protected:Npn \__cmd_grab_v_aux_put:N #1
  {
    \tl_put_right:Nx \l__cmd_v_arg_tl
      {
        \token_if_active:NTF #1
          { \exp_not:N #1 }
          {
            \int_compare:nNnTF {`#1} = \tex_endlinechar:D
              { \exp_not:N \obeyedline }
```

```
                { \token_to_str:N #1 }
            }
        }
    }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__cmd_grab_v_aux_put:N}%
⟨latexrelease⟩  {Endlines~as~\obeyedline}
⟨latexrelease⟩\cs_new_protected:Npn \__cmd_grab_v_aux_put:N #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_put_right:Nx \l__cmd_v_arg_tl
⟨latexrelease⟩      {
⟨latexrelease⟩        \token_if_active:NTF #1
⟨latexrelease⟩          { \exp_not:N #1 } { \token_to_str:N #1 }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__cmd_grab_v_aux_put:N`.)

`\__cmd_grab_v_token_if_char:NTF`

This function assumes that the escape character is printable. Then the string representation of control sequences is at least two characters, and `\str_tail:n` only removes the escape character. Macro parameter characters are doubled by `\tl_to_str:n`, and will also yield a non-empty result, hence are not considered as characters.

```
\cs_new_protected:Npn \__cmd_grab_v_token_if_char:NTF #1
  { \str_if_eq:eeTF { } { \str_tail:n {#1} } }
```

(*End of definition for* `\__cmd_grab_v_token_if_char:NTF`.)

`\__cmd_add_arg:n`
`\__cmd_add_arg:V`
`\__cmd_add_arg:o`
`\__cmd_add_arg:x`

When an argument is found it is stored, then further arguments are grabbed by calling `\l__cmd_signature_tl`.

```
\cs_new_protected:Npn \__cmd_add_arg:n #1
  {
    \tl_put_right:Nn \l__cmd_args_tl { {#1} }
    \l__cmd_signature_tl \__cmd_run_code:
  }
\cs_generate_variant:Nn \__cmd_add_arg:n { V , o , x }
```

(*End of definition for* `\__cmd_add_arg:n`.)

1.9 Grabbing arguments expandably

`\__cmd_expandable_grab_D:w`
`\__cmd_expandable_grab_D:NNNwNNn`
`\__cmd_expandable_grab_D:NNNwNNnnn`
`\__cmd_expandable_grab_D:Nw`
`\__cmd_expandable_grab_D:nnNNNwNN`

The first step is to grab the first token or group. The generic grabbers `\`⟨*function*⟩`␣` and `\`⟨*function*⟩`␣` are just after `\q__cmd`, we go and find them (and use the long one).

```
\cs_new:Npn \__cmd_expandable_grab_D:w #1 \q__cmd #2#3
  { #2 { \__cmd_expandable_grab_D:NNNwNNn #1 \q__cmd #2 #3 } }
```

We then wish to test whether `#7`, which we just grabbed, is exactly `#2`. A preliminary test is whether their string representations coincide, then expand the only grabber function we have, `#1`, once: the two strings below are equal if and only if `#7` matches `#2` exactly.[2] The

[2] It is obvious that if `#7` matches `#2` then the strings are equal. We must check the converse. The right-hand-side of `\str_if_eq:onTF` does not end with `#3`, implying that the grabber function took everything as its arguments. The first brace group can only be empty if `#7` starts with `#2`, otherwise the brace group preceding `#7` would not vanish. The third brace group is empty, thus the `\q__cmd` that was used by our grabber `#1` must be the one that we inserted (not some token in `#7`), hence the second brace group contains the end of `#7` followed by `#2`. Since this is `#2` on the right-hand-side, and no brace can be lost there, `#7` must contain nothing else than its leading `#2`.

preliminary test is needed as #7 could validly contain \par (because a later mandatory argument could be long) and our grabber may be short. If #7 does not match #2, then the optional argument is missing, we use the default -NoValue-, and put back the argument #7 in the input stream.

If it does match, then interesting things need to be done. We will grab the argument piece by piece, with the following pattern:

⟨*grabber*⟩ {⟨*tokens*⟩}
\q_nil {⟨*piece 1*⟩} ⟨*piece 2*⟩ \ERROR \q__cmd
\q_nil ⟨*input stream*⟩

The ⟨*grabber*⟩ will find an opening delimiter in ⟨*piece 2*⟩, take the \q__cmd as a second delimiter, and find more material delimited by the closing delimiter in the ⟨*input stream*⟩. We then move the part before the opening delimiter from ⟨*piece 2*⟩ to ⟨*piece 1*⟩, and the material taken from the ⟨*input stream*⟩ to the ⟨*piece 2*⟩. Thus, the argument moves gradually from the ⟨*input stream*⟩ to the ⟨*piece 2*⟩, then to the ⟨*piece 1*⟩ when we have made sure to find all opening and closing delimiters. This two-step process ensures that nesting works: the number of opening delimiters minus closing delimiters in ⟨*piece 1*⟩ is always equal to the number of closing delimiters in ⟨*piece 2*⟩. We stop grabbing arguments once the ⟨*piece 2*⟩ contains no opening delimiter any more, hence the balance is reached, and the final argument is ⟨*piece 1*⟩ ⟨*piece 2*⟩. The indirection via __cmd_tmp:w allows to insert -NoValue- expanded.

```
\cs_set_protected:Npn \__cmd_tmp:w #1
  {
    \cs_new:Npn \__cmd_expandable_grab_D:NNNwNNn ##1##2##3##4 \q__cmd ##5##6##7
      {
        \str_if_eq:nnTF {##2} {##7}
          {
            \str_if_eq:onTF
              { ##1 { } { } ##7 ##2 \q__cmd ##3 }
              { { } {##2} { } }
          }
          { \use_ii:nn }
          {
            ##1
              { \__cmd_expandable_grab_D:NNNwNNnnn ##1##2##3##4 \q__cmd ##5##6 }
              \q_nil { } ##2 \ERROR \q__cmd \ERROR
          }
          { ##4 {#1} \q__cmd ##5 ##6 {##7} }
      }
  }
\exp_args:No \__cmd_tmp:w { \c_novalue_tl }
```

At this stage, #7 is \q_nil {⟨*piece 1*⟩} ⟨*more for piece 1*⟩, and we want to concatenate all that, removing \q_nil, and keeping the opening delimiter #2. Simply use \use_ii:nn. Also, #8 is ⟨*remainder of piece 2*⟩ \ERROR, and #9 is \ERROR ⟨*more for piece 2*⟩. We concatenate those, replacing the two \ERROR by the closing delimiter #3.

```
\cs_new:Npn \__cmd_expandable_grab_D:NNNwNNnnn #1#2#3#4 \q__cmd #5#6#7#8#9
  {
    \exp_args:Nof \__cmd_expandable_grab_D:nnNNNwNN
      { \use_ii:nn #7 #2 }
      { \__cmd_expandable_grab_D:Nw #3 \exp_stop_f: #8 #9 }
    #1#2#3 #4 \q__cmd #5 #6
```

```
  }
\cs_new:Npn \__cmd_expandable_grab_D:Nw #1#2 \ERROR \ERROR { #2 #1 }
```

Armed with our two new ⟨*pieces*⟩, we are ready to loop. However, we must first see if ⟨*piece 2*⟩ (here #2) contains any opening delimiter #4. Again, we expand #3, this time removing its whole output with \use_none:nnn. The test is similar to \tl_if_in:nnTF. The token list is empty if and only if #2 does not contain the opening delimiter. In that case, we are done, and put the argument (from which we remove a spurious pair of delimiters coming from how we started the loop). Otherwise, we go back to looping with __cmd_expandable_grab_D:NNNwNNnnn. The code to deal with brace stripping is much the same as for the non-expandable case.

```
\cs_new:Npn \__cmd_expandable_grab_D:nnNNNwNN #1#2#3#4#5#6 \q__cmd #7#8
  {
    \exp_args:No \tl_if_empty:oTF
      { #3 { \use_none:nnn } #2 \q__cmd #5 #4 \q__cmd #5 }
      {
        \tl_if_blank:oTF { \use_none:nn #1#2 }
          { \__cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
          {
            \str_if_eq:eeTF
              { \exp_not:o { \use_none:nn #1#2 } }
              { { \exp_not:o { \use_iii:nnnn #1#2 \q_nil } } }
              { \__cmd_put_arg_expandable:ow { \use_iii:nnn #1#2 } }
              { \__cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
          }
            #6 \q__cmd #7 #8
      }
      {
        #3
          { \__cmd_expandable_grab_D:NNNwNNnnn #3#4#5#6 \q__cmd #7 #8 }
          \q_nil {#1} #2 \ERROR \q__cmd \ERROR
      }
  }
```

(*End of definition for* __cmd_expandable_grab_D:w *and others.*)

__cmd_expandable_grab_D_alt:w
__cmd_expandable_grab_D_alt:NNwNNn
__cmd_expandable_grab_D_alt:Nwn

When the delimiters are identical, nesting is not possible and a simplified approach is used. The test concept here is the same as for the case where the delimiters are different but there cannot be any nesting.

```
\cs_new:Npn \__cmd_expandable_grab_D_alt:w #1 \q__cmd #2#3
  { #2 { \__cmd_expandable_grab_D_alt:NNwNNn #1 \q__cmd #2 #3 } }
\cs_set_protected:Npn \__cmd_tmp:w #1
  {
    \cs_new:Npn \__cmd_expandable_grab_D_alt:NNwNNn ##1##2##3 \q__cmd ##4##5##6
      {
        \str_if_eq:nnTF {##6} {##2}
          {
            \str_if_eq:onTF
              { ##1 { } ##6 ##2 ##2 }
              { { } ##2 }
          }
          { \use_ii:nn }
          {
            ##1
```

```
              { \__cmd_expandable_grab_D_alt:NNwn ##4 ##5 ##3 \q__cmd }
              ##6 \ERROR
          }
          { ##3 {#1} \q__cmd ##4 ##5 {##6} }
      }
  }
\exp_args:No \__cmd_tmp:w { \c_novalue_tl }
\cs_new:Npn \__cmd_expandable_grab_D_alt:NNwn #1#2#3 \q__cmd #4
  {
    \tl_if_blank:oTF { \use_none:n #4 }
      { \__cmd_put_arg_expandable:ow { \use_none:n #4 } }
      {
        \str_if_eq:eeTF
          { \exp_not:o { \use_none:n #4 } }
          { { \exp_not:o { \use_ii:nnn #4 \q_nil } } }
          { \__cmd_put_arg_expandable:ow { \use_ii:nn #4 } }
          { \__cmd_put_arg_expandable:ow { \use_none:n #4 } }
      }
        #3 \q__cmd #1 #2
  }
```

(*End of definition for* `\__cmd_expandable_grab_D_alt:w`, `\__cmd_expandable_grab_D_alt:NNwNNn`, *and* `\__cmd_expandable_grab_D_alt:Nwn`.)

`\__cmd_expandable_grab_E:w`
`\__cmd_expandable_grab_E_long:w`
`\__cmd_expandable_grab_E_aux:w`
`\__cmd_expandable_grab_E_test:nnw`
`\__cmd_expandable_grab_E_loop:nnnNNw`
`\__cmd_expandable_grab_E_find:w`
`\__cmd_expandable_grab_E_find:nnw`
`\__cmd_expandable_grab_E_end:nnw`

We keep track of long/short by placing the appropriate grabber as the third token after `\q__cmd`; it is eventually removed by the `end:nnw` auxiliary. The `aux:w` auxiliary will be called repeatedly with two arguments: the set of pairs ⟨*parser*⟩ ⟨*token*⟩, and the set of arguments found so far (initially all `{-NoValue-}`). At each step, grab what follows in the input stream then call the `loop:nnnNNw` auxiliary to compare it with each possible embellishment in turn. This auxiliary's `#1` is what was found in the input, `#2` collects ⟨*parser*⟩ ⟨*token*⟩ pairs that did not match, `#3` collects the corresponding arguments found previously, `#4` and `#5` is the current pair, `#6` is the remaining pairs, `#7` is empty or two `\q_nil`, and `#8` is the current argument. If none of the pairs matched (determined by `\quark_if_nil:NTF`) then call the `end` auxiliary to stop looking for embellishments, remembering to put what was grabbed in the input back where it belongs, and storing the arguments found just before `\q__cmd`. If the current argument `#8` is not `-NoValue-` or if the input `#1` does not match `#5` (see `t`-type arguments below for a similar `\str_if_eq:onTF` test) then carry on the loop. Otherwise, we found a new embellishment: grab the corresponding argument in the input using the `find:w` auxiliary. To avoid losing braces around that auxiliary's argument we include a space, which will be eliminated in the next loop through embellishments.

```
\cs_new:Npn \__cmd_expandable_grab_E:w #1 \q__cmd #2#3
  { \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2 #3 #3 }
\cs_new:Npn \__cmd_expandable_grab_E_long:w #1 \q__cmd #2#3
  { \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2 #3 #2 }
\cs_new:Npn \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2#3#4
  { #2 { \__cmd_expandable_grab_E_test:nnw #1 \q__cmd #2 #3 #4 } }
\cs_new:Npn \__cmd_expandable_grab_E_test:nnw #1#2#3 \q__cmd #4#5#6#7
  {
    \__cmd_expandable_grab_E_loop:nnnNNw {#7} { } { }
      #1 \q_nil \q_nil \q_nil \q_mark #2 \q_nil
    #3 \q__cmd #4 #5 #6
  }
```

```
\cs_new:Npn \__cmd_expandable_grab_E_loop:nnnNNw
    #1#2#3#4#5#6 \q_nil #7 \q_mark #8
  {
    \quark_if_nil:NTF #4
      { \__cmd_expandable_grab_E_end:nnw {#1} {#3} }
      {
        \tl_if_novalue:nTF {#8}
          { \str_if_eq:onTF { #4 { } #1 #5 } {#5} }
          { \use_ii:nn }
            { \__cmd_expandable_grab_E_find:w { #2 #4 #5 #6 } {#3} ~ }
            {
              \__cmd_expandable_grab_E_loop:nnnNNw
                {#1} { #2 #4 #5 } { #3 {#8} }
                #6 \q_nil #7 \q_mark
            }
      }
  }
\cs_new:Npn \__cmd_expandable_grab_E_find:w #1 \q__cmd #2#3#4
  { #4 { \__cmd_expandable_grab_E_find:nnw #1 \q__cmd #2 #3 #4 } }
\cs_new:Npn \__cmd_expandable_grab_E_find:nnw #1#2#3 \q_nil #4 \q__cmd #5#6#7#8
  { \__cmd_expandable_grab_E_aux:w {#1} { #2 {#8} #3 } #4 \q__cmd #5 #6 #7 }
\cs_new:Npn \__cmd_expandable_grab_E_end:nnw #1#2#3 \q__cmd #4#5#6
  { #3 #2 \q__cmd #4 #5 {#1} }
```

(*End of definition for* `\__cmd_expandable_grab_E:w` *and others.*)

`\__cmd_expandable_grab_m:w`
`\__cmd_expandable_grab_m_long:w`
`\__cmd_expandable_grab_m_aux:wNn`

The mandatory case is easy: find the auxiliary after the `\q__cmd`, and use it directly to grab the argument, then correctly position the argument before `\q__cmd`.

```
\cs_new:Npn \__cmd_expandable_grab_m:w #1 \q__cmd #2#3
  { #3 { \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2 #3 } }
\cs_new:Npn \__cmd_expandable_grab_m_long:w #1 \q__cmd #2#3
  { #2 { \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2 #3 } }
\cs_new:Npn \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2#3#4
  { #1 {#4} \q__cmd #2 #3 }
```

(*End of definition for* `\__cmd_expandable_grab_m:w`, `\__cmd_expandable_grab_m_long:w`, *and* `\__cmd_expandable_grab_m_aux:wNn`.)

`\__cmd_expandable_grab_R:w`
`\__cmd_expandable_grab_R_aux:NNNwNNn`

Much the same as for the D-type argument, with only the lead-off function varying.

```
\cs_new:Npn \__cmd_expandable_grab_R:w #1 \q__cmd #2#3
  { #2 { \__cmd_expandable_grab_R_aux:NNNwNNn #1 \q__cmd #2#3 } }
\cs_set_protected:Npn \__cmd_tmp:w #1
  {
    \cs_new:Npn \__cmd_expandable_grab_R_aux:NNNwNNn ##1##2##3##4 \q__cmd ##5##6##7
      {
        \str_if_eq:nnTF {##7} {##2}
          {
            \str_if_eq:onTF
              { ##1 { } { } ##7 ##2 \q__cmd ##3 }
              { { } {##2} { } }
          }
          { \use_ii:nn }
          {
            ##1
              { \__cmd_expandable_grab_D:NNNwNNnnn ##1##2##3##4 \q__cmd ##5##6 }
```

```
              \q_nil { } ##2 \ERROR \q__cmd \ERROR
          }
          {
            \msg_expandable_error:nnff { cmd } { missing-required }
              { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N ##5 } }
              { \tl_to_str:n {##2} }
            ##4 {#1} \q__cmd ##5 ##6 {##7}
          }
      }
  }
\exp_args:No \__cmd_tmp:w { \c_novalue_tl }
```

(*End of definition for* `\__cmd_expandable_grab_R:w` *and* `\__cmd_expandable_grab_R_aux:NNNwNNn`.)

`\__cmd_expandable_grab_R_alt:w`
`\__cmd_expandable_grab_R_alt_aux:NNwNNn`

When the delimiters are identical, nesting is not possible and a simplified approach is used. The test concept here is the same as for the case where the delimiters are different.

```
\cs_new:Npn \__cmd_expandable_grab_R_alt:w #1 \q__cmd #2#3
  { #2 { \__cmd_expandable_grab_R_alt_aux:NNwNNn #1 \q__cmd #2#3 } }
\cs_set_protected:Npn \__cmd_tmp:w #1
  {
    \cs_new:Npn \__cmd_expandable_grab_R_alt_aux:NNwNNn ##1##2##3 \q__cmd ##4##5##6
      {
        \str_if_eq:nnTF {##6} {##2}
          {
            \str_if_eq:onTF
              { ##1 { } ##6 ##2 ##2 }
              { { } ##2 }
          }
          { \use_ii:nn }
          {
            ##1
              { \__cmd_expandable_grab_D_alt:NNwn ##4 ##5 ##3 \q__cmd }
              ##6 \ERROR
          }
          {
            \msg_expandable_error:nnff { cmd } { missing-required }
              { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N ##4 } }
              { \tl_to_str:n {##2} }
            ##3 {#1} \q__cmd ##4 ##5 {##6}
          }
      }
  }
\exp_args:No \__cmd_tmp:w { \c_novalue_tl }
```

(*End of definition for* `\__cmd_expandable_grab_R_alt:w` *and* `\__cmd_expandable_grab_R_alt_aux:NNwNNn`.)

`\__cmd_expandable_grab_t:w`
`\__cmd_expandable_grab_t_aux:NNwn`

As for a D-type argument, here we compare the grabbed tokens using the only parser we have in order to work out if #2 is exactly equal to the output of the grabber.

```
\cs_new:Npn \__cmd_expandable_grab_t:w #1 \q__cmd #2#3
  { #2 { \__cmd_expandable_grab_t_aux:NNwn #1 \q__cmd #2 #3 } }
\cs_new:Npn \__cmd_expandable_grab_t_aux:NNwn #1#2#3 \q__cmd #4#5#6
  {
    \str_if_eq:onTF { #1 { } #6 #2 } {#2}
```

```
      { #3 { \BooleanTrue } \q__cmd #4 #5 }
      { #3 { \BooleanFalse } \q__cmd #4 #5 {#6} }
  }
```

(*End of definition for* `\__cmd_expandable_grab_t:w` *and* `\__cmd_expandable_grab_t_aux:NNwn`.)

`\__cmd_put_arg_expandable:nw`
`\__cmd_put_arg_expandable:ow`

A useful helper, to store arguments when they are ready.

```
\cs_new:Npn \__cmd_put_arg_expandable:nw #1#2 \q__cmd { #2 {#1} \q__cmd }
\cs_generate_variant:Nn \__cmd_put_arg_expandable:nw { o }
```

(*End of definition for* `\__cmd_put_arg_expandable:nw`.)

1.10 Argument processors

`\__cmd_bool_reverse:N`

A simple reversal.

```
\cs_new_protected:Npn \__cmd_bool_reverse:N #1
  {
    \bool_if:NTF #1
      { \tl_set:Nn \ProcessedArgument { \c_false_bool } }
      { \tl_set:Nn \ProcessedArgument { \c_true_bool } }
  }
```

(*End of definition for* `\__cmd_bool_reverse:N`.)

\l__cmd_split_list_seq
\l__cmd_split_list_tl
__cmd_split_list_multi:nn
__cmd_split_list_multi:nV
__cmd_split_list_single:Nn

Splitting can take place either at a single token or at a longer identifier. To deal with single active tokens, a two-part procedure is needed.

```
\seq_new:N \l__cmd_split_list_seq
\tl_new:N \l__cmd_split_list_tl
\cs_new_protected:Npn \__cmd_split_list:nn #1#2
  {
    \tl_if_single:nTF {#1}
      {
        \token_if_cs:NTF #1
          { \__cmd_split_list_multi:nn {#1} {#2} }
          { \__cmd_split_list_single:Nn #1 {#2} }
      }
      { \__cmd_split_list_multi:nn {#1} {#2} }
  }
\cs_new_protected:Npn \__cmd_split_list_multi:nn #1#2
  {
    \seq_set_split:Nnn \l__cmd_split_list_seq {#1} {#2}
    \tl_clear:N \ProcessedArgument
    \seq_map_inline:Nn \l__cmd_split_list_seq
      { \tl_put_right:Nn \ProcessedArgument { {##1} } }
  }
\cs_generate_variant:Nn \__cmd_split_list_multi:nn { nV }
\group_begin:
\char_set_catcode_active:N \^^@
\cs_new_protected:Npn \__cmd_split_list_single:Nn #1#2
  {
    \tl_set:Nn \l__cmd_split_list_tl {#2}
    \group_begin:
    \char_set_lccode:nn { `\^^@ } { `#1 }
    \tex_lowercase:D
      {
        \group_end:
        \tl_replace_all:Nnn \l__cmd_split_list_tl { ^^@ }
      }   {#1}
     \__cmd_split_list_multi:nV {#1} \l__cmd_split_list_tl
   }
\group_end:
```

(*End of definition for* `\__cmd_split_list:nn`, `\__cmd_split_list_multi:nn`, *and* `\__cmd_split_list_single:Nn`.)

__cmd_split_argument:nnn
__cmd_split_argument_aux:nnnn
__cmd_split_argument_aux:n
__cmd_split_argument_aux:wn

Splitting to a known number of items is a special version of splitting a list, in which the limit is hard-coded and where there will always be exactly the correct number of output items. An auxiliary function is used to save on working out the token list length several times.

```
\cs_new_protected:Npn \__cmd_split_argument:nnn #1#2#3
  {
    \__cmd_split_list:nn {#2} {#3}
    \exp_args:Nf \__cmd_split_argument_aux:nnnn
      { \tl_count:N \ProcessedArgument }
      {#1} {#2} {#3}
  }
```

```
\cs_new_protected:Npn \__cmd_split_argument_aux:nnnn #1#2#3#4
  {
    \int_compare:nNnF {#1} = { #2 + 1 }
      {
        \int_compare:nNnTF {#1} > { #2 + 1 }
          {
            \tl_set:Nx \ProcessedArgument
              {
                \exp_last_unbraced:NnNo
                  \__cmd_split_argument_aux:n
                  { #2 + 1 }
                  \use_none_delimit_by_q_stop:w
                  \ProcessedArgument
                  \q_stop
              }
            \msg_error:nnxxx { cmd } { arg-split }
              { \tl_to_str:n {#3} } { \int_eval:n { #2 + 1 } }
              { \tl_to_str:n {#4} }
          }
          {
            \tl_put_right:Nx \ProcessedArgument
              {
                \prg_replicate:nn { #2 + 1 - (#1) }
                  { { \exp_not:V \c_novalue_tl } }
              }
          }
      }
  }
```

Auxiliaries to leave exactly the correct number of arguments in `\ProcessedArgument`.

```
\cs_new:Npn \__cmd_split_argument_aux:n #1
  { \prg_replicate:nn {#1} { \__cmd_split_argument_aux:wn } }
\cs_new:Npn \__cmd_split_argument_aux:wn #1 \use_none_delimit_by_q_stop:w #2
  {
    \exp_not:n { {#2} }
    #1
    \use_none_delimit_by_q_stop:w
  }
```

(*End of definition for* `\__cmd_split_argument:nnn` *and others.*)

`\__cmd_trim_spaces:n` This one is almost trivial.

```
\cs_new_protected:Npn \__cmd_trim_spaces:n #1
  { \tl_set:Nx \ProcessedArgument { \tl_trim_spaces:n {#1} } }
```

(*End of definition for* `\__cmd_trim_spaces:n`.)

1.11 Conversion to key–value form

This is implemented as a process but with no public interfaces, hence is treated separately from the others: it's a feature of ltcmd which just happens to use the same mechanism as a processor.

__cmd_arg_to_keyvalue:nn
__cmd_arg_to_keyvalue_braces:nnn
__cmd_arg_to_keyvalue_auxi:nnn
__cmd_arg_to_keyvalue_auxii:Nnnn
__cmd_arg_to_keyvalue_auxiii:nnn
__cmd_arg_to_keyvalue_auxiv:Nnnn
__cmd_arg_to_keyvalue_auxv:nn
__cmd_arg_to_keyvalue_loop:w
__cmd_arg_to_keyvalue_loop_group:n
__cmd_arg_to_keyvalue_loop_space:w
__cmd_arg_to_keyvalue_loop_N_type:N
__cmd_arg_to_keyvalue_math:w
__cmd_arg_to_keyvalue_math_N_type:N
__cmd_arg_to_keyvalue_math_group:n
__cmd_arg_to_keyvalue_math_space:w
__cmd_arg_to_keyvalue_set_default:nn
__cmd_arg_to_keyvalue_set_keyvalue:nn
__cmd_split_N_head_apply:Nn
__cmd_split_N_head_apply_aux:NNw

If the entire argument is braced, we treat as free text and return as the value for the text key. Alternatively, if the start of the input is `=`, then it is forced to be key–value. To avoid needing to worry about catcodes for this, and to allow spaces around the `=`, we use a series of steps rather than a delimited argument.

```
\cs_new_protected:Npn \__cmd_arg_to_keyvalue:nn #1#2
  {
    \tl_trim_spaces_apply:nN {#2} \__cmd_arg_to_keyvalue_braces:nnn
      {#1} {#2}
  }
\cs_new_protected:Npn \__cmd_arg_to_keyvalue_braces:nnn #1#2#3
  {
    \tl_if_head_is_group:nT {#1}
      {
        \tl_if_blank:oT { \use_none:n #1 }
          {
            \tl_set:Nx \ProcessedArgument { #2 = { \exp_not:n #1 } }
            \use_none:nnnn
          }
      }
    \__cmd_arg_to_keyvalue_auxi:nnn {#1} {#2} {#3}
  }
\cs_new:Npn \__cmd_arg_to_keyvalue_auxi:nnn #1
  {
    \tl_if_head_is_N_type:nTF {#1}
      { \__cmd_split_N_head_apply:Nn \__cmd_arg_to_keyvalue_auxii:Nnnn {#1} }
      { \__cmd_arg_to_keyvalue_auxv:nn }
  }
\cs_new:Npn \__cmd_arg_to_keyvalue_auxii:Nnnn #1#2
  {
    \str_if_eq:eeTF { \exp_not:n {#1} } { = }
      { \tl_trim_spaces_apply:nN {#2} \__cmd_arg_to_keyvalue_auxiii:nnn }
      { \__cmd_arg_to_keyvalue_auxv:nn }
  }
\cs_new:Npn \__cmd_arg_to_keyvalue_auxiii:nnn #1
  {
    \tl_if_head_is_N_type:nTF {#1}
      { \__cmd_split_N_head_apply:Nn \__cmd_arg_to_keyvalue_auxiv:Nnnn {#1} }
      { \__cmd_arg_to_keyvalue_auxv:nn }
  }
\cs_new:Npn \__cmd_arg_to_keyvalue_auxiv:Nnnn #1#2
  {
    \str_if_eq:eeTF { \exp_not:n {#1} } { , }
      { \tl_set:Nn \ProcessedArgument {#2} \use_none:nn }
      { \__cmd_arg_to_keyvalue_auxv:nn }
  }
```

The two clear-cut cases have been eliminated, and we therefore have to deal with a search for `=` signs. We need an "action" loop here so we do not get mislead by for example `{=}`. As the code here is for very much predictable types of input, we hard-code what constitutes math mode opening and closing. At the very beginning, the default key (`#1`) and the argument as given by the user (`#2`) are placed right after the `\q__cmd_recursion_stop`, so that when the recursion ends, the macros `\__cmd_arg_to_keyvalue_set_default:nn` or `\__cmd_arg_to_keyvalue_set_keyvalue:nn` can be used to grab these two items and

set the \ProcessedArgument accordingly.

```
\cs_new_protected:Npn \__cmd_arg_to_keyvalue_auxv:nn #1#2
  {
    \__cmd_arg_to_keyvalue_loop:w #2
      \q__cmd_recursion_tail \q__cmd_recursion_stop {#1} {#2}
  }
\cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop:w #1 \q__cmd_recursion_stop
  {
    \tl_if_head_is_N_type:nTF {#1}
      { \__cmd_arg_to_keyvalue_loop_N_type:N }
      {
        \tl_if_head_is_group:nTF {#1}
          { \__cmd_arg_to_keyvalue_loop_group:n }
          { \__cmd_arg_to_keyvalue_loop_space:w }
      }
        #1 \q__cmd_recursion_stop
  }
\cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_group:n #1
  { \__cmd_arg_to_keyvalue_loop:w }
\use:n { \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_space:w } ~
  { \__cmd_arg_to_keyvalue_loop:w }
\cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_N_type:N #1
  {
    \__cmd_if_recursion_tail_stop_do:Nn #1
      { \__cmd_arg_to_keyvalue_set_default:nn }
    \str_if_eq:nnTF {#1} { = }
      {
        \__cmd_use_i_delimit_by_q_recursion_stop:nw
          { \__cmd_arg_to_keyvalue_set_keyvalue:nn }
      }
      {
        \bool_lazy_or:nnTF
            { \token_if_math_toggle_p:N #1 }
            { \str_if_eq_p:nn {#1} { \( } }
          { \__cmd_arg_to_keyvalue_math:w }
          { \__cmd_arg_to_keyvalue_loop:w }
      }
  }
\cs_new_protected:Npn \__cmd_arg_to_keyvalue_math:w #1 \q__cmd_recursion_stop
  {
    \tl_if_head_is_N_type:nTF {#1}
      { \__cmd_arg_to_keyvalue_math_N_type:N }
      {
        \tl_if_head_is_group:nTF {#1}
          { \__cmd_arg_to_keyvalue_math_group:n }
          { \__cmd_arg_to_keyvalue_math_space:w }
      }
        #1 \q__cmd_recursion_stop
  }
\cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_N_type:N #1
  {
    \__cmd_if_recursion_tail_stop_do:Nn #1
      { \__cmd_arg_to_keyvalue_set_default:nn }
    \bool_lazy_or:nnTF
```

```
        { \token_if_math_toggle_p:N #1 }
        { \str_if_eq_p:nn {#1} { \) } }
      { \__cmd_arg_to_keyvalue_loop:w }
      { \__cmd_arg_to_keyvalue_math:w }
  }
\cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_group:n #1
  { \__cmd_arg_to_keyvalue_math:w }
\use:n { \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_space:w } ~
  { \__cmd_arg_to_keyvalue_math:w }
\cs_new_protected:Npn \__cmd_arg_to_keyvalue_set_default:nn #1#2
  { \tl_set:Nn \ProcessedArgument { #1 = {#2} } }
\cs_new_protected:Npn \__cmd_arg_to_keyvalue_set_keyvalue:nn #1#2
  { \tl_set:Nn \ProcessedArgument {#2} }
```

A utility to allow us to grab the first N-type token without risking brace stripping the rest of the input.

```
\cs_new:Npn \__cmd_split_N_head_apply:Nn #1#2
  { \exp:w \if_false: { \fi: \__cmd_split_N_head_apply_aux:NNw #1#2 } }
\cs_new:Npn \__cmd_split_N_head_apply_aux:NNw #1#2
  {
    \exp_after:wN \exp_end:
    \exp_after:wN #1 \exp_after:wN #2 \exp_after:wN { \if_false: } \fi:
  }

```

(*End of definition for* `\__cmd_arg_to_keyvalue:nn` *and others.*)

1.12 Utilities

`\__cmd_check_definable:nNT`
`\__cmd_check_definable_aux:nN`

Check that a token list is appropriate as a first argument of `\NewDocumentCommand` and similar functions and otherwise produce an error. First trim whitespace to allow for spaces around the actual command to be defined. If the result has multiple tokens, it is not a valid argument. The single token is a control sequence exactly if its string representation has more than one character (using `\token_to_str:N` rather than `\tl_to_str:n` to avoid problems with macro parameter characters, and setting `\tex_escapechar:D` to prevent it from being non-printable). Finally, check for an active character: this is done by lowercasing the token to fix its character code (arbitrarily to that of `?`) and comparing the result to an active `?`. Both control sequences and active characters are valid arguments, and non-active character tokens are not. In all cases, the group opened to keep assignments local must be closed.

```
\cs_new_protected:Npn \__cmd_check_definable:nNT #1
  { \tl_trim_spaces_apply:nN {#1} \__cmd_check_definable_aux:nN }
\group_begin:
  \char_set_catcode_active:n { `? }
  \cs_new_protected:Npn \__cmd_check_definable_aux:nN #1#2
    {
      \group_begin:
      \tl_if_single_token:nTF {#1}
        {
          \int_set:Nn \tex_escapechar:D { 92 }
          \exp_args:Nx \tl_if_empty:nTF
            { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
            {
```

```
              \exp_args:Nx \char_set_lccode:nn
                { ‘ \str_head:n {#1} } { ‘? }
              \tex_lowercase:D { \tl_if_eq:nnTF {#1} } { ? }
                { \group_end: \use_iii:nnn }
                { \group_end: \use_i:nnn }
            }
            { \group_end: \use_iii:nnn }
        }
        { \group_end: \use_ii:nnn }
      {
        \msg_error:nnxx { cmd } { not-definable }
          { \tl_to_str:n {#1} } { \token_to_str:N #2 }
      }
      {
        \msg_error:nnxx { cmd } { not-one-token }
          { \tl_to_str:n {#1} } { \token_to_str:N #2 }
      }
    }
\group_end:
```

(*End of definition for* `\__cmd_check_definable:nNT` *and* `\__cmd_check_definable_aux:nN`.)

`\__cmd_token_if_cs:NTF` Based on the definition of `\__cmd_check_definable_aux:nN` above, but only checks for an actual control sequence (*i.e.*, `\`⟨*anything*⟩). `\tex_escapechar:D` is temporarily changed to a known value and then it checks if `\string#1` contains more than one character: if it does, it's a control sequence. This test differs from `\token_if_cs:NTF` for example in `\token_if_cs:NTF \c_group_begin_token {T}{F}`, where `\token_if_cs:NTF` returns false.

```
\cs_new_protected:Npn \__cmd_token_if_cs:NTF #1
  {
    \group_begin:
      \int_set:Nn \tex_escapechar:D { 92 }
      \exp_args:Nx \tl_if_empty:nTF
          { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
        { \group_end: \use_ii:nn }
        { \group_end: \use_i:nn }
  }
```

(*End of definition for* `\__cmd_token_if_cs:NTF`.)

`\__cmd_tl_mapthread_function:NNN`
`\__cmd_tl_mapthread_function:nnN`
`\__cmd_tl_mapthread_loop:w`

Analogue of `\seq_mapthread_function:NNN` for token lists.

```
\cs_new:Npn \__cmd_tl_mapthread_function:NNN #1#2#3
  {
    \exp_after:wN \exp_after:wN
    \exp_after:wN \__cmd_tl_mapthread_loop:w
    \exp_after:wN \exp_after:wN
    \exp_after:wN #3
    \exp_after:wN #1
    \exp_after:wN \q_recursion_tail
    \exp_after:wN \q_mark
    #2
    \q_recursion_tail
    \q_recursion_stop
  }
```

```
\cs_new:Npn \__cmd_tl_mapthread_function:nnN #1#2#3
  {
    \__cmd_tl_mapthread_loop:w #3
      #1 \q_recursion_tail \q_mark
      #2 \q_recursion_tail \q_recursion_stop
  }
\cs_new:Npn \__cmd_tl_mapthread_loop:w #1#2#3 \q_mark #4
  {
    \quark_if_recursion_tail_stop:n {#2}
    \quark_if_recursion_tail_stop:n {#4}
    #1 {#2} {#4}
    \__cmd_tl_mapthread_loop:w #1#3 \q_mark
  }
```

(*End of definition for* `\__cmd_tl_mapthread_function:NNN`, `\__cmd_tl_mapthread_function:nnN`, *and* `\__cmd_tl_mapthread_loop:w`.)

`\__kernel_cmd_if_xparse:NTF`
`\__cmd_cmd_type_cases:NnnnnnF`
`\__cmd_cmd_if_xparse_aux:N`

To determine whether the command is an xparse command check that its `arg_spec` is empty (this also excludes non-macros) and that its `replacement_spec` starts with either `\__cmd_start:nNNnnn` (non-expandable command) or `\__cmd_start_expandable:nNNNNn` (expandable command) or `\__cmd_start_optimized:` (optimized command) or `\__cmd_start_env:nnnnn` (environment) or `\environment #1 end aux` (environment end).

This conditional is needed in several kernel modules and is therefore has a kernel-internal name.

```
\cs_new_protected:Npn \__cmd_cmd_type_cases:NnnnnnF #1 #2 #3 #4 #5 #6 #7
  {
    \exp_args:Ne \str_case_e:nnF
      {
        \exp_args:Nf \tl_if_empty:nT { \__kernel_cs_parameter_spec:N #1 }
          { \exp_not:N \exp_not:n { \exp_not:e { \tl_head:N #1 } } }
      }
      {
        { \exp_not:N \__cmd_start:nNNnnn } {#2}
        { \exp_not:N \__cmd_start_expandable:nNNNNn } {#3}
        { \exp_not:N \__cmd_start_optimized: } {#4}
        { \exp_not:N \__cmd_start_env:nnnnn } {#5}
        {
          \exp_after:wN \exp_not:N
            \cs:w environment~
              \exp_last_unbraced:Ne \use_none:nnn
                { \cs_to_str:N #1 } ~end~aux \cs_end:
        } {#6}
      }
      {#7}
  }
\cs_new_protected:Npn \__kernel_cmd_if_xparse:NTF #1
  {
    \__cmd_cmd_type_cases:NnnnnnF #1
      { } { } { } { } { } { \use_iii:nnn }
    \use_i:nn
  }
```

(*End of definition for* `\__kernel_cmd_if_xparse:NTF`, `\__cmd_cmd_type_cases:NnnnnnF`, *and* `\__cmd_cmd_if_xparse_aux:N`.)

__cmd_peek_nonspace:NTF
__cmd_peek_nonspace_remove:NTF
__cmd_peek_nonspace_aux:nNNTF

Collect spaces in a loop, and put the collected spaces back in the false branch of a call to \peek_meaning:NTF or \peek_meaning_remove:NTF.

```
\cs_new_protected:Npn \__cmd_peek_nonspace:NTF
  { \__cmd_peek_nonspace_aux:nNNTF { } \__cmd_peek_meaning:NTF }
\cs_new_protected:Npn \__cmd_peek_nonspace_remove:NTF
  { \__cmd_peek_nonspace_aux:nNNTF { } \__cmd_peek_meaning_remove:NTF }
\cs_new_protected:Npn \__cmd_peek_nonspace_aux:nNNTF #1#2#3#4#5
  {
    \peek_meaning_remove:NTF \c_space_token
      { \__cmd_peek_nonspace_aux:nNNTF { #1 ~ } #2 #3 {#4} {#5} }
      { #2 #3 { #4 } { #5 #1 } }
  }
```

(*End of definition for* __cmd_peek_nonspace:NTF*,* __cmd_peek_nonspace_remove:NTF*, and* __cmd_peek_nonspace_aux:nNNTF.)

__cmd_peek_meaning:NTF
__cmd_peek_meaning_remove:NTF
__cmd_peek_cs_check_equal:NNN
__cmd_peek_meaning_aux:NNTF
__cmd_peek_true_remove:NNw

Peek ahead for a token with a given meaning. In case the search token is a control sequence, also check that the ⟨*csname*⟩ is the same as the control sequence peeked at. This extra verification is necessary when the command is delimited by control sequence tokens (as opposed to character tokens), and we want the exact same control sequence to match.

```
\cs_new_protected:Npn \__cmd_peek_meaning:NTF
  { \__cmd_peek_meaning_aux:NNTF \c_false_bool }
\cs_new_protected:Npn \__cmd_peek_meaning_remove:NTF
  { \__cmd_peek_meaning_aux:NNTF \c_true_bool }
\cs_new_protected:Npn \__cmd_peek_meaning_aux:NNTF #1#2#3#4
  {
    \tl_set:Nn \l__cmd_tmpa_tl {#3}
    \tl_set:Nn \l__cmd_tmpb_tl {#4}
    \peek_meaning:NTF #2
      {
        \token_if_eq_meaning:NNTF #2 \c_group_begin_token
          { \__cmd_peek_true_remove:Nw #1 }
          {
            \__cmd_token_if_cs:NTF #2
              { \__cmd_peek_cs_check_equal:NNN #1 #2 }
              { \__cmd_peek_true_remove:Nw #1 }
          }
      }
      { \l__cmd_tmpb_tl }
  }
\cs_new_protected:Npn \__cmd_peek_cs_check_equal:NNN #1#2#3
  {
    \str_if_eq:nnTF {#2} {#3}
      { \__cmd_peek_true_remove:Nw #1 }
      { \l__cmd_tmpb_tl }
    #3
  }
\cs_new_protected:Npn \__cmd_peek_true_remove:Nw #1
  {
    \bool_if:NTF #1
      {
        \tex_afterassignment:D \l__cmd_tmpa_tl
        \cs_set_eq:NN \__cmd_tmp:w
```

```
      }
      { \l__cmd_tmpa_tl }
  }
```

(*End of definition for* `\__cmd_peek_meaning:NTF` *and others.*)

1.13 Messages

`\c__cmd_ignore_def_tl`

```
\tl_const:Nn \c__cmd_ignore_def_tl
  { \\ \\ LaTeX~will~ignore~this~entire~definition. }
```

`\__cmd_environment_or_command:` Two texts used in several messages.

```
\cs_new:Npn \__cmd_environment_or_command:
  {
    \bool_if:NTF \l__cmd_environment_bool
      { environment ~ ' \l__cmd_environment_str ' }
      {
        command ~
          ' \c_backslash_str \tl_to_str:N \l__cmd_function_tl '
      }
  }
```

(*End of definition for* `\__cmd_environment_or_command:`.)

Some messages intended as errors when defining commands/environments.

```
\msg_new:nnnn { cmd } { arg-after-body }
  { Argument~type~'b'~must~be~last~in~#1. }
  {
    The~'b'~argument~type~must~come~last~but~it~is~followed~
    by~'#2'~in~the~argument~specification.~This~is~not~allowed.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { bad-arg-spec }
  { Bad~argument~specification~'#2'~for~#1. }
  {
    The~argument~specification~provided~is~not~valid:~
    one~or~more~mandatory~parts~are~missing.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { already-defined }
  { Command~'#1'~already~defined. }
  {
    You~have~used~#2~
    with~a~command~that~already~has~a~definition. \\ \\
    The~existing~definition~of~'#1'~will~not~be~altered.
  }
\msg_new:nnnn { cmd } { undefined }
  { Command ~'#1'~undefined. }
  {
    You~have~used~#2~
    with~a~command~that~was~never~defined.
    \c__cmd_ignore_def_tl
```

```
  }
\msg_new:nnnn { cmd } { env-already-defined }
  { Environment~'#1'~already~defined. }
  {
    You~have~used~\NewDocumentEnvironment
    with~an~environment~that~already~has~a~definition. \\ \\
    The~existing~definition~of~'#1'~will~not~be~altered.
  }
\msg_new:nnnn { cmd } { env-end-already-defined }
  { End~of~environment~'#1'~already~defined. }
  {
    You~have~used~\NewDocumentEnvironment
    with~an~environment~that~already~has~a~definition~for~'end#1'. \\ \\
    The~existing~definition~of~'#1'~will~not~be~altered.
  }
\msg_new:nnnn { cmd } { env-undefined }
  { Environment~'#1'~undefined. }
  {
    You~have~used~\RenewDocumentEnvironment
    with~an~environment~that~was~never~defined.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { expandable-ending-optional }
  { Bad~argument~specification~'#2'~for~#1. }
  {
    Expandable~commands~must~have~a~final~mandatory~argument~
    (or~no~arguments~at~all).~You~cannot~have~a~terminal~optional~
    argument~with~expandable~commands.
  }
\msg_new:nnnn { cmd } { long-short-mix }
  { Invalid~argument~prefix~'+'~in~command~'#1'. }
  {
    The~arguments~for~an~expandable~command~must~not~involve~short~
    arguments~after~long~arguments.~You~have~tried~to~mix~the~two~types~
    when~defining~'#1'.
  }
\msg_new:nnnn { cmd } { invalid-command-arg }
  { Invalid~argument~type~'#2'~in~#1. }
  {
    The~letter~'#2'~can~only~be~used~in~environment~argument~
    specifications,~but~not~for~commands.
    \\ \\
    LaTeX~will~ignore~the~entire~definition.
  }
\msg_new:nnnn { cmd } { invalid-expandable-arg }
  { Invalid~argument~type~'#2'~in~#1. }
  {
    The~letter~'#2'~specifies~an~argument~type~which~cannot~be~used~
    in~an~expandable~command.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { invalid-after-optional-expandably }
  { Argument~'#2'~invalid~after~optional~arg~in~#1. }
  {
```

```
    The~letter~'#2'~specifies~an~argument~type~which~cannot~be~used~
    in~an~expandable~command~after~an~optional~argument.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { invalid-bang }
  { Invalid~argument~prefix~'!'~in~#1. }
  {
    The~prefix~'!'~is~only~allowed~for~trailing~optional~arguments.~
    You~tried~to~apply~it~to~#2.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { not-definable }
  { First~argument~of~'#2'~must~be~a~command. }
  {
    The~first~argument~of~'#2'~should~be~the~document~command~that~will~
    be~defined.~The~provided~argument~'#1'~is~a~character.~Perhaps~a~
    backslash~is~missing?
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { not-one-token }
  { First~argument~of~'#2'~must~be~a~command. }
  {
    The~first~argument~of~'#2'~should~be~the~document~command~that~will~
    be~defined.~The~provided~argument~'#1'~contains~more~than~one~
    token.~Perhaps~a~backslash~is~missing?
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { not-single-token }
  { Argument~delimiter~'#2'~invalid~in~#1. }
  {
    The~argument~specification~contains~
    \tl_if_empty:nTF{#2}{nothing}{'#2'}~
    in~a~place~
    where~a~single~token~is~required.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { forbidden-group-token }
  { Argument~delimiter~'#2'~invalid~in~#1. }
  {
    The~argument~specification~contains~the~implicit~
    #3-group~token~'#2'~which~is~not~allowed~as~an~argument~delimiter.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { processor-in-expandable }
  { Invalid~argument~prefix~'>'~in~command~'#1'. }
  {
    The~argument~specification~for~'#1'~contains~the~processor~function~'>{#2}'.~
    This~is~only~supported~for~robust~commands,~but~not~for~expandable~ones.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { keyval-in-expandable }
  { Invalid~argument~prefix~'='~in~command~'#1'. }
  {
    The~argument~specification~for~'#1'~contains~a~key--value~marker~'={#2}'.~
```

```
    This~is~only~supported~for~robust~commands,~but~not~for~expandable~ones.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { too-many-args }
  { Too~many~arguments~for~#1. }
  {
    The~argument~specification~'#2'~asks~for~more~than~9~arguments.~
    This~cannot~be~implemented.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { two-markers }
  { Invalid~argument~prefix~'#2'~in~#1. }
  {
    The~argument~specification~provided~for~#1~has~two~'#2'~markers~applied~
    to~the~same~argument;~one~is~redundant.
  }
\msg_new:nnnn { cmd } { unknown-argument-type } % should be unkown-arg-type but dep in xparse
  { Invalid~argument~type~'#2'~in~#1. }
  {
    The~letter~'#2'~does~not~specify~a~known~argument~type.
    \c__cmd_ignore_def_tl
  }
\msg_new:nnnn { cmd } { xparse-arg-type }
  { Invalid~argument~type~'#2'~in~#1~(requires~xparse). }
  {
    The~letter~'#2'~specifies~a~known~but~deprecated~argument~type.~
    If~you~really~need~it~you~have~to~load~the~xparse~package.
    \c__cmd_ignore_def_tl
  }
```

Errors when using commands/environments. The `if-boolean` message is always used in expandable errors. The `default-loop` and `missing-required` messages can be expandable or not expandable.

```
\msg_new:nnn { cmd } { if-boolean }
  { Invalid~argument~{#1}~to~\iow_char:N\\IfBoolean... }
\msg_new:nnnn { cmd } { default-loop }
  { Circular~dependency~in~defaults~of~#1. }
  {
    The~default~values~of~two~or~more~arguments~of~the~#1~
    depend~on~each~other~in~a~way~that~cannot~be~resolved.
  }
\msg_new:nnnn { cmd } { missing-required }
  { Required~argument~missing~for~#1. }
  {
    The~#1~expects~one~of~its~arguments~to~start~with~'#2'.~
    LaTeX~did~not~find~this~argument~and~will~insert~a~default~value~
    for~further~processing.
  }
\msg_new:nnnn { cmd } { arg-split }
  { Too~many~'#1'~separators~in~argument. }
  {
    LaTeX~was~asked~to~split~the~input~'#3'~
    at~each~occurrence~of~the~separator~'#1'~into~#2~parts.~
    Too~many~separators~were~found.
```

```
  }
\msg_new:nnnn { cmd } { verbatim-nl }
  { Verbatim-like~#1~ended~by~end~of~line. }
  {
    The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
    but~the~end~
    of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
    closing~delimiter.
    \\ \\
    LaTeX~will~ignore~'#2'~and~you~may~get~some~additional~
    (low-level)~errors.
  }
\msg_new:nnnn { cmd } { verbatim-tokenized }
  { Verbatim-like~#1~illegal~in~argument. }
  {
    The~#1~takes~a~verbatim~argument~and~should~therefore~normally~
    not~be~used~in~arguments~of~other~commands~or~environments.~
    LaTeX~found~an~illegal~token~ \tl_if_empty:nF {#3} { (#3)~ }
    after~'#2'~and~will~drop~everything~up~to~this~point.
    \\ \\
    Expect~further~(low-level)~errors.
  }
```

Intended more for information.

```
\msg_new:nnn { cmd } { define-command }    % should be just ``define'' but dep in xparse
  {
    Defining~command~#1~
    with~sig.~'#2'~\msg_line_context:.
  }
\msg_new:nnn { cmd } { define-env }
  {
    Defining~environment~'#1'~
    with~sig.~'#2'~\msg_line_context:.
  }
\msg_new:nnn { cmd } { redefine }
  {
    Redefining~command~#1~
    with~sig.~'#2'~\msg_line_context:.
  }
\msg_new:nnn { cmd } { redefine-env }
  {
    Redefining~environment~'#1'~
    with~sig.~'#2'~\msg_line_context:.
  }
\msg_new:nnn { cmd } { optional-mandatory }
  {
    Optional~and~mandatory~argument~with~same~delimiter~'#2'.
    \\ \\
    The~mandatory~argument~specified~with~
    '\str_case:nnF{#1}{ {R/r}{r'~or~'R} }{#1}'~has~the~
    same~delimiter~'#2'~as~an~earlier~optional~argument.~
    It~will~therefore~not~be~possible~to~omit~all~the~earlier~
    optional~arguments~when~calling~this~command.
    \\ \\
    This~may~be~intentional,~but~then~it~might~be~a~mistake.
```

```
  }
\msg_new:nnn { cmd } { unsupported-let }
  {
    The~command~'#1'~was~undefined~but~not~the~associated~commands~
    '#1~code'~and/or~'#1~defaults'.~Maybe~you~tried~using~
    \iow_char:N\\let.~This~may~lead~to~an~infinite~loop.
  }
```

1.14 User functions

The user functions are more or less just the internal functions renamed.

\BooleanFalse
\BooleanTrue

Design-space names for the Boolean values.

```
\cs_new_eq:NN \BooleanFalse \c_false_bool
\cs_new_eq:NN \BooleanTrue  \c_true_bool
```

(*End of definition for* \BooleanFalse *and* \BooleanTrue.)

\NewDocumentCommand
\RenewDocumentCommand
\ProvideDocumentCommand
\DeclareDocumentCommand

The user macros are pretty simple wrappers around the internal ones. There is however a check that the first argument is a single token, possibly surrounded by spaces (hence the strange \use:nnn), and is definable.

```
\cs_new_protected:Npn \NewDocumentCommand #1#2#3
  {
    \__cmd_check_definable:nNT {#1} \NewDocumentCommand
      {
        \cs_if_exist:NTF #1
          {
            \msg_error:nnxx { cmd } { already-defined }
              { \use:nnn \token_to_str:N #1 { } }
              { \token_to_str:N \NewDocumentCommand }
          }
          { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
      }
  }
\cs_new_protected:Npn \RenewDocumentCommand #1#2#3
  {
    \__cmd_check_definable:nNT {#1} \RenewDocumentCommand
      {
        \cs_if_exist:NTF #1
          { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
          {
            \msg_error:nnxx { cmd } { undefined }
              { \use:nnn \token_to_str:N #1 { } }
              { \token_to_str:N \RenewDocumentCommand }
          }
      }
  }
\cs_new_protected:Npn \ProvideDocumentCommand #1#2#3
  {
    \__cmd_check_definable:nNT {#1} \ProvideDocumentCommand
      { \cs_if_exist:NF #1 { \__cmd_declare_cmd:Nnn #1 {#2} {#3} } }
 }
\cs_new_protected:Npn \DeclareDocumentCommand #1#2#3
  {
```

```
    \__cmd_check_definable:nNT {#1} \DeclareDocumentCommand
      { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
  }
```

(*End of definition for* `\NewDocumentCommand` *and others.*)

`\NewDocumentEnvironment`
`\RenewDocumentEnvironment`
`\ProvideDocumentEnvironment`
`\DeclareDocumentEnvironment`
`\__cmd_new_env:nnnn`
`\__cmd_renew_env:nnnn`
`\__cmd_provide_env:nnnn`
`\__cmd_new_env:ennn`
`\__cmd_renew_env:ennn`
`\__cmd_provide_env:ennn`

Very similar for environments. Trim spaces from user-specified ⟨*envname*⟩, do existence check then hand off to `\__cmd_declare_env:nnnn`.

```
⟨latexrelease⟩\IncludeInRelease{2024/11/01}{\NewDocumentEnvironment}%
⟨latexrelease⟩                  {Trim~spaces~from~envname~first}
\cs_new_protected:Npn \NewDocumentEnvironment #1#2#3#4
  {
    \__cmd_new_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
  }
\cs_new_protected:Npn \RenewDocumentEnvironment #1#2#3#4
  {
    \__cmd_renew_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
  }
\cs_new_protected:Npn \ProvideDocumentEnvironment #1#2#3#4
  {
    \__cmd_provide_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
  }
\cs_new_protected:Npn \DeclareDocumentEnvironment #1#2#3#4
  {
    \__cmd_declare_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
  }
```

Each of `\__cmd_(new|renew|provide)_env:nnnn` is curried.

```
\cs_new_protected:Npn \__cmd_new_env:nnnn #1
  {
    \cs_if_exist:cTF {#1}
      {
        \msg_error:nnx { cmd } { env-already-defined } {#1}
        \use_none:nnn
      }
      {
        \cs_if_exist:cTF { end #1 }
          {
            \msg_error:nnx { cmd } { env-end-already-defined } {#1}
            \use_none:nnn
          }
          { \__cmd_declare_env:nnnn {#1} }
      }
  }
\cs_new_protected:Npn \__cmd_renew_env:nnnn #1
  {
    \cs_if_exist:cTF {#1}
      { \__cmd_declare_env:nnnn {#1} }
      {
        \msg_error:nnx { cmd } { env-undefined } {#1}
        \use_none:nnn
      }
  }
\cs_new_protected:Npn \__cmd_provide_env:nnnn #1
  {
```

```
    \cs_if_exist:cTF {#1}
      { \use_none:nnn }
      { \__cmd_declare_env:nnnn {#1} }
  }
\cs_generate_variant:Nn \__cmd_new_env:nnnn { e }
\cs_generate_variant:Nn \__cmd_renew_env:nnnn { e }
\cs_generate_variant:Nn \__cmd_provide_env:nnnn { e }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2024/06/01}{\NewDocumentEnvironment}%
⟨latexrelease⟩                 {Trim~spaces~from~envname~first}
⟨latexrelease⟩\cs_new_protected:Npn \NewDocumentEnvironment #1#2#3#4
⟨latexrelease⟩  {
⟨latexrelease⟩    \cs_if_exist:cTF {#1}
⟨latexrelease⟩      { \msg_error:nnx { cmd } { env-already-defined } {#1} }
⟨latexrelease⟩      {
⟨latexrelease⟩        \cs_if_exist:cTF { end #1 }
⟨latexrelease⟩          { \msg_error:nnx { cmd } { env-end-already-defined } {#1} }
⟨latexrelease⟩          { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new_protected:Npn \RenewDocumentEnvironment #1#2#3#4
⟨latexrelease⟩  {
⟨latexrelease⟩    \cs_if_exist:cTF {#1}
⟨latexrelease⟩      { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
⟨latexrelease⟩      { \msg_error:nnx { cmd } { env-undefined } {#1} }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new_protected:Npn \ProvideDocumentEnvironment #1#2#3#4
⟨latexrelease⟩  { \cs_if_exist:cF {#1} { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} } }
⟨latexrelease⟩\cs_new_protected:Npn \DeclareDocumentEnvironment #1#2#3#4
⟨latexrelease⟩  { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
⟨latexrelease⟩\cs_undefine:N \__cmd_new_env:nnnn
⟨latexrelease⟩\cs_undefine:N \__cmd_new_env:ennn
⟨latexrelease⟩\cs_undefine:N \__cmd_renew_env:nnnn
⟨latexrelease⟩\cs_undefine:N \__cmd_renew_env:ennn
⟨latexrelease⟩\cs_undefine:N \__cmd_provide_env:nnnn
⟨latexrelease⟩\cs_undefine:N \__cmd_provide_env:ennn
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\NewDocumentEnvironment` *and others.*)

`\NewExpandableDocumentCommand`
`\RenewExpandableDocumentCommand`
`\ProvideExpandableDocumentCommand`
`\DeclareExpandableDocumentCommand`

The expandable versions are essentially the same as the basic functions. The strange `\use:nnn` is there in case `#1` is surrounded with spaces, as can happen with usual document catcodes in `\RenewExpandableDocumentCommand { \! }` ...

```
\cs_new_protected:Npn \NewExpandableDocumentCommand #1#2#3
  {
    \__cmd_check_definable:nNT {#1} \NewExpandableDocumentCommand
      {
        \cs_if_exist:NTF #1
          {
            \msg_error:nnxx { cmd } { already-defined }
              { \use:nnn \token_to_str:N #1 { } }
              { \token_to_str:N \NewExpandableDocumentCommand }
          }
```

```
          { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
      }
  }
\cs_new_protected:Npn \RenewExpandableDocumentCommand #1#2#3
  {
    \__cmd_check_definable:nNT {#1} \RenewExpandableDocumentCommand
      {
        \cs_if_exist:NTF #1
          { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
          {
            \msg_error:nnxx { cmd } { undefined }
              { \use:nnn \token_to_str:N #1 { } }
              { \token_to_str:N \RenewExpandableDocumentCommand }
          }
      }
  }
\cs_new_protected:Npn \ProvideExpandableDocumentCommand #1#2#3
  {
    \__cmd_check_definable:nNT {#1} \ProvideExpandableDocumentCommand
      {
        \cs_if_exist:NF #1
          { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
      }
 }
\cs_new_protected:Npn \DeclareExpandableDocumentCommand #1#2#3
  {
    \__cmd_check_definable:nNT {#1} \DeclareExpandableDocumentCommand
      { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
  }
```

(*End of definition for* `\NewExpandableDocumentCommand` *and others.*)

`\IfBooleanT` `\IfBooleanF` `\IfBooleanTF`

The logical ⟨*true*⟩ and ⟨*false*⟩ statements are just the normal `\c_true_bool` and `\c_false_bool` so `\bool_if:NTF` is almost enough. However, this code-level function blows up badly when passed invalid input. We want `\IfBooleanTF` to accept a single (non-space) token equal to `\c_true_bool` or `\c_false_bool`, possibly surrounded by spaces. If the input is blank or multiple items, jump to the error and pick the false branch. If the input, ignoring spaces (we do this by omitting braces in the `\tl_if_single_token:nF` test), is not a single token then jump to the error as well. It is then safe to compare the token to the two booleans, picking the appropriate branch. If neither matches, we jump to the error as well.

```
\cs_new:Npn \IfBooleanTF #1
  {
    \tl_if_single:nF {#1}
      { \prg_break:n { \use:n } }
    \tl_if_single_token:nF #1
      { \prg_break:n { \use:n } }
    \token_if_eq_meaning:NNT #1 \c_true_bool
      { \prg_break:n { \use_ii:nnn } }
    \token_if_eq_meaning:NNT #1 \c_false_bool
      { \prg_break:n { \use_iii:nnn } }
    \prg_break:n { \use:n }
    \prg_break_point:
    {
```

```
      \msg_expandable_error:nnn { cmd } { if-boolean } {#1}
      \use_ii:nn
    }
  }
\cs_new:Npn \IfBooleanT #1#2 { \IfBooleanTF {#1} {#2} { } }
\cs_new:Npn \IfBooleanF #1 { \IfBooleanTF {#1} { } }
```

(*End of definition for* `\IfBooleanT`, `\IfBooleanF`, *and* `\IfBooleanTF`.)

`\IfNoValueT` `\IfNoValueF` `\IfNoValueTF`

Simple re-naming.

```
\cs_new_eq:NN \IfNoValueF  \tl_if_novalue:nF
\cs_new_eq:NN \IfNoValueT  \tl_if_novalue:nT
\cs_new_eq:NN \IfNoValueTF \tl_if_novalue:nTF
```

(*End of definition for* `\IfNoValueT`, `\IfNoValueF`, *and* `\IfNoValueTF`.)

`\IfValueT` `\IfValueF` `\IfValueTF`

Inverted logic.

```
\cs_new:Npn \IfValueF { \tl_if_novalue:nT }
\cs_new:Npn \IfValueT { \tl_if_novalue:nF }
\cs_new:Npn \IfValueTF #1#2#3 { \tl_if_novalue:nTF {#1} {#3} {#2} }
```

(*End of definition for* `\IfValueT`, `\IfValueF`, *and* `\IfValueTF`.)

`\IfBlankT` `\IfBlankF` `\IfBlankTF`

Another simple re-naming.

```
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                 {\IfBlankTF}{Testing~for~empty~or~blank}%
\cs_new_eq:NN \IfBlankF  \tl_if_blank:nF
\cs_new_eq:NN \IfBlankT  \tl_if_blank:nT
\cs_new_eq:NN \IfBlankTF \tl_if_blank:nTF
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                 {\IfBlankTF}{Testing~for~empty~or~blank}%
⟨latexrelease⟩\cs_undefine:N \IfBlankF
⟨latexrelease⟩\cs_undefine:N \IfBlankT
⟨latexrelease⟩\cs_undefine:N \IfBlankTF
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\IfBlankT`, `\IfBlankF`, *and* `\IfBlankTF`.)

`\ProcessedArgument`

Processed arguments are returned using this name, which is reserved here although the definition will change.

```
\tl_new:N \ProcessedArgument
```

(*End of definition for* `\ProcessedArgument`.)

`\ReverseBoolean` `\SplitArgument` `\SplitList` `\TrimSpaces`

Simple copies.

```
\cs_new_eq:NN \ReverseBoolean \__cmd_bool_reverse:N
\cs_new_eq:NN \SplitArgument  \__cmd_split_argument:nnn
\cs_new_eq:NN \SplitList      \__cmd_split_list:nn
\cs_new_eq:NN \TrimSpaces     \__cmd_trim_spaces:n
```

(*End of definition for* `\ReverseBoolean` *and others.*)

`\ProcessList`

To support `\SplitList`.

```
\cs_new_eq:NN \ProcessList \tl_map_function:nN
```

(*End of definition for* `\ProcessList`.)

Finally as promised, restore `\__kernel_chk_if_free_cs:N`:

```
⟨latexrelease⟩\cs_gset_eq:NN \__kernel_chk_if_free_cs:N \__cmd_chk_if_free_cs:N
⟨latexrelease⟩\cs_undefine:N \__cmd_chk_if_free_cs:N
```

```
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{ltcmd}%
⟨latexrelease⟩                  {Document~command~parser}%
⟨latexrelease⟩
⟨latexrelease⟩\EndModuleRelease
\ExplSyntaxOff
```

Now in `latexrelease` mode, redefine `\NewDocumentCommand` to not complain on commands already defined.

```
⟨latexrelease⟩\@ifundefined{ExplSyntaxOff}{}{\latexrelease@postltcmd}
⟨latexrelease⟩\catcode`\^^@=\@latexrelease@catcode@null\relax
⟨/2ekernel | latexrelease⟩
```

We need to stop DocStrip treating `@@` in a special way at this point.

```
⟨@@=⟩
```

File 08
lthooks.dtx

1 Introduction

Hooks are points in the code of commands or environments where it is possible to add processing code into existing commands. This can be done by different packages that do not know about each other and to allow for hopefully safe processing it is necessary to sort different chunks of code added by different packages into a suitable processing order.

This is done by the packages adding chunks of code (via `\AddToHook`) and labeling their code with some label by default using the package name as a label.

At `\begin{document}` all code for a hook is then sorted according to some rules (given by `\DeclareHookRule`) for fast execution without processing overhead. If the hook code is modified afterwards (or the rules are changed), a new version for fast processing is generated.

Some hooks are used already in the preamble of the document. If that happens then the hook is prepared for execution (and sorted) already at that point.

2 Package writer interface

The hook management system is offered as a set of CamelCase commands for traditional LaTeX 2_ε packages (and for use in the document preamble if needed) as well as `expl3` commands for modern packages, that use the L3 programming layer of LaTeX. Behind the scenes, a single set of data structures is accessed so that packages from both worlds can coexist and access hooks in other packages.

2.1 LaTeX 2_ε interfaces

2.1.1 Declaring hooks

With a few exceptions, hooks have to be declared before they can be used. The exceptions are the generic hooks for commands and environments (executed at `\begin` and `\end`), and the hooks run when loading files (see section 3.1).

`\NewHook` `\NewHook {⟨hook⟩}`

Creates a new ⟨*hook*⟩. If this hook is declared within a package it is suggested that its name is always structured as follows: ⟨*package-name*⟩/⟨*hook-name*⟩. If necessary you can further subdivide the name by adding more `/` parts. If a hook name is already taken, an error is raised and the hook is not created.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5. The string `??` can't be used as a hook name because it has a special significance as a placeholder in hook rules.

\NewReversedHook `\NewReversedHook {⟨hook⟩}`

Like \NewHook declares a new ⟨*hook*⟩. the difference is that the code chunks for this hook are in reverse order by default (those added last are executed first). Any rules for the hook are applied after the default ordering. See sections 2.3 and 2.4 for further details.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\NewMirroredHookPair `\NewMirroredHookPair {⟨hook-1⟩} {⟨hook-2⟩}`

A shorthand for \NewHook{⟨*hook-1*⟩}\NewReversedHook{⟨*hook-2*⟩}.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\NewHookWithArguments `\NewHookWithArguments {⟨hook⟩} {⟨number⟩}`

New: 2023-06-01

Creates a new ⟨*hook*⟩ whose code takes ⟨*number*⟩ arguments, and otherwise works exactly like \NewHook. Section 2.7 explains hooks with arguments.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\NewReversedHookWithArguments `\NewReversedHookWithArguments {⟨hook⟩} {⟨number⟩}`

New: 2023-06-01

Like \NewReversedHook, but creates a hook whose code takes ⟨*number*⟩ arguments. Section 2.7 explains hooks with arguments.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\NewMirroredHookPairWithArguments `\NewMirroredHookPairWithArguments {⟨hook-1⟩} {⟨hook-2⟩} {⟨number⟩}`

New: 2023-06-01

A shorthand for \NewHookWithArguments{⟨*hook-1*⟩}{⟨*number*⟩} \NewReversedHookWithArguments{⟨*hook-2*⟩}{⟨*number*⟩}. Section 2.7 explains hooks with arguments.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.2 Special declarations for generic hooks

The declarations here should normally not be used. They are available to provide support for special use cases mainly involving generic command hooks.

\DisableGenericHook `\DisableGenericHook {⟨hook⟩}`

After this declaration[3] the ⟨*hook*⟩ is no longer usable: Any further attempt to add code to it will result in an error and any use, e.g., via \UseHook, will simply do nothing.

This is intended to be used with generic command hooks (see ltcmdhooks-doc) as depending on the definition of the command such generic hooks may be unusable. If that is known, a package developer can disable such hooks up front.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

[3] In the 2020/06 release this command was called \DisableHook, but that name was misleading as it shouldn't be used to disable non-generic hooks.

\ActivateGenericHook `\ActivateGenericHook {⟨hook⟩}`

This declaration activates a generic hook provided by a package/class (e.g., one used in code with `\UseHook` or `\UseOneTimeHook`) without it being explicitly declared with `\NewHook`). If the hook is already activated, this command does nothing.

Note that this command does not undo the effect of `\DisableGenericHook`. See section 2.6 for a discussion of when this declaration is appropriate.

2.1.3 Using hooks in code

\UseHook `\UseHook {⟨hook⟩}`

Execute the code stored in the ⟨*hook*⟩.

Before `\begin{document}` the fast execution code for a hook is not set up, so in order to use a hook there it is explicitly initialized first. As that involves assignments using a hook at those times is not 100% the same as using it after `\begin{document}`.

The ⟨*hook*⟩ *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

\UseHookWithArguments `\UseHookWithArguments {⟨hook⟩} {⟨number⟩} {⟨arg$_1$⟩} ... {⟨arg$_n$⟩}`

New: 2023-06-01

Execute the code stored in the ⟨*hook*⟩ and pass the arguments {⟨arg_1⟩} through {⟨arg_n⟩} to the ⟨*hook*⟩. Otherwise, it works exactly like `\UseHook`. The ⟨*number*⟩ should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove ⟨*number*⟩ items from the input. Section 2.7 explains hooks with arguments.

The ⟨*hook*⟩ *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

\UseOneTimeHook `\UseOneTimeHook {⟨hook⟩}`

Some hooks are only used (and can be only used) in one place, for example, those in `\begin{document}` or `\end{document}`. From that point onwards, adding to the hook through a defined `\`⟨*addto-cmd*⟩ command (e.g., `\AddToHook` or `\AtBeginDocument`, etc.) would have no effect (as would the use of such a command inside the hook code itself). It is therefore customary to redefine `\`⟨*addto-cmd*⟩ to simply process its argument, i.e., essentially make it behave like `\@firstofone`.

`\UseOneTimeHook` does that: it records that the hook has been consumed and any further attempt to add to it will result in executing the code to be added immediately.

Using `\UseOneTimeHook` several times with the same {⟨*hook*⟩} means that it only executes the first time it is used. For example, if it is used in a command that can be called several times then the hook executes during only the *first* invocation of that command; this allows its use as an "initialization hook".

Mixing `\UseHook` and `\UseOneTimeHook` for the same {⟨*hook*⟩} should be avoided, but if this is done then neither will execute after the first `\UseOneTimeHook`.

The ⟨*hook*⟩ *cannot* be specified using the dot-syntax. A leading `.` is treated literally. See section 2.1.5 for details.

`\UseOneTimeHookWithArguments` `\UseOneTimeHookWithArguments {⟨hook⟩} {⟨number⟩} {⟨arg$_1$⟩} ... {⟨arg$_n$⟩}`

New: 2023-06-01

Works exactly like `\UseOneTimeHook`, but passes arguments `{⟨arg$_1$⟩}` through `{⟨arg$_n$⟩}` to the ⟨*hook*⟩. The ⟨*number*⟩ should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove ⟨*number*⟩ items from the input.

It should be noted that after a one-time hook is used, it is no longer possible to use `\AddToHookWithArguments` or similar with that hook. `\AddToHook` continues to work as normal. Section 2.7 explains hooks with arguments.

The ⟨*hook*⟩ *cannot* be specified using the dot-syntax. A leading `.` is treated literally. See section 2.1.5 for details.

2.1.4 Updating code for hooks

`\AddToHook` `\AddToHook {⟨hook⟩} [⟨label⟩] {⟨code⟩}`

Adds ⟨*code*⟩ to the ⟨*hook*⟩ labeled by ⟨*label*⟩. When the optional argument ⟨*label*⟩ is not provided, the ⟨*default label*⟩ is used (see section 2.1.5). If `\AddToHook` is used in a package/class, the ⟨*default label*⟩ is the package/class name, otherwise it is `top-level` (the `top-level` label is treated differently: see section 2.1.6).

If there already exists code under the ⟨*label*⟩ then the new ⟨*code*⟩ is appended to the existing one (even if this is a reversed hook). If you want to replace existing code under the ⟨*label*⟩, first apply `\RemoveFromHook`.

The hook doesn't have to exist for code to be added to it. However, if it is not declared, then obviously the added ⟨*code*⟩ will never be executed. This allows for hooks to work regardless of package loading order and enables packages to add to hooks from other packages without worrying whether they are actually used in the current document. See section 2.1.8.

The ⟨*hook*⟩ and ⟨*label*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\AddToHookWithArguments` `\AddToHookWithArguments {⟨hook⟩} [⟨label⟩] {⟨code⟩}`

New: 2023-06-01

Works exactly like `\AddToHook`, except that the ⟨*code*⟩ can access the arguments passed to the hook using `#1`, `#2`, ..., `#n` (up to the number of arguments declared for the hook). If the ⟨*code*⟩ should contain *parameter tokens* (`#`) that are not supposed to be understood as the arguments of the hook, such tokens should be doubled. For example, with `\AddToHook` one can write:

```
\AddToHook{myhook}{\def\foo#1{Hello, #1!}}
```

but to achieve the same with `\AddToHookWithArguments`, one should write:

```
\AddToHookWithArguments{myhook}{\def\foo##1{Hello, ##1!}}
```

because in the latter case, `#1` refers to the first argument of the hook `myhook`. Section 2.7 explains hooks with arguments.

The ⟨*hook*⟩ and ⟨*label*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\RemoveFromHook

\RemoveFromHook {⟨*hook*⟩} [⟨*label*⟩]

Removes any code labeled by ⟨*label*⟩ from the ⟨*hook*⟩. When the optional argument ⟨*label*⟩ is not provided, the ⟨*default label*⟩ is used (see section 2.1.5).

If there is no code under the ⟨*label*⟩ in the ⟨*hook*⟩, or if the ⟨*hook*⟩ does not exist, a warning is issued when you attempt to \RemoveFromHook, and the command is ignored. \RemoveFromHook should be used only when you know exactly what labels are in a hook. Typically this will be when some code gets added to a hook by a package, then later this code is removed by that same package. If you want to prevent the execution of code from another package, use the voids rule instead (see section 2.1.7).

Important: *The* \RemoveFromHook *command should be only used if one has full control over the code chunk to be removed. In particular it should not be used to remove code chunks from other packages! For this the* voids *relation is provided.*

If the optional ⟨*label*⟩ argument is *, then all code chunks are removed. This is rather dangerous as it may well drop code from other packages (that one may not know about); it should therefore not be used in packages but only in document preambles!

The ⟨*hook*⟩ and ⟨*label*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

In contrast to the voids relationship between two labels in a \DeclareHookRule this is a destructive operation as the labeled code is removed from the hook data structure, whereas the relationship setting can be undone by providing a different relationship later.

A useful application for this declaration inside the document body is when one wants to temporarily add code to hooks and later remove it again, e.g.,

```
\AddToHook{env/quote/before}{\small}
\begin{quote}
  A quote set in a smaller typeface
\end{quote}
...
\RemoveFromHook{env/quote/before}
... now back to normal for further quotes
```

Note that you can't cancel the setting with

```
\AddToHook{env/quote/before}{}
```

because that only "adds" a further empty chunk of code to the hook. Adding \normalsize would work but that means the hook then contained \small\normalsize which means two font size changes for no good reason.

The above is only needed if one wants to typeset several quotes in a smaller typeface. If the hook is only needed once then \AddToHookNext is simpler, because it resets itself after one use.

`\AddToHookNext` `\AddToHookNext {⟨hook⟩} {⟨code⟩}`

Adds ⟨*code*⟩ to the next invocation of the ⟨*hook*⟩. The code is executed after the normal hook code has finished and it is executed only once, i.e. it is deleted after it was used.

Using this declaration is a global operation, i.e., the code is not lost even if the declaration is used inside a group and the next invocation of the hook happens after the end of that group. If the declaration is used several times before the hook is executed then all code is executed in the order in which it was declared.[4]

If this declaration is used with a one-time hook then the code is only ever used if the declaration comes before the hook's invocation. This is because, in contrast to `\AddToHook`, the code in this declaration is not executed immediately in the case when the invocation of the hook has already happened—in other words, this code will truly execute only on the next invocation of the hook (and in the case of a one-time hook there is no such "next invocation"). This gives you a choice: should my code execute always, or should it execute only at the point where the one-time hook is used (and not at all if this is impossible)? For both of these possibilities there are use cases.

It is possible to nest this declaration using the same hook (or different hooks): e.g.,

`\AddToHookNext{`⟨*hook*⟩`}{`⟨*code-1*⟩`\AddToHookNext{`⟨*hook*⟩`}{`⟨*code-2*⟩`}}`

will execute ⟨*code-1*⟩ next time the ⟨*hook*⟩ is used and at that point puts ⟨*code-2*⟩ into the ⟨*hook*⟩ so that it gets executed on following time the hook is run.

A hook doesn't have to exist for code to be added to it. This allows for hooks to work regardless of package loading order. See section 2.1.8.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\AddToHookNextWithArguments` `\AddToHookNextWithArguments {⟨hook⟩} {⟨code⟩}`

New: 2023-06-01

Works exactly like `\AddToHookNext`, but the ⟨*code*⟩ can contain references to the arguments of the ⟨*hook*⟩ as described for `\AddToHookWithArguments` above. Section 2.7 explains hooks with arguments.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\ClearHookNext` `\ClearHookNext {⟨hook⟩}`

Normally `\AddToHookNext` is only used when you know precisely where it will apply and why you want some extra code at that point. However, there are a few use cases in which such a declaration needs to be canceled, for example, when discarding a page with `\DiscardShipoutBox` (but even then not always), and in such situations `\ClearHookNext` can be used.

2.1.5 Hook names and default labels

It is best practice to use `\AddToHook` in packages or classes *without specifying a* ⟨*label*⟩ because then the package or class name is automatically used, which is helpful if rules are needed, and avoids mistyping the ⟨*label*⟩.

[4] There is no mechanism to reorder such code chunks (or delete them).

Using an explicit ⟨*label*⟩ is only necessary in very specific situations, e.g., if you want to add several chunks of code into a single hook and have them placed in different parts of the hook (by providing some rules).

The other case is when you develop a larger package with several sub-packages. In that case you may want to use the same ⟨*label*⟩ throughout the sub-packages in order to avoid that the labels change if you internally reorganize your code.

Except for `\UseHook`, `\UseOneTimeHook` and `\IfHookEmptyTF` (and their expl3 interfaces `\hook_use:n`, `\hook_use_once:n` and `\hook_if_empty:nTF`), all ⟨*hook*⟩ and ⟨*label*⟩ arguments are processed in the same way: first, spaces are trimmed around the argument, then it is fully expanded until only character tokens remain. If the full expansion of the ⟨*hook*⟩ or ⟨*label*⟩ contains a non-expandable non-character token, a low-level TEX error is raised (namely, the ⟨*hook*⟩ is expanded using TEX's `\csname`...`\endcsname`, as such, Unicode characters are allowed in ⟨*hook*⟩ and ⟨*label*⟩ arguments). The arguments of `\UseHook`, `\UseOneTimeHook`, and `\IfHookEmptyTF` are processed much in the same way except that spaces are not trimmed around the argument, for better performance.

It is not enforced, but highly recommended that the hooks defined by a package, and the ⟨*labels*⟩ used to add code to other hooks contain the package name to easily identify the source of the code chunk and to prevent clashes. This should be the standard practice, so this hook management code provides a shortcut to refer to the current package in the name of a ⟨*hook*⟩ and in a ⟨*label*⟩. If the ⟨*hook*⟩ name or the ⟨*label*⟩ consist just of a single dot (`.`), or starts with a dot followed by a slash (`./`) then the dot denotes the ⟨*default label*⟩ (usually the current package or class name—see `\SetDefaultHookLabel`). A "`.`" or "`./`" anywhere else in a ⟨*hook*⟩ or in ⟨*label*⟩ is treated literally and is not replaced.

For example, inside the package `mypackage.sty`, the default label is `mypackage`, so the instructions:

```
\NewHook   {./hook}
\AddToHook {./hook}[.]{code}     % Same as \AddToHook{./hook}{code}
\AddToHook {./hook}[./sub]{code}
\DeclareHookRule{begindocument}{.}{before}{babel}
\AddToHook {file/foo.tex/after}{code}
```

are equivalent to:

```
\NewHook   {mypackage/hook}
\AddToHook {mypackage/hook}[mypackage]{code}
\AddToHook {mypackage/hook}[mypackage/sub]{code}
\DeclareHookRule{begindocument}{mypackage}{before}{babel}
\AddToHook {file/foo.tex/after}{code}                  % unchanged
```

The ⟨*default label*⟩ is automatically set equal to the name of the current package or class at the time the package is loaded. If the hook command is used outside of a package, or the current file wasn't loaded with `\usepackage` or `\documentclass`, then the `top-level` is used as the ⟨*default label*⟩. This may have exceptions—see `\PushDefaultHookLabel`.

This syntax is available in all ⟨*label*⟩ arguments and most ⟨*hook*⟩ arguments, both in the LATEX 2_ε interface, and the LATEX3 interface described in section 2.2.

Important: *The dot-syntax is* ***not*** *available with* `\UseHook` *and some other commands that are typically used within code!*

Note, however, that the replacement of `.` by the ⟨*default label*⟩ takes place when the hook command is executed, so actions that are somehow executed after the package

ends will have the wrong ⟨*default label*⟩ if the dot-syntax is used. For that reason, this syntax is not available in `\UseHook` (and `\hook_use:n`) because the hook is most of the time used outside of the package file in which it was defined. This syntax is also not available in the hook conditionals `\IfHookEmptyTF` (and `\hook_if_empty:nTF`), because these conditionals are used in some performance-critical parts of the hook management code, and because they are usually used to refer to other package's hooks, so the dot-syntax doesn't make much sense.

In some cases, for example in large packages, one may want to separate the code in logical parts, but still use the main package name as the ⟨*label*⟩, then the ⟨*default label*⟩ can be set using `\PushDefaultHookLabel{...}`...`\PopDefaultHookLabel` or `\SetDefaultHookLabel{...}`.

`\PushDefaultHookLabel`
`\PopDefaultHookLabel`

```
\PushDefaultHookLabel {⟨default label⟩}
  ⟨code⟩
\PopDefaultHookLabel
```

`\PushDefaultHookLabel` sets the current ⟨*default label*⟩ to be used in ⟨*label*⟩ arguments, or when replacing a leading "`.`" (see above). `\PopDefaultHookLabel` reverts the ⟨*default label*⟩ to its previous value.

Inside a package or class, the ⟨*default label*⟩ is equal to the package or class name, unless explicitly changed. Everywhere else, the ⟨*default label*⟩ is `top-level` (see section 2.1.6) unless explicitly changed.

The effect of `\PushDefaultHookLabel` holds until the next `\PopDefaultHookLabel`. `\usepackage` (and `\RequirePackage` and `\documentclass`) internally use

```
\PushDefaultHookLabel{⟨package name⟩}
  ⟨package code⟩
\PopDefaultHookLabel
```

to set the ⟨*default label*⟩ for the package or class file. Inside the ⟨*package code*⟩ the ⟨*default label*⟩ can also be changed with `\SetDefaultHookLabel`. `\input` and other file input-related commands from the LaTeX kernel do not use `\PushDefaultHookLabel`, so code within files loaded by these commands does *not* get a dedicated ⟨*label*⟩! (that is, the ⟨*default label*⟩ is the current active one when the file was loaded.)

Packages that provide their own package-like interfaces (Ti*k*Z's `\usetikzlibrary`, for example) can use `\PushDefaultHookLabel` and `\PopDefaultHookLabel` to set dedicated labels and to emulate `\usepackage`-like hook behavior within those contexts.

The `top-level` label is treated differently, and is reserved to the user document, so it is not allowed to change the ⟨*default label*⟩ to `top-level`.

\SetDefaultHookLabel \SetDefaultHookLabel {⟨*default label*⟩}

Similarly to \PushDefaultHookLabel, sets the current ⟨*default label*⟩ to be used in ⟨*label*⟩ arguments, or when replacing a leading ".". The effect holds until the label is changed again or until the next \PopDefaultHookLabel. The difference between \PushDefaultHookLabel and \SetDefaultHookLabel is that the latter does not save the current ⟨*default label*⟩.

This command is useful when a large package is composed of several smaller packages, but all should have the same ⟨*label*⟩, so \SetDefaultHookLabel can be used at the beginning of each package file to set the correct label.

\SetDefaultHookLabel is not allowed in the main document, where the ⟨*default label*⟩ is top-level and there is no \PopDefaultHookLabel to end its effect. It is also not allowed to change the ⟨*default label*⟩ to top-level.

2.1.6 The top-level label

The top-level label, assigned to code added from the main document, is different from other labels. Code added to hooks (usually \AtBeginDocument) in the preamble is almost always to change something defined by a package, so it should go at the very end of the hook.

Therefore, code added in the top-level is always executed at the end of the hook, regardless of where it was declared. If the hook is reversed (see \NewReversedHook), the top-level chunk is executed at the very beginning instead.

Rules regarding top-level have no effect: if a user wants to have a specific set of rules for a code chunk, they should use a different label to said code chunk, and provide a rule for that label instead.

The top-level label is exclusive for the user, so trying to add code with that label from a package results in an error.

2.1.7 Defining relations between hook code

The default assumption is that code added to hooks by different packages are independent and the order in which they are executed is irrelevant. While this is true in many cases it is obviously false in others.

Before the hook management system was introduced packages had to take elaborate precaution to determine of some other package got loaded as well (before or after) and find some ways to alter its behavior accordingly. In addition is was often the user's responsibility to load packages in the right order so that code added to hooks got added in the right order and some cases even altering the loading order wouldn't resolve the conflicts.

With the new hook management system it is now possible to define rules (i.e., relationships) between code chunks added by different packages and explicitly describe in which order they should be processed.

\DeclareHookRule `\DeclareHookRule {⟨hook⟩} {⟨label1⟩} {⟨relation⟩} {⟨label2⟩}`

Defines a relation between ⟨*label1*⟩ and ⟨*label2*⟩ for a given ⟨*hook*⟩. If ⟨*hook*⟩ is `??` this defines a default relation for all hooks that use the two labels, i.e., that have chunks of code labeled with ⟨*label1*⟩ and ⟨*label2*⟩.

Currently, the supported relations are the following:

`before` or `<` Code for ⟨*label1*⟩ comes before code for ⟨*label2*⟩.

`after` or `>` Code for ⟨*label1*⟩ comes after code for ⟨*label2*⟩.

`incompatible-warning` Only code for either ⟨*label1*⟩ or ⟨*label2*⟩ can appear for that hook (a way to say that two packages—or parts of them—are incompatible). A warning is raised if both labels appear in the same hook.

`incompatible-error` Like `incompatible-error` but instead of a warning a LaTeX error is raised, and the code for both labels are dropped from that hook until the conflict is resolved.

`voids` Code for ⟨*label1*⟩ overwrites code for ⟨*label2*⟩. More precisely, code for ⟨*label2*⟩ is dropped for that hook. This can be used, for example if one package is a superset in functionality of another one and therefore wants to undo code in some hook and replace it with its own version.

`unrelated` The order of code for ⟨*label1*⟩ and ⟨*label2*⟩ is irrelevant. This rule is there to undo an incorrect rule specified earlier.

There can only be a single relation between two labels for a given hook, i.e., a later `\DeclareHookRule` overwrites any previous declaration. In all cases rules specific to a given hook take precedence over default rules that use `??` as the ⟨*hook*⟩.

If a default rule is applied, it is done before reversing the label order in a reversed hook, e.g., `before` in a default rule effectively becomes `after` in such a hook. In contrast, a rule for a specific hook is always applied to the state after any reversal (i.e., the state you see when using `\ShowHook` on that hook).

The ⟨*hook*⟩ and ⟨*label*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\ClearHookRule `\ClearHookRule {⟨hook⟩} {⟨label1⟩} {⟨label2⟩}`

Syntactic sugar for saying that ⟨*label1*⟩ and ⟨*label2*⟩ are unrelated for the given ⟨*hook*⟩.

\DeclareDefaultHookRule

\DeclareDefaultHookRule {⟨*label1*⟩} {⟨*relation*⟩} {⟨*label2*⟩}

This sets up a relation between ⟨*label1*⟩ and ⟨*label2*⟩ for all hooks unless overwritten by a specific rule for a hook. Useful for cases where one package has a specific relation to some other package, e.g., is `incompatible` or always needs a special ordering `before` or `after`. (Technically it is just a shorthand for using `\DeclareHookRule` with `??` as the hook name.)

If such a rule is applied to a reversed hook it behaves as if the rule is reversed (e.g., `after` becomes `before`) because those rules are applied first and then the order is reversed. The rationale is that in hook pairs (in which the ordering in one is reversed) default rules have to be reversed too in nearly all scenarios. If this is not the case, a default rule can't be used or has to be overwritten with an explicit `\DeclareHookRule` for that specific hook.

Declaring default rules is only supported in the document preamble.[5]

The ⟨*label*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.8 Querying hooks

Simpler data types, like token lists, have three possible states; they can:

- exist and be empty;
- exist and be non-empty; and
- not exist (in which case emptiness doesn't apply);

Hooks are a bit more complicated: a hook may exist or not, and independently it may or may not be empty. This means that even a hook that doesn't exist may be non-empty and it can also be disabled.

This seemingly strange state may happen when, for example, package A defines hook `A/foo`, and package B adds some code to that hook. However, a document may load package B before package A, or may not load package A at all. In both cases some code is added to hook `A/foo` without that hook being defined yet, thus that hook is said to be non-empty, whereas it doesn't exist. Therefore, querying the existence of a hook doesn't imply its emptiness, neither does the other way around.

Given that code or rules can be added to a hook even if it doesn't physically exist yet, means that a querying its existence has no real use case (in contrast to other variables that can only be update if they have already been declared). For that reason only the test for emptiness has a public interface.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its "next" token list. The hook doesn't need to be declared to have code added to its code pool. A hook is said to exist when it was declared with `\NewHook` or some variant thereof. Generic hooks such as `file` and `env` hooks are automatically declared when code is added to them.

[5] Trying to do so, e.g., via `\DeclareHookRule` with `??` has bad side-effects and is not supported (though not explicitly caught for performance reasons).

\IfHookEmptyTF ⋆
\IfHookEmptyT ⋆
\IfHookEmptyF ⋆

\IfHookEmptyTF {⟨*hook*⟩} {⟨*true code*⟩} {⟨*false code*⟩}

Tests if the ⟨*hook*⟩ is empty (*i.e.*, no code was added to it using either \AddToHook or \AddToHookNext) or such code was removed again (via \RemoveFromHook), and branches to either ⟨*true code*⟩ or ⟨*false code*⟩ depending on the result.

The ⟨*hook*⟩ *cannot* be specified using the dot-syntax. A leading . is treated literally.

2.1.9 Displaying hook code

If one has to adjust the code execution in a hook using a hook rule it is helpful to get some information about the code associated with a hook, its current order and the existing rules.

\ShowHook
\LogHook

\ShowHook {⟨*hook*⟩}
\LogHook {⟨*hook*⟩}

Displays information about the ⟨*hook*⟩ such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

\LogHook prints the information to the .log file, and \ShowHook prints them to the terminal/command window and starts TEX's prompt (only in \errorstopmode) to wait for user action.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

Suppose a hook example-hook whose output of \ShowHook{example-hook} is:

```
1  -> The hook 'example-hook':
2  > Code chunks:
3  >     foo -> [code from package 'foo']
4  >     bar -> [from package 'bar']
5  >     baz -> [package 'baz' is here]
6  > Document-level (top-level) code (executed last):
7  >     -> [code from 'top-level']
8  > Extra code for next invocation:
9  >     -> [one-time code]
10 > Rules:
11 >     foo|baz with relation >
12 >     baz|bar with default relation <
13 > Execution order (after applying rules):
14 >     baz, foo, bar.
```

In the listing above, lines 3 to 5 show the three code chunks added to the hook and their respective labels in the format

⟨*label*⟩ -> ⟨*code*⟩

Line 7 shows the code chunk added by the user in the main document (labeled `top-level`) in the format

```
Document-level (top-level) code (executed ⟨first|last⟩):
  -> ⟨top-level code⟩
```

This code will be either the first or last code executed by the hook (`last` if the hook is normal, `first` if it is reversed). This chunk is not affected by rules and does not take part in sorting.

Line 9 shows the code chunk for the next execution of the hook in the format

```
-> ⟨next-code⟩
```

This code will be used and disappear at the next `\UseHook{example-hook}`, in contrast to the chunks mentioned earlier, which can only be removed from that hook by doing `\RemoveFromHook{⟨label⟩}[example-hook]`.

Lines 11 and 12 show the rules declared that affect this hook in the format

```
⟨label-1⟩|⟨label-2⟩ with ⟨default?⟩ relation ⟨relation⟩
```

which means that the ⟨*relation*⟩ applies to ⟨*label-1*⟩ and ⟨*label-2*⟩, in that order, as detailed in `\DeclareHookRule`. If the relation is `default` it means that this rule applies to ⟨*label-1*⟩ and ⟨*label-2*⟩ in *all* hooks, (unless overridden by a non-default relation).

Finally, line 14 lists the labels in the hook after sorting; that is, in the order they will be executed when the hook is used.

2.1.10 Debugging hook code

`\DebugHooksOn`
`\DebugHooksOff`

`\DebugHooksOn ... \DebugHooksOff`

Turn the debugging of hook code on or off. This displays most changes made to the hook data structures. The output is rather coarse and not really intended for normal use.

2.2 L3 programming layer (`expl3`) interfaces

This is a quick summary of the LaTeX3 programming interfaces for use with packages written in `expl3`. In contrast to the LaTeX 2_ε interfaces they always use mandatory arguments only, e.g., you always have to specify the ⟨*label*⟩ for a code chunk. We therefore suggest to use the declarations discussed in the previous section even in `expl3` packages, but the choice is yours.

`\hook_new:n`
`\hook_new_reversed:n`
`\hook_new_pair:nn`

`\hook_new:n {⟨hook⟩}`
`\hook_new_reversed:n {⟨hook⟩}`
`\hook_new_pair:nn {⟨hook-1⟩} {⟨hook-2⟩}`

Creates a new ⟨*hook*⟩ with normal or reverse ordering of code chunks. `\hook_new_pair:nn` creates a pair of such hooks with `{⟨hook-2⟩}` being a reversed hook. If a hook name is already taken, an error is raised and the hook is not created.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\hook_new_with_args:nn \hook_new_with_args:nn {⟨hook⟩} {⟨number⟩}
\hook_new_reversed_with_args:nn \hook_new_reversed_with_args:nn {⟨hook⟩} {⟨number⟩}
\hook_new_pair_with_args:nnn \hook_new_pair_with_args:nnn {⟨hook-1⟩} {⟨hook-2⟩} {⟨number⟩}

New: 2023-06-01

Creates a new ⟨*hook*⟩ with normal or reverse ordering of code chunks, that takes ⟨*number*⟩ arguments from the input stream when it is used. \hook_new_pair_with_args:nn creates a pair of such hooks with {⟨*hook-2*⟩} being a reversed hook. If a hook name is already taken, an error is raised and the hook is not created.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\hook_disable_generic:n \hook_disable_generic:n {⟨*hook*⟩}

Marks {⟨*hook*⟩} as disabled. Any further attempt to add code to it or declare it, will result in an error and any call to \hook_use:n will simply do nothing.

This declaration is intended for use with generic hooks that are known not to work (see ltcmdhooks-doc) if they receive code.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\hook_activate_generic:n \hook_activate_generic:n {⟨*hook*⟩}

This is like \hook_new:n but it does nothing if the hook was previously declared with \hook_new:n. This declaration should be used only in special situations, e.g., when a command from another package needs to be altered and it is not clear whether a generic cmd hook (for that command) has been previously explicitly declared.

Normally \hook_new:n should be used instead of this.

\hook_use:n \hook_use:n {⟨*hook*⟩}

\hook_use:nnw \hook_use:nnw {⟨*hook*⟩} {⟨*number*⟩} {⟨arg_1⟩} ... {⟨arg_n⟩}

New: 2023-06-01

Executes the {⟨*hook*⟩} code followed (if set up) by the code for next invocation only, then empties that next invocation code. \hook_use:nnw should be used for hooks declared with arguments, and should be followed by as many brace groups as the declared number of arguments. The ⟨*number*⟩ should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove ⟨*number*⟩ items from the input.

The ⟨*hook*⟩ *cannot* be specified using the dot-syntax. A leading . is treated literally.

\hook_use_once:n \hook_use_once:n {⟨*hook*⟩}

\hook_use_once:nnw \hook_use_once:nnw {⟨*hook*⟩} {⟨*number*⟩} {⟨arg_1⟩} ... {⟨arg_n⟩}

New: 2023-06-01

Changes the {⟨*hook*⟩} status so that from now on any addition to the hook code is executed immediately. Then execute any {⟨*hook*⟩} code already set up. \hook_use_once:nnw should be used for hooks declared with arguments, and should be followed by as many brace groups as the declared number of arguments. The ⟨*number*⟩ should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove ⟨*number*⟩ items from the input.

The ⟨*hook*⟩ *cannot* be specified using the dot-syntax. A leading . is treated literally.

\hook_gput_code:nnn `\hook_gput_code:nnn {⟨hook⟩} {⟨label⟩} {⟨code⟩}`

\hook_gput_code_with_args:nnn `\hook_gput_code_with_args:nnn {⟨hook⟩} {⟨label⟩} {⟨code⟩}`

New: 2023-06-01

Adds a chunk of ⟨*code*⟩ to the ⟨*hook*⟩ labeled ⟨*label*⟩. If the label already exists the ⟨*code*⟩ is appended to the already existing code.

If `\hook_gput_code_with_args:nnn` is used, the ⟨*code*⟩ can access the arguments passed to `\hook_use:nnw` (or `\hook_use_once:nnw`) with `#1`, `#2`, ..., `#n` (up to the number of arguments declared for the hook). In that case, if an actual parameter token should be added to the code, it should be doubled.

If code is added to an external ⟨*hook*⟩ (of the kernel or another package) then the convention is to use the package name as the ⟨*label*⟩ not some internal module name or some other arbitrary string.

The ⟨*hook*⟩ and ⟨*label*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\hook_gput_next_code:nn `\hook_gput_next_code:nn {⟨hook⟩} {⟨code⟩}`

\hook_gput_next_code_with_args:nn `\hook_gput_next_code_with_args:nn {⟨hook⟩} {⟨code⟩}`

New: 2023-06-01

Adds a chunk of ⟨*code*⟩ for use only in the next invocation of the ⟨*hook*⟩. Once used it is gone.

If `\hook_gput_next_code_with_args:nn` is used, the ⟨*code*⟩ can access the arguments passed to `\hook_use:nnw` (or `\hook_use_once:nnw`) with `#1`, `#2`, ..., `#n` (up to the number of arguments declared for the hook). In that case, if an actual parameter token should be added to the code, it should be doubled.

This is simpler than `\hook_gput_code:nnn`, the code is simply appended to the hook in the order of declaration at the very end, i.e., after all standard code for the hook got executed. Thus if one needs to undo what the standard does one has to do that as part of ⟨*code*⟩.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\hook_gclear_next_code:n `\hook_gclear_next_code:n {⟨hook⟩}`

Undo any earlier `\hook_gput_next_code:nn`.

\hook_gremove_code:nn `\hook_gremove_code:nn {⟨hook⟩} {⟨label⟩}`

Removes any code for ⟨*hook*⟩ labeled ⟨*label*⟩.

If there is no code under the ⟨*label*⟩ in the ⟨*hook*⟩, or if the ⟨*hook*⟩ does not exist, a warning is issued when you attempt to use `\hook_gremove_code:nn`, and the command is ignored.

If the second argument is `*`, then all code chunks are removed. This is rather dangerous as it drops code from other packages one may not know about, so think twice before using that!

The ⟨*hook*⟩ and ⟨*label*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_gset_rule:nnnn` `\hook_gset_rule:nnnn {⟨hook⟩} {⟨label1⟩} {⟨relation⟩} {⟨label2⟩}`

Relate ⟨*label1*⟩ with ⟨*label2*⟩ when used in ⟨*hook*⟩. See `\DeclareHookRule` for the allowed ⟨*relation*⟩s. If ⟨*hook*⟩ is `??` a default rule is specified.

The ⟨*hook*⟩ and ⟨*label*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5. The dot-syntax is parsed in both ⟨*label*⟩ arguments, but it usually makes sense to be used in only one of them.

`\hook_if_empty_p:n` ⋆
`\hook_if_empty:nTF` ⋆

`\hook_if_empty:nTF {⟨hook⟩} {⟨true code⟩} {⟨false code⟩}`

Tests if the ⟨*hook*⟩ is empty (*i.e.*, no code was added to it using either `\AddToHook` or `\AddToHookNext`), and branches to either ⟨*true code*⟩ or ⟨*false code*⟩ depending on the result.

The ⟨*hook*⟩ *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

`\hook_show:n` `\hook_show:n {⟨hook⟩}`
`\hook_log:n` `\hook_log:n {⟨hook⟩}`

Displays information about the ⟨*hook*⟩ such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

`\hook_log:n` prints the information to the `.log` file, and `\hook_show:n` prints them to the terminal/command window and starts TEX's prompt (only if `\errorstopmode`) to wait for user action.

The ⟨*hook*⟩ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_debug_on:`
`\hook_debug_off:`

`\hook_debug_on:`

Turns the debugging of hook code on or off. This displays changes to the hook data.

2.3 On the order of hook code execution

Chunks of code for a ⟨*hook*⟩ under different labels are supposed to be independent if there are no special rules set up that define a relation between the chunks. This means that you can't make assumptions about the order of execution!

Suppose you have the following declarations:

```
\NewHook{myhook}
\AddToHook{myhook}[packageA]{\typeout{A}}
\AddToHook{myhook}[packageB]{\typeout{B}}
\AddToHook{myhook}[packageC]{\typeout{C}}
```

then executing the hook with `\UseHook` will produce the typeout `A B C` in that order. In other words, the execution order is computed to be `packageA`, `packageB`, `packageC` which you can verify with `\ShowHook{myhook}`:

```
-> The hook 'myhook':
> Code chunks:
>     packageA -> \typeout {A}
>     packageB -> \typeout {B}
>     packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>     ---
> Extra code for next invocation:
>     ---
> Rules:
>     ---
> Execution order:
>     packageA, packageB, packageC.
```

The reason is that the code chunks are internally saved in a property list and the initial order of such a property list is the order in which key-value pairs got added. However, that is only true if nothing other than adding happens!

Suppose, for example, you want to replace the code chunk for `packageA`, e.g.,

```
\RemoveFromHook{myhook}[packageA]
\AddToHook{myhook}[packageA]{\typeout{A alt}}
```

then your order becomes `packageB`, `packageC`, `packageA` because the label got removed from the property list and then re-added (at its end).

While that may not be too surprising, the execution order is also sometimes altered if you add a redundant rule, e.g. if you specify

```
\DeclareHookRule{myhook}{packageA}{before}{packageB}
```

instead of the previous lines we get

```
-> The hook 'myhook':
> Code chunks:
>     packageA -> \typeout {A}
>     packageB -> \typeout {B}
>     packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>     ---
> Extra code for next invocation:
>     ---
> Rules:
>     packageB|packageA with relation >
> Execution order (after applying rules):
>     packageA, packageC, packageB.
```

As you can see the code chunks are still in the same order, but in the execution order for the labels `packageB` and `packageC` have swapped places. The reason is that, with the rule there are two orders that satisfy it, and the algorithm for sorting happened to pick a different one compared to the case without rules (where it doesn't run at all as there is nothing to resolve). Incidentally, if we had instead specified the redundant rule

```
\DeclareHookRule{myhook}{packageB}{before}{packageC}
```

the execution order would not have changed.

In summary: it is not possible to rely on the order of execution unless there are rules that partially or fully define the order (in which you can rely on them being fulfilled).

2.4 The use of "reversed" hooks

You may have wondered why you can declare a "reversed" hook with `\NewReversedHook` and what that does exactly.

In short: the execution order of a reversed hook (without any rules!) is exactly reversed to the order you would have gotten for a hook declared with `\NewHook`.

This is helpful if you have a pair of hooks where you expect to see code added that involves grouping, e.g., starting an environment in the first and closing that environment in the second hook. To give a somewhat contrived example[6], suppose there is a package adding the following:

```
\AddToHook{env/quote/before}[package-1]{\begin{itshape}}
\AddToHook{env/quote/after} [package-1]{\end{itshape}}
```

As a result, all quotes will be in italics. Now suppose further that another `package-too` makes the quotes also in blue and therefore adds:

```
\usepackage{color}
\AddToHook{env/quote/before}[package-too]{\begin{color}{blue}}
\AddToHook{env/quote/after} [package-too]{\end{color}}
```

Now if the `env/quote/after` hook would be a normal hook we would get the same execution order in both hooks, namely:

```
package-1, package-too
```

(or vice versa) and as a result, would get:

```
\begin{itshape}\begin{color}{blue} ...
\end{itshape}\end{color}
```

and an error message saying that `\begin{color}` was ended by `\end{itshape}`. With `env/quote/after` declared as a reversed hook the execution order is reversed and so all environments are closed in the correct sequence and `\ShowHook` would give us the following output:

```
-> The hook 'env/quote/after':
> Code chunks:
>     package-1 -> \end {itshape}
>     package-too -> \end {color}
> Document-level (top-level) code (executed first):
>     ---
> Extra code for next invocation:
>     ---
> Rules:
>     ---
> Execution order (after reversal):
>     package-too, package-1.
```

If there is a matching default rule (done with `\DeclareDefaultHookRule` or with `??` for the hook name) then this default rule is applied before the reversal so that the order in the reversed hook mirrors the one in the normal hook. However, all rules specific to a hook happen always after the reversal of the execution order, so if you alter the order you will probably have to alter it in both hooks, not just in one, but that depends on the use case.

[6] There are simpler ways to achieve the same effect.

2.5 Difference between "normal" and "one-time" hooks

When executing a hook a developer has the choice of using either `\UseHook` or `\UseOneTimeHook` (or their expl3 equivalents `\hook_use:n` and `\hook_use_once:n`). This choice affects how `\AddToHook` is handled after the hook has been executed for the first time.

With normal hooks adding code via `\AddToHook` means that the code chunk is added to the hook data structure and then used each time `\UseHook` is called.

With one-time hooks it this is handled slightly differently: After `\UseOneTimeHook` has been called, any further attempts to add code to the hook via `\AddToHook` will simply execute the ⟨*code*⟩ immediately.

This has some consequences one needs to be aware of:

- If ⟨*code*⟩ is added to a normal hook after the hook was executed and it is never executed again for one or the other reason, then this new ⟨*code*⟩ will never be executed.
- In contrast if that happens with a one-time hook the ⟨*code*⟩ is executed immediately.

In particular this means that construct such as

```
\AddToHook{myhook}
          { ⟨code-1⟩ \AddToHook{myhook}{⟨code-2⟩} ⟨code-3⟩ }
```

works for one-time hooks[7] (all three code chunks are executed one after another), but it makes little sense with a normal hook, because with a normal hook the first time `\UseHook{myhook}` is executed it would

- execute ⟨*code-1*⟩,
- then execute `\AddToHook{myhook}{code-2}` which adds the code chunk ⟨*code-2*⟩ to the hook for use on the next invocation,
- and finally execute ⟨*code-3*⟩.

The second time `\UseHook` is called it would execute the above and in addition ⟨*code-2*⟩ as that was added as a code chunk to the hook in the meantime. So each time the hook is used another copy of ⟨*code-2*⟩ is added and so that code chunk is executed ⟨*# of invocations*⟩ − 1 times.

2.6 Generic hooks provided by packages

The hook management system also implements a category of hooks that are called "Generic Hooks". Normally a hook has to be explicitly declared before it can be used in code. This ensures that different packages are not using the same hook name for unrelated purposes—something that would result in absolute chaos. However, there are a number of "standard" hooks where it is unreasonable to declare them beforehand, e.g, each and every command has (in theory) an associated `before` and `after` hook. In such cases, i.e., for command, environment or file hooks, they can be used simply by adding code to them with `\AddToHook`. For more specialized generic hooks, e.g., those provided

[7] This is sometimes used with `\AtBeginDocument` which is why it is supported.

by babel, you have to additionally enable them with \ActivateGenericHook as explained below.

The generic hooks provided by LaTeX are those for cmd, env, file, include, package, and class, and all these are available out of the box: you only have to use \AddToHook to add code to them, but you don't have to add \UseHook or \UseOneTimeHook to your code, because this is already done for you (or, in the case of cmd hooks, the command's code is patched at \begin{document}, if necessary).

However, if you want to provide further generic hooks in your own code, the situation is slightly different. To do this you should use \UseHook or \UseOneTimeHook, but *without declaring the hook* with \NewHook. As mentioned earlier, a call to \UseHook with an undeclared hook name does nothing. So as an additional setup step, you need to explicitly activate your generic hook. Note that a generic hook produced in this way is always a normal hook.

For a truly generic hook, with a variable part in the hook name, such upfront activation would be difficult or impossible, because you typically do not know what kind of variable parts may come up in real documents.

For example, babel provides hooks such as babel/⟨*language*⟩/afterextras. However, language support in babel is often done through external language packages. Thus doing the activation for all languages inside the core babel code is not a viable approach. Instead it needs to be done by each language package (or by the user who wants to use a particular hook).

Because the hooks are not declared with \NewHook their names should be carefully chosen to ensure that they are (likely to be) unique. Best practice is to include the package or command name, as was done in the babel example above.

Generic hooks defined in this way are always normal hooks (i.e., you can't implement reversed hooks this way). This is a deliberate limitation, because it speeds up the processing considerably.

2.7 Hooks with arguments

Sometimes it is necessary to pass contextual information to a hook, and, for one reason or another, it is not feasible to store such information in macros. To serve this purpose, hooks can be declared with arguments, so that the programmer can pass along the data necessary for the code in the hook to function properly.

A hook with arguments works mostly like a regular hook, and most commands that work for regular hooks, also work for hooks that take arguments. The differences are when the hook is declared (\NewHookWithArguments is used instead of \NewHook), then code can be added with both \AddToHook and \AddToHookWithArguments, and when the hook is used (\UseHookWithArguments instead of \UseHook).

A hook with arguments must be declared as such (before it is first used, as all regular hooks) using \NewHookWithArguments{⟨*hook*⟩}{⟨*number*⟩}. All code added to that hook can then use #1 to access the first argument, #2 to access the second, and so forth up to the number of arguments declared. However, it is still possible to add code with references to the arguments of a hook that was not yet declared (we will discuss that later). At their core, hooks are macros, so TeX's limit of 9 arguments applies, and a low-level TeX error is raised if you try to reference an argument number that doesn't exist.

To use a hook with arguments, just write `\UseHookWithArguments{⟨hook⟩}{⟨number⟩}` followed by a braced list of the arguments. For example, if the hook `test` takes three arguments, write:

```
\UseHookWithArguments{test}{3}{arg-1}{arg-2}{arg-3}
```

then, in the ⟨*code*⟩ of the hook, all instances of `#1` will be replaced by `arg-1`, `#2` by `arg-2` and so on. If, at the point of usage, the programmer provides more arguments than the hook is declared to take, the excess arguments are simply ignored by the hook. Behaviour is unpredictable[8] if too few arguments are provided. If the hook isn't declared, ⟨*number*⟩ arguments are removed from the input stream.

Adding code to a hook with arguments can be done with `\AddToHookWithArguments` as well as with the regular `\AddToHook`, to achieve different outcomes. The main difference when it comes to adding code to a hook, in this case, is firstly the possibility of accessing a hook's arguments, of course, and second, how parameter tokens ($\#_6$) are treated.

Using `\AddToHook` in a hook that takes arguments will work as it does for all other hooks. This allows a package developer to add arguments to a hook that otherwise had none without having to worry about compatibility. This means that, for example:

```
\AddToHook{test}{\def\foo#1{Hello, #1!}}
```

will define the same macro `\foo` regardless if the hook `test` takes arguments or not.

Using `\AddToHookWithArguments` allows the ⟨*code*⟩ added to access the arguments of the hook with `#1`, `#2`, and so forth, up to the number of the arguments declared in the hook. This means that if one wants to add a $\#_6$ to the ⟨*code*⟩ that token must be doubled in the input. The same definition from above, using `\AddToHookWithArguments`, needs to be rewritten:

```
\AddToHookWithArguments{test}{\def\foo##1{Hello, ##1!}}
```

Extending the above example to use the hook arguments, we could rewrite something like (now from declaration to usage, to get the whole picture):

```
\NewHookWithArguments{test}{1}
\AddToHookWithArguments{test}{%
  \typeout{Defining foo with "#1"}
  \def\foo##1{Hello, ##1! Some text after: #1}%
}
\UseHook{test}{Howdy!}
\ShowCommand\foo
```

Running the code above prints in the terminal:

```
Defining foo with "Howdy!"
> \foo=macro:
#1->Hello, #1! Some text after: Howdy!.
```

[8] The hook *will* take the declared number of arguments, and what will happen depends on what was grabbed, and what the hook code does with its arguments.

Note how `##1` in the call to `\AddToHookWithArguments` became `#1`, and the `#1` was replaced by the argument passed to the hook. Should the hook be used again, with a different argument, the definition would naturally change.

It is possible to add code referencing a hook's arguments before such hook is declared and the number of hooks is fixed. However, if some code is added to the hook, that references more arguments than will be declared for the hook, there will be a low-level TEX error about an "Illegal parameter number" at the time the hook is declared, which will be hard to track down because at that point TEX can't know whence the offending code came from. Thus it is important that package writers explicitly document how many arguments (if any) each hook can take, so users of those packages know how many arguments can be referenced, and equally important, what each argument means.

2.8 Private LATEX kernel hooks

There are a few places where it is absolutely essential for LATEX to function correctly that code is executed in a precisely defined order. Even that could have been implemented with the hook management (by adding various rules to ensure the appropriate ordering with respect to other code added by packages). However, this makes every document unnecessary slow, because there has to be sorting even though the result is predetermined. Furthermore it forces package writers to unnecessarily add such rules if they add further code to the hook (or break LATEX).

For that reason such code is not using the hook management, but instead private kernel commands directly before or after a public hook with the following naming convention: `\@kernel@before@`⟨*hook*⟩ or `\@kernel@after@`⟨*hook*⟩. For example, in `\enddocument` you find

```
\UseHook{enddocument}%
\@kernel@after@enddocument
```

which means first the user/package-accessible `enddocument` hook is executed and then the internal kernel hook. As their name indicates these kernel commands should not be altered by third-party packages, so please refrain from that in the interest of stability and instead use the public hook next to it.[9]

2.9 Legacy LATEX 2_ε interfaces

LATEX 2_ε offered a small number of hooks together with commands to add to them. They are listed here and are retained for backwards compatibility.

With the new hook management, several additional hooks have been added to LATEX and more will follow. See the next section for what is already available.

[9] As with everything in TEX there is not enforcement of this rule, and by looking at the code it is easy to find out how the kernel adds to them. The main reason of this section is therefore to say "please don't do that, this is unconfigurable code!"

\AtBeginDocument `\AtBeginDocument [⟨label⟩] {⟨code⟩}`

If used without the optional argument ⟨*label*⟩, it works essentially like before, i.e., it is adding ⟨*code*⟩ to the hook `begindocument` (which is executed inside `\begin{document}`). However, all code added this way is labeled with the label `top-level` (see section 2.1.6) if done outside of a package or class or with the package/class name if called inside such a file (see section 2.1.5).

This way one can add code to the hook using `\AddToHook` or `\AtBeginDocument` using a different label and explicitly order the code chunks as necessary, e.g., run some code before or after another package's code. When using the optional argument the call is equivalent to running `\AddToHook {begindocument} [⟨label⟩] {⟨code⟩}`.

`\AtBeginDocument` is a wrapper around the `begindocument` hook (see section 3.2), which is a one-time hook. As such, after the `begindocument` hook is executed at `\begin{document}` any attempt to add ⟨*code*⟩ to this hook with `\AtBeginDocument` or with `\AddToHook` will cause that ⟨*code*⟩ to execute immediately instead. See section 2.5 for more on one-time hooks.

For important packages with known order requirement we may over time add rules to the kernel (or to those packages) so that they work regardless of the loading-order in the document.

\AtEndDocument `\AtEndDocument [⟨label⟩] {⟨code⟩}`

Like `\AtBeginDocument` but for the `enddocument` hook.

The few hooks that existed previously in LaTeX 2ε used internally commands such as `\@begindocumenthook` and packages sometimes augmented them directly rather than working through `\AtBeginDocument`. For that reason there is currently support for this, that is, if the system detects that such an internal legacy hook command contains code it adds it to the new hook system under the label `legacy` so that it doesn't get lost.

However, over time the remaining cases of direct usage need updating because in one of the future release of LaTeX we will turn this legacy support off, as it does unnecessary slow down the processing.

3 LaTeX 2ε commands and environments augmented by hooks

In this section we describe the standard hooks that are now offered by LaTeX, or give pointers to other documents in which they are described. This section will grow over time (and perhaps eventually move to usrguide3).

3.1 Generic hooks

As stated earlier, with the exception of generic hooks, all hooks must be declared with `\NewHook` before they can be used. All generic hooks have names of the form "⟨*type*⟩/⟨*name*⟩/⟨*position*⟩", where ⟨*type*⟩ is from the predefined list shown below, and ⟨*name*⟩ is the variable part whose meaning will depend on the ⟨*type*⟩. The last component, ⟨*position*⟩, has more complex possibilities: it can always be `before` or `after`; for `env` hooks, it can also be `begin` or `end`; and for `include` hooks it can also be `end`. Each specific hook is documented below, or in `ltcmdhooks-doc.pdf` or `ltfilehook-doc.pdf`.

The generic hooks provided by LaTeX belong to one of the six types:

env Hooks executed before and after environments – ⟨*name*⟩ is the name of the environment, and available values for ⟨*position*⟩ are `before`, `begin`, `end`, and `after`;

cmd Hooks added to and executed before and after commands – ⟨*name*⟩ is the name of the command, and available values for ⟨*position*⟩ are `before` and `after`;

file Hooks executed before and after reading a file – ⟨*name*⟩ is the name of the file (with extension), and available values for ⟨*position*⟩ are `before` and `after`;

package Hooks executed before and after loading packages – ⟨*name*⟩ is the name of the package, and available values for ⟨*position*⟩ are `before` and `after`;

class Hooks executed before and after loading classes – ⟨*name*⟩ is the name of the class, and available values for ⟨*position*⟩ are `before` and `after`;

include Hooks executed before and after `\included` files – ⟨*name*⟩ is the name of the included file (without the `.tex` extension), and available values for ⟨*position*⟩ are `before`, `end`, and `after`.

Each of the hooks above are detailed in the following sections and in linked documentation.

3.1.1 Generic hooks for all environments

Every environment ⟨*env*⟩ has now four associated hooks coming with it:

`env/`⟨*env*⟩`/before` This hook is executed as part of `\begin` as the very first action, in particular prior to starting the environment group. Its scope is therefore not restricted by the environment.

`env/`⟨*env*⟩`/begin` This hook is executed as part of `\begin` directly in front of the code specific to the environment start (e.g., the third argument of `\NewDocumentEnvironment` and the second argument of `\newenvironment`). Its scope is the environment body.

`env/`⟨*env*⟩`/end` This hook is executed as part of `\end` directly in front of the code specific to the end of the environment (e.g., the forth argument of `\NewDocumentEnvironment` and the third argument of `\newenvironment`).

`env/`⟨*env*⟩`/after` This hook is executed as part of `\end` after the code specific to the environment end and after the environment group has ended. Its scope is therefore not restricted by the environment.

The hook is implemented as a reversed hook so if two packages add code to `env/`⟨*env*⟩`/before` and to `env/`⟨*env*⟩`/after` they can add surrounding environments and the order of closing them happens in the right sequence.

Given that these generic hook names involve `/` as part of their name they would not work if one tries to define an environment using a name that involves a `/`.[10]

Generic environment hooks are never one-time hooks even with environments that are supposed to appear only once in a document.[11] In contrast to other hooks there is also no need to declare them using `\NewHook`.

[10] Officially, LaTeX names for environments should only consist of a sequence of letters, numbers, and the character `*`, i.e., this is not a new restriction.

[11] Thus if one adds code to such hooks after the environment has been processed, it will only be executed if the environment appears again and if that doesn't happen the code will never get executed.

The hooks are only executed if `\begin{⟨env⟩}` and `\end{⟨env⟩}` is used. If the environment code is executed via low-level calls to `\⟨env⟩` and `\end⟨env⟩` (e.g., to avoid the environment grouping) they are not available. If you want them available in code using this method, you would need to add them yourself, i.e., write something like

```
\UseHook{env/quote/before}\quote
    ...
\endquote\UseHook{env/quote/after}
```

to add the outer hooks, etc.

Largely for compatibility with existing packages, the following four commands are also available to set the environment hooks; but for new packages we recommend directly using the hook names and `\AddToHook`.

`\BeforeBeginEnvironment`

`\BeforeBeginEnvironment [⟨label⟩] {⟨env⟩} {⟨code⟩}`

This declaration adds to the `env/⟨env⟩/before` hook using the ⟨*label*⟩. If ⟨*label*⟩ is not given, the ⟨*default label*⟩ is used (see section 2.1.5).

`\AtBeginEnvironment`

`\AtBeginEnvironment [⟨label⟩] {⟨env⟩} {⟨code⟩}`

This is like `\BeforeBeginEnvironment` but it adds to the `env/⟨env⟩/begin` hook.

`\AtEndEnvironment`

`\AtEndEnvironment [⟨label⟩] {⟨env⟩} {⟨code⟩}`

This is like `\BeforeBeginEnvironment` but it adds to the `env/⟨env⟩/end` hook.

`\AfterEndEnvironment`

`\AfterEndEnvironment [⟨label⟩] {⟨env⟩} {⟨code⟩}`

This is like `\BeforeBeginEnvironment` but it adds to the `env/⟨env⟩/after` hook.

3.1.2 Generic hooks for commands

Similar to environments there are now (at least in theory) two generic hooks available for any LaTeX command. These are

`cmd/⟨name⟩/before` This hook is executed at the very start of the command execution.

`cmd/⟨name⟩/after` This hook is executed at the very end of the command body. It is implemented as a reversed hook.

In practice there are restrictions and especially the `after` hook works only with a subset of commands. Details about these restrictions are documented in `ltcmdhooks-doc.pdf` or with code in `ltcmdhooks-code.pdf`.

3.1.3 Generic hooks provided by file loading operations

There are several hooks added to LaTeX's process of loading file via its high-level interfaces such as `\input`, `\include`, `\usepackage`, `\RequirePackage`, etc. These are documented in `ltfilehook-doc.pdf` or with code in `ltfilehook-code.pdf`.

3.2 Hooks provided by \begin{document}

Until 2020 \begin{document} offered exactly one hook that one could add to using \AtBeginDocument. Experiences over the years have shown that this single hook in one place was not enough and as part of adding the general hook management system a number of additional hooks have been added at this point. The places for these hooks have been chosen to provide the same support as offered by external packages, such as etoolbox and others that augmented \document to gain better control.

Supported are now the following hooks (all of them one-time hooks):

begindocument/before This hook is executed at the very start of \document, one can think of it as a hook for code at the end of the preamble section and this is how it is used by etoolbox's \AtEndPreamble.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

begindocument This hook is added to by using \AddToHook{begindocument} or by using \AtBeginDocument and it is executed after the .aux file has been read and most initialization are done, so they can be altered and inspected by the hook code. It is followed by a small number of further initializations that shouldn't be altered and are therefore coming later.

The hook should not be used to add material for typesetting as we are still in LaTeX's initialization phase and not in the document body. If such material needs to be added to the document body use the next hook instead.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

begindocument/end This hook is executed at the end of the \document code in other words at the beginning of the document body. The only command that follows it is \ignorespaces.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

The generic hooks executed by \begin also exist, i.e., env/document/before and env/document/begin, but with this special environment it is better use the dedicated one-time hooks above.

3.3 Hooks provided by \end{document}

LaTeX 2$_\varepsilon$ has always provided \AtEndDocument to add code to the \end{document}, just in front of the code that is normally executed there. While this was a big improvement over the situation in LaTeX 2.09, it was not flexible enough for a number of use cases and so packages, such as etoolbox, atveryend and others patched \enddocument to add additional points where code could be hooked into.

Patching using packages is always problematical as leads to conflicts (code availability, ordering of patches, incompatible patches, etc.). For this reason a number of additional hooks have been added to the \enddocument code to allow packages to add code in various places in a controlled way without the need for overwriting or patching the core code.

Supported are now the following hooks (all of them one-time hooks):

enddocument The hook associated with `\AtEndDocument`. It is immediately called at the beginning of `\enddocument`.

When this hook is executed there may be still unprocessed material (e.g., floats on the deferlist) and the hook may add further material to be typeset. After it, `\clearpage` is called to ensure that all such material gets typeset. If there is nothing waiting the `\clearpage` has no effect.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/afterlastpage As the name indicates this hook should not receive code that generates material for further pages. It is the right place to do some final housekeeping and possibly write out some information to the `.aux` file (which is still open at this point to receive data, but since there will be no more pages you need to write to it using `\immediate\write`). It is also the correct place to set up any testing code to be run when the `.aux` file is re-read in the next step.

After this hook has been executed the `.aux` file is closed for writing and then read back in to do some tests (e.g., looking for missing references or duplicated labels, etc.).

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/afteraux At this point, the `.aux` file has been reprocessed and so this is a possible place for final checks and display of information to the user. However, for the latter you might prefer the next hook, so that your information is displayed after the (possibly longish) list of files if that got requested via `\listfiles`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/info This hook is meant to receive code that write final information messages to the terminal. It follows immediately after the previous hook (so both could have been combined, but then packages adding further code would always need to also supply an explicit rule to specify where it should go.

This hook already contains some code added by the kernel (under the labels `kernel/filelist` and `kernel/warnings`), namely the list of files when `\listfiles` has been used and the warnings for duplicate labels, missing references, font substitutions etc.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/end Finally, this hook is executed just in front of the final call to `\@@end`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).is it even possible to add code after this one?

There is also the hook `shipout/lastpage`. This hook is executed as part of the last `\shipout` in the document to allow package to add final `\special`'s to that page. Where this hook is executed in relation to those from the above list can vary from document to document. Furthermore to determine correctly which of the `\shipout`s is the last one,

LaTeX needs to be run several times, so initially it might get executed on the wrong page. See section 3.4 for where to find the details.

It is in also possible to use the generic `env/document/end` hook which is executed by `\end`, i.e., just in front of the first hook above. Note however that the other generic `\end` environment hook, i.e., `env/document/after` will never get executed, because by that time LaTeX has finished the document processing.

3.4 Hooks provided by `\shipout` operations

There are several hooks and mechanisms added to LaTeX's process of generating pages. These are documented in `ltshipout-doc.pdf` or with code in `ltshipout-code.pdf`.

3.5 Hooks provided for paragraphs

The paragraph processing has been augmented to include a number of internal and public hooks. These are documented in `ltpara-doc.pdf` or with code in `ltpara-code.pdf`.

3.6 Hooks provided in NFSS commands

In languages that need to support for more than one script in parallel (and thus several sets of fonts, e.g., supporting both Latin and Japanese fonts), NFSS font commands such as `\sffamily` need to switch both the Latin family to "Sans Serif" and in addition alter a second set of fonts.

To support this, several NFSS commands have hooks to which such support can be added.

`rmfamily` After `\rmfamily` has done its initial checks and prepared a font series update, this hook is executed before `\selectfont`.

`sffamily` This is like the `rmfamily` hook, but for the `\sffamily` command.

`ttfamily` This is like the `rmfamily` hook, but for the `\ttfamily` command.

`normalfont` The `\normalfont` command resets the font encoding, family, series and shape to their document defaults. It then executes this hook and finally calls `\selectfont`.

`expand@font@defaults` The internal `\expand@font@defaults` command expands and saves the current defaults for the meta families (rm/sf/tt) and the meta series (bf/md). If the NFSS machinery has been augmented, e.g., for Chinese or Japanese fonts, then further defaults may need to be set at this point. This can be done in this hook which is executed at the end of this macro.

`bfseries/defaults, bfseries` If the `\bfdefault` was explicitly changed by the user, its new value is used to set the bf series defaults for the meta families (rm/sf/tt) when `\bfseries` is called. The `bfseries/defaults` hook allows further adjustments to be made in this case. This hook is only executed if such a change is detected. In contrast, the `bfseries` hook is always executed just before `\selectfont` is called to change to the new series.

`mdseries/defaults, mdseries` These two hooks are like the previous ones but they are in the `\mdseries` command.

selectfont This hook is executed inside `\selectfont`, after the current values for *encoding*, *family*, *series*, *shape*, and *size* are evaluated and the new font is selected (and if necessary loaded). After the hook has executed, NFSS will still do any updates necessary for a new *size* (such as changing the size of `\strut`) and any updates necessary to a change in *encoding*.

This hook is intended for use cases where, in parallel to a change in the main font, some other fonts need to be altered (e.g., in CJK processing where you may need to deal with several different alphabets).

3.7 Hook provided by the mark mechanism

See `ltmarks-doc.pdf` for details.

insertmark This hook allows for a special setup while `\InsertMark` inserts a mark. It is executed in group so local changes only apply to the mark being inserted.

4 The Implementation

```
⟨@@=hook⟩
⟨*2ekernel | latexrelease⟩
\ExplSyntaxOn
⟨latexrelease⟩\NewModuleRelease{2020/10/01}{lthooks}
⟨latexrelease⟩                    {The~hook~management~system}
```

4.1 Debugging

`\g__hook_debug_bool` Holds the current debugging state.

```
\bool_new:N \g__hook_debug_bool
```

(*End of definition for* `\g__hook_debug_bool`.)

`\hook_debug_on:` `\hook_debug_off:` `\__hook_debug:n` `\__hook_debug_gset:` Turns debugging on and off by redefining `\__hook_debug:n`.

```
\cs_new_eq:NN \__hook_debug:n \use_none:n
\cs_new_protected:Npn \hook_debug_on:
  {
    \bool_gset_true:N \g__hook_debug_bool
    \__hook_debug_gset:
  }
\cs_new_protected:Npn \hook_debug_off:
  {
    \bool_gset_false:N \g__hook_debug_bool
    \__hook_debug_gset:
  }
\cs_new_protected:Npn \__hook_debug_gset:
  {
    \cs_gset_protected:Npx \__hook_debug:n ##1
      { \bool_if:NT \g__hook_debug_bool {##1} }
  }
```

(*End of definition for* `\hook_debug_on:` *and others. These functions are documented on page* 209.)

4.2 Borrowing from internals of other kernel modules

__hook_str_compare:nn Private copy of __str_if_eq:nn

```
\cs_new_eq:NN \__hook_str_compare:nn \__str_if_eq:nn
```

(*End of definition for* __hook_str_compare:nn.)

4.3 Declarations

\l__hook_tmpa_bool Scratch boolean used throughout the package.

```
\bool_new:N \l__hook_tmpa_bool
```

(*End of definition for* \l__hook_tmpa_bool.)

\l__hook_return_tl
\l__hook_tmpa_tl
\l__hook_tmpb_tl Scratch variables used throughout the package.

```
\tl_new:N \l__hook_return_tl
\tl_new:N \l__hook_tmpa_tl
\tl_new:N \l__hook_tmpb_tl
```

(*End of definition for* \l__hook_return_tl, \l__hook_tmpa_tl, *and* \l__hook_tmpb_tl.)

\g__hook_all_seq In a few places we need a list of all hook names ever defined so we keep track if them in this sequence.

```
\seq_new:N \g__hook_all_seq
```

(*End of definition for* \g__hook_all_seq.)

\l__hook_cur_hook_tl Stores the name of the hook currently being sorted.

```
\tl_new:N \l__hook_cur_hook_tl
```

(*End of definition for* \l__hook_cur_hook_tl.)

\l__hook_work_prop A property list holding a copy of the \g__hook_⟨*hook*⟩_code_prop of the hook being sorted to work on, so that changes don't act destructively on the hook data structure.

```
\prop_new:N \l__hook_work_prop
```

(*End of definition for* \l__hook_work_prop.)

\g__hook_used_prop All hooks that receive code (for use in debugging display).

```
\prop_new:N \g__hook_used_prop
```

(*End of definition for* \g__hook_used_prop.)

\g__hook_hook_curr_name_tl
\g__hook_name_stack_seq Default label used for hook commands, and a stack to keep track of packages within packages.

```
\tl_new:N \g__hook_hook_curr_name_tl
\seq_new:N \g__hook_name_stack_seq
```

(*End of definition for* \g__hook_hook_curr_name_tl *and* \g__hook_name_stack_seq.)

__hook_tmp:w Temporary macro for generic usage.

```
\cs_new_eq:NN \__hook_tmp:w ?
```

(*End of definition for* __hook_tmp:w.)

\c__hook_empty_tl
\c__hook_nine_parameters_tl

An empty token list, and one containing nine parameters.

```
\tl_const:Nn \c__hook_empty_tl { }
\tl_const:Nn \c__hook_nine_parameters_tl { #1#2#3#4#5#6#7#8#9 }
```

(*End of definition for* \c__hook_empty_tl *and* \c__hook_nine_parameters_tl.)

\tl_gremove_once:Nx
\tl_show:x
\tl_log:x
\tl_set:Ne
\cs_replacement_spec:c
\prop_put:Nne
\str_count:e

Some variants of expl3 functions.

FMi: should probably be moved to expl3

```
\cs_generate_variant:Nn \tl_gremove_once:Nn { Nx }
\cs_generate_variant:Nn \tl_show:n { x }
\cs_generate_variant:Nn \tl_log:n { x }
\cs_generate_variant:Nn \tl_set:Nn { Ne }
\cs_generate_variant:Nn \cs_replacement_spec:N { c }
\cs_generate_variant:Nn \prop_put:Nnn { Nne }
\cs_generate_variant:Nn \str_count:n { e }
```

(*End of definition for* \tl_gremove_once:Nx *and others.*)

\s__hook_mark

Scan mark used for delimited arguments.

```
\scan_new:N \s__hook_mark
```

(*End of definition for* \s__hook_mark.)

__hook_use_none_delimit_by_s_mark:w
__hook_use_i_delimit_by_s_mark:nw

Removes tokens until the next \s__hook_mark.

```
\cs_new:Npn \__hook_use_none_delimit_by_s_mark:w #1 \s__hook_mark { }
\cs_new:Npn \__hook_use_i_delimit_by_s_mark:nw #1 #2 \s__hook_mark {#1}
```

(*End of definition for* __hook_use_none_delimit_by_s_mark:w *and* __hook_use_i_delimit_by_s_mark:nw.)

__hook_tl_set:cn

Private copies of a few expl3 functions. l3debug will only add debugging to the public names, not to these copies, so we don't have to use \debug_suspend: and \debug_resume: everywhere.

Functions like __hook_tl_set:Nn have to be redefined, rather than copied because in expl3 they use __kernel_tl_(g)set:Nx, which is also patched by l3debug.

```
\cs_new_protected:Npn \__hook_tl_set:cn #1#2
  { \cs_set_nopar:cpx {#1} { \__kernel_exp_not:w {#2} } }
```

(*End of definition for* __hook_tl_set:cn.)

__hook_tl_gset:Nn
__hook_tl_gset:Nx
__hook_tl_gset:cn
__hook_tl_gset:co
__hook_tl_gset:cx

Same as above.

```
\cs_new_protected:Npn \__hook_tl_gset:Nn #1#2
  { \cs_gset_nopar:Npx #1 { \__kernel_exp_not:w {#2} } }
\cs_new_protected:Npn \__hook_tl_gset:Nx #1#2
  { \cs_gset_nopar:Npx #1 {#2} }
\cs_generate_variant:Nn \__hook_tl_gset:Nn { c, co }
\cs_generate_variant:Nn \__hook_tl_gset:Nx { c }
```

(*End of definition for* __hook_tl_gset:Nn.)

__hook_tl_gput_right:Nn
__hook_tl_gput_right:Ne
__hook_tl_gput_right:cn

Same as above.

```
\cs_new_protected:Npn \__hook_tl_gput_right:Nn #1#2
  { \__hook_tl_gset:Nx #1 { \__kernel_exp_not:w \exp_after:wN { #1 #2 } } }
\cs_generate_variant:Nn \__hook_tl_gput_right:Nn { Ne, cn }
```

(*End of definition for* `\__hook_tl_gput_right:Nn`.)

`\__hook_tl_gput_left:Nn` Same as above.

```
\cs_new_protected:Npn \__hook_tl_gput_left:Nn #1#2
  {
    \__hook_tl_gset:Nx #1
      { \__kernel_exp_not:w {#2} \__kernel_exp_not:w \exp_after:wN {#1} }
  }
```

(*End of definition for* `\__hook_tl_gput_left:Nn`.)

`\__hook_tl_gset_eq:NN` Same as above.

```
\cs_new_eq:NN \__hook_tl_gset_eq:NN \tl_gset_eq:NN
```

(*End of definition for* `\__hook_tl_gset_eq:NN`.)

`\__hook_tl_gclear:N`
`\__hook_tl_gclear:c` Same as above.

```
\cs_new_protected:Npn \__hook_tl_gclear:N #1
  { \__hook_tl_gset_eq:NN #1 \c_empty_tl }
\cs_generate_variant:Nn \__hook_tl_gclear:N { c }
```

(*End of definition for* `\__hook_tl_gclear:N`.)

4.4 Providing new hooks

4.4.1 The data structures of a hook

`\g_@@_⟨hook⟩_code_prop`
`\@@␣⟨hook⟩`
`\g_@@_⟨hook⟩_reversed_tl`
`\g_@@_⟨hook⟩_declared_tl`
`\g_@@_⟨hook⟩_parameter_tl`
`\@@_next␣⟨hook⟩`
`\@@_toplevel␣⟨hook⟩`

Hooks have a name (called ⟨*hook*⟩ in the description below) and for each hook we have to provide a number of data structures. These are

`\g__hook_⟨hook⟩_code_prop` A property list holding the code for the hook in separate chunks. The keys are by default the package names that add code to the hook, but it is possible for packages to define other keys.

`\g__hook_⟨hook⟩_rule_⟨label1⟩|⟨label2⟩_tl` A token list holding the relation between ⟨*label1*⟩ and ⟨*label2*⟩ in the ⟨*hook*⟩. The ⟨*labels*⟩ are lexically (reverse) sorted to ensure that two labels always point to the same token list. For global rules, the ⟨*hook*⟩ name is `??`.

`\__hook␣⟨hook⟩` The code that is actually executed when the hook is called in the document is stored in this token list. It is constructed from the code chunks applying the information. This token list is named like that so that in case of an error inside the hook, the reported token list in the error is shorter, and to make it simpler to normalize hook names in `\__hook_make_name:n`.

`\g__hook_⟨hook⟩_reversed_tl` Some hooks are "reversed". This token list stores a `-` for such hook so that it can be identified. The `-` character is used because ⟨*reversed*⟩1 is $+1$ for normal hooks and -1 for reversed ones.

`\g__hook_⟨hook⟩_declared_tl` This token list serves as a marker for the hook being officially declared. Its existence is tested to raise an error in case another declaration is attempted.

`\c__hook_⟨hook⟩_parameter_tl` This token list stores the parameter text for a declared hook (its existence almost completely intersects the token list above), which is used for managing hooks with arguments.

`\__hook_toplevel␣⟨hook⟩` This token list stores the code inserted in the hook from the user's document, in the `top-level` label. This label is special, and doesn't participate in sorting. Instead, all code is appended to it and executed after (or before, if the hook is reversed) the normal hook code, but before the `next` code chunk.

`\__hook_next␣⟨hook⟩` Finally there is extra code (normally empty) that is used on the next invocation of the hook (and then deleted). This can be used to define some special behavior for a single occasion from within the document. This token list follows the same naming scheme than the main `\__hook␣⟨hook⟩` token list. It is called `\__hook_next␣⟨hook⟩` rather than `\__hook␣next_⟨hook⟩` because otherwise a hook whose name is `next_⟨hook⟩` would clash with the next code-token list of the hook called ⟨*hook*⟩.

4.4.2 On the existence of hooks

A hook may be in different states of existence. Here we give an overview of the internal commands to set up hooks and explain how the different states are distinguished. The actual implementation then follows in subsequent sections.

One problem we have to solve is that we need to be able to add code to hooks (e.g., with `\AddToHook`) even if that code has not yet been declared. For example, one package needs to write into a hook of another package, but that package may not get loaded, or is loaded only later. Another problem is that most hooks, but not the generic hooks, require a declaration.

We therefore distinguish the following states for a hook, which are managed by four different tests: structure existence (`\__hook_if_structure_exist:nTF`), creation (`\__hook_if_usable:nTF`), declaration (`\__hook_if_declared:nTF`) and disabled or not (`\__hook_if_disabled:nTF`)

not existing Nothing is known about the hook so far. This state can be detected with `\__hook_if_structure_exist:nTF` (which uses the false branch).

In this state the hook can be declared, disabled, rules can be defined or code could be added to it, but it is not possible to use the hook (with `\UseHook`).

basic data structure set up A hook is this state when its basic data structure has been set up (using `\__hook_init_structure:n`). The data structure setup happens automatically when commands such as `\AddToHook` are used and the hook is at that point in state "not existing".

In this state the four tests give the following results:

`\__hook_if_structure_exist:nTF`	returns `true`.
`\__hook_if_usable:nTF`	returns `false`.
`\__hook_if_declared:nTF`	returns `false`.
`\__hook_if_disabled:nTF`	returns `false`.

The allowed actions are the same as in the "not existing" state.

declared A hook is in this state it is not disabled and was explicitly declared (e.g., with `\NewHook`). In this case the four tests give the following results:

`\__hook_if_structure_exist:nTF` returns `true`.
`\__hook_if_usable:nTF` returns `true`.
`\__hook_if_declared:nTF` returns `true`.
`\__hook_if_disabled:nTF` returns `false`.

usable A hook is in this state if it is not disabled, was not explicitly declared but nevertheless is allowed to be used (with `\UseHook` or `\hook_use:n`). This state is only possible for generic hooks as they do not need to be declared. Therefore such hooks move directly from state "not existing" to "usable" the moment a declaration such as `\AddToHook` wants to add to the hook data structure. In this state the tests give the following results:

`\__hook_if_structure_exist:nTF` returns `true`.
`\__hook_if_usable:nTF` returns `true`.
`\__hook_if_declared:nTF` returns `false`.
`\__hook_if_disabled:nTF` returns `false`.

disabled A generic hook in any state is moved to this state when `\DisableGenericHook` is used. This changes the tests to give the following results:

`\__hook_if_structure_exist:nTF` *unchanged*.
`\__hook_if_usable:nTF` returns `false`.
`\__hook_if_declared:nTF` returns `true`.
`\__hook_if_disabled:nTF` returns `true`.

The structure test is unchanged (if the hook was unknown before it is `false`, otherwise `true`). The usable test returns `false` so that any `\UseHook` will bypass the hook from now on. The declared test returns true so that any further `\NewHook` generates an error and the disabled test returns true so that `\AddToHook` can return an error.

FMi: maybe it should do this only after begin document?

4.4.3 Setting hooks up

`\hook_new:n`
`\hook_new_with_args:nn`
`\__hook_new:nn`

The `\hook_new:n` declaration declares a new hook and expects the hook ⟨*name*⟩ as its argument, e.g., `begindocument`.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_new_with_args:nn}
⟨latexrelease⟩                    {Hooks~with~args}
\cs_new_protected:Npn \hook_new:n #1
  { \__hook_normalize_hook_args:Nn \__hook_new:nn {#1} { 0 } }
\cs_new_protected:Npn \hook_new_with_args:nn #1 #2
  { \__hook_normalize_hook_args:Nn \__hook_new:nn {#1} {#2} }

\cs_new_protected:Npn \__hook_new:nn #1 #2
  {
```

We check if the hook was already *explicitly* declared with `\hook_new:n`, and if it already exists we complain, otherwise set the "created" flag for the hook so that it errors next time `\hook_new:n` is used.

```
    \__hook_if_declared:nTF {#1}
      { \msg_error:nnn { hooks } { exists } {#1} }
      {
        \tl_new:c { g__hook_#1_declared_tl }
        \cs_undefine:c { __hook~#1 }
        \cs_undefine:c { c__hook_#1_parameter_tl }
        \__hook_make_usable:nn {#1} {#2}
```

In case there is already code in a hook, but it's undeclared, run `\__hook_update_hook_code:n` to make it ready to be executed (see test `lthooks-034`).

```
        \__hook_update_hook_code:n {#1}
      }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_new_with_args:nn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \hook_new:n #1
⟨latexrelease⟩  { \__hook_normalize_hook_args:Nn \__hook_new:n {#1} }
⟨latexrelease⟩\cs_undefine:N \__hook_new:nn
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_new:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_declared:nTF {#1}
⟨latexrelease⟩      { \msg_error:nnn { hooks } { exists } {#1} }
⟨latexrelease⟩      {
⟨latexrelease⟩        \tl_new:c { g__hook_#1_declared_tl }
⟨latexrelease⟩        \__hook_make_usable:n {#1}
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_gset_protected:Npn \hook_new_with_args:nn #1 { }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_new:n`, `\hook_new_with_args:nn`, *and* `\__hook_new:nn`. *These functions are documented on page* 206.)

`\__hook_make_usable:nn` This initializes all hook data structures for the hook but if used on its own doesn't mark the hook as declared (as `\hook_new:n` does, so a later `\hook_new:n` on that hook will not result in an error. This command is internally used by `\hook_gput_code:nnn` when adding code to a generic hook.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_make_usable:nn}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_make_usable:nn #1 #2
  {
```

Now we check if the hook's data structure can be safely created without expl3 raising errors, then we add the hook name to the list of all hooks and allocate the necessary data structures for the new hook, otherwise just do nothing.

```
    \__hook_if_usable:nF {#1}
      {
        \seq_gput_right:Nn \g__hook_all_seq {#1}
```

Here we'll define the \c__hook_⟨*hook*⟩_parameter_tl to hold a run of parameters up to the number of arguments of the hook (#2).

```
      \__kernel_cs_parm_from_arg_count:nnF
        { \tl_const:cn { c__hook_#1_parameter_tl } } {#2}
        {
          \msg_error:nnnn { hooks } { too-many-args } {#1} {#2}
          \tl_const:cx { c__hook_#1_parameter_tl }
            { \exp_not:V \c__hook_nine_parameters_tl }
        }
```

After that, use __hook_normalise_cs_args:nn to correct the number of parameters of the macros __hook_toplevel␣⟨*hook*⟩ and __hook_next␣⟨*hook*⟩. We need to be able to add code with arguments to a hook without prior knowledge of the number of arguments of that hook, so lthooks assumes 9 until the hook is properly declared and the number of arguments is known. __hook_normalise_cs_args:nn does the normalisation by using the \c__hook_⟨*hook*⟩_parameter_tl defined just above.

```
      \__hook_normalise_cs_args:nn { _toplevel } {#1}
      \__hook_normalise_cs_args:nn { _next } {#1}
```

This is only used by the actual code of the current hook, so declare it normally:

```
      \__hook_code_gset:nn {#1} { }
```

Now ensure that the base data structure for the hook exists:

```
      \__hook_init_structure:n {#1}
```

The call to __hook_normalise_code_pool:n will correct any improper reference to arguments that don't exist in the hook, raising a low-level TEX error and doubling the offending parameter tokens. It has to be done after __hook_init_structure:n because it operates on \g__hook_⟨*hook*⟩_code_prop.

```
      \__hook_normalise_code_pool:n {#1}
```

The \g__hook_⟨*hook*⟩_labels_clist holds the sorted list of labels (once it got sorted). This is used only for debugging. These are defined conditionally, in case __hook_make_usable:nn is being used to redefine a hook.

```
      \clist_if_exist:cF { g__hook_#1_labels_clist }
        {
          \clist_new:c { g__hook_#1_labels_clist }
```

Some hooks should reverse the default order of code chunks. To signal this we have a token list which is empty for normal hooks and contains a - for reversed hooks.

```
          \tl_new:c { g__hook_#1_reversed_tl }
        }
```

The above is all in L3 convention, but we also provide an interface to legacy LaTeX 2ε hooks of the form \@...hook, e.g., \@begindocumenthook. there have been a few of them and they have been added to using \g@addto@macro. If there exists such a macro matching the name of the new hook, i.e., \@⟨*hook-name*⟩hook and it is not empty then we add its contents as a code chunk under the label legacy.

Warning: this support will vanish in future releases!

```
      \__hook_include_legacy_code_chunk:n {#1}
    }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_make_usable:nn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_make_usable:nn
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_make_usable:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_if_exist:cF { __hook~#1 }
⟨latexrelease⟩      {
⟨latexrelease⟩        \seq_gput_right:Nn \g__hook_all_seq {#1}
⟨latexrelease⟩        \tl_new:c { __hook~#1 }
⟨latexrelease⟩        \__hook_init_structure:n {#1}
⟨latexrelease⟩        \clist_new:c { g__hook_#1_labels_clist }
⟨latexrelease⟩        \tl_new:c { g__hook_#1_reversed_tl }
⟨latexrelease⟩        \__hook_include_legacy_code_chunk:n {#1}
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_make_usable:nn`.)

`\__hook_init_structure:n` This function declares the basic data structures for a hook without explicit declaring the hook itself. This is needed to allow adding to undeclared hooks. Here it is unnecessary to check whether all variables exist, since all three are declared at the same time (either all of them exist, or none).

It creates the hook code pool (`\g__hook_`⟨*hook*⟩`_code_prop`) and the `top-level` and `next` token lists. A hook is initialized with `\__hook_init_structure:n` the first time anything is added to it. Initializing a hook just with `\__hook_init_structure:n` will not make it usable with `\hook_use:n`.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_init_structure:n}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_init_structure:n #1
  {
    \__hook_if_structure_exist:nF {#1}
      {
        \prop_new:c { g__hook_#1_code_prop }
        \__hook_toplevel_gset:nn {#1} { }
        \__hook_next_gset:nn {#1} { }
      }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_init_structure:n}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_init_structure:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_structure_exist:nF {#1}
⟨latexrelease⟩      {
⟨latexrelease⟩        \prop_new:c { g__hook_#1_code_prop }
⟨latexrelease⟩        \tl_new:c { __hook_toplevel~#1 }
⟨latexrelease⟩        \tl_new:c { __hook_next~#1 }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_init_structure:n`.)

\hook_new_reversed:n
\hook_new_reversed_with_args:nn
__hook_new_reversed:nn

Declare a new hook. The default ordering of code chunks is reversed, signaled by setting the token list to a minus sign.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_new_reversed_with_args:nn}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \hook_new_reversed:n #1
  { \__hook_normalize_hook_args:Nn \__hook_new_reversed:nn {#1} { 0 } }
\cs_new_protected:Npn \hook_new_reversed_with_args:nn #1 #2
  { \__hook_normalize_hook_args:Nn \__hook_new_reversed:nn {#1} {#2} }
\cs_new_protected:Npn \__hook_new_reversed:nn #1 #2
  {
    \__hook_if_declared:nTF {#1}
      { \msg_error:nnn { hooks } { exists } {#1} }
      {
        \__hook_new:nn {#1} {#2}
        \tl_gset:cn { g__hook_#1_reversed_tl } { - }
      }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_new_reversed_with_args:nn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \hook_new_reversed:n #1
⟨latexrelease⟩  { \__hook_normalize_hook_args:Nn \__hook_new_reversed:n {#1} }
⟨latexrelease⟩\cs_undefine:N \__hook_new_reversed:nn
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_new_reversed:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_new:n {#1}
⟨latexrelease⟩    \tl_gset:cn { g__hook_#1_reversed_tl } { - }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_undefine:N \__hook_new_reversed:nn
⟨latexrelease⟩\cs_gset_protected:Npn \hook_new_reversed_with_args:nn #1 #2 { }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_new_reversed:n`, `\hook_new_reversed_with_args:nn`, *and* `\__hook_new_reversed:nn`. *These functions are documented on page* 206.)

\hook_new_pair:nn
\hook_new_pair_with_args:nnn

A shorthand for declaring a normal and a (matching) reversed hook in one go.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_new_pair_with_args:nnn}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \hook_new_pair:nn #1#2
  { \__hook_normalize_hook_args:Nnn \__hook_new_pair:nnn {#1} {#2} { 0 } }
\cs_new_protected:Npn \hook_new_pair_with_args:nnn #1#2#3
  { \__hook_normalize_hook_args:Nnn \__hook_new_pair:nnn {#1} {#2} {#3} }
\cs_new_protected:Npn \__hook_new_pair:nnn #1 #2 #3
  {
    \__hook_if_declared:nTF {#1}
      { \msg_error:nnn { hooks } { exists } {#1} }
      {
        \__hook_if_declared:nTF {#2}
          { \msg_error:nnn { hooks } { exists } {#2} }
          {
            \__hook_new:nn {#1} {#3}
            \__hook_new_reversed:nn {#2} {#3}
          }
      }
```

```
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_new_pair_with_args:nnn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \hook_new_pair:nn #1#2
⟨latexrelease⟩  {
⟨latexrelease⟩    \hook_new:n {#1}
⟨latexrelease⟩    \hook_new_reversed:n {#2}
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_gset_protected:Npn \hook_new_pair_with_args:nnn #1#2#3
⟨latexrelease⟩  { }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_new_pair:nn` *and* `\hook_new_pair_with_args:nnn`*. These functions are documented on page 206.*)

`\__hook_include_legacy_code_chunk:n`

The LaTeX legacy concept for hooks uses with hooks the following naming scheme in the code: `\@...hook`.

If this macro is not empty we add it under the label `legacy` to the current hook and then empty it globally. This way packages or classes directly manipulating commands such as `\@begindocumenthook` still get their hook data added.

Warning: this support will vanish in future releases!

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_include_legacy_code_chunk:n}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_include_legacy_code_chunk:n #1
  {
```

If the macro doesn't exist (which is the usual case) then nothing needs to be done.

```
    \tl_if_exist:cT { @#1hook }
      {
```

Of course if the legacy hook exists but is empty, there is no need to add anything under `legacy` the legacy label.

```
        \tl_if_empty:cF { @#1hook }
          {
```

Here we set `\__hook_replacing_args_false:` because no legacy code will reference hook arguments.

```
            \__hook_replacing_args_false:
            \use:e
              {
                \__hook_hook_gput_code_do:nnn {#1} { legacy }
                  { \exp_not:v { @#1hook } }
              }
            \__hook_replacing_args_reset:
```

Once added to the hook, we need to clear it otherwise it might get added again later if the hook data gets updated.

```
            \__hook_tl_gclear:c { @#1hook }
          }
      }
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_include_legacy_code_chunk:n}
```

```
⟨latexrelease⟩                    {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_include_legacy_code_chunk:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_if_exist:cT { @#1hook }
⟨latexrelease⟩      {
⟨latexrelease⟩        \tl_if_empty:cF { @#1hook }
⟨latexrelease⟩          {
⟨latexrelease⟩            \exp_args:Nnnv \__hook_hook_gput_code_do:nnn
⟨latexrelease⟩              {#1} { legacy } { @#1hook }
⟨latexrelease⟩            \__hook_tl_gclear:c { @#1hook }
⟨latexrelease⟩          }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_include_legacy_code_chunk:n`.)

4.4.4 Disabling and providing hooks

`\hook_disable_generic:n`
`\__hook_disable:n`
`\__hook_if_disabled_p:n`
`\__hook_if_disabled:nTF`

Disables a hook by creating its `\g__hook_`⟨*hook*⟩`_declared_tl` so that the hook errors when used with `\hook_new:n`, then it undefines `\__hook␣`⟨*hook*⟩ so that it may not be executed.

This does not clear any code that may be already stored in the hook's structure, but doesn't allow adding more code. `\__hook_if_disabled:nTF` uses that specific combination to check if the hook is disabled.

```
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\hook_disable_generic:n}
⟨latexrelease⟩                    {Disable~hooks}
```

```
\cs_new_protected:Npn \hook_disable_generic:n #1
  { \__hook_normalize_hook_args:Nn \__hook_disable:n {#1} }
\cs_new_protected:Npn \__hook_disable:n #1
  {
    \tl_gclear_new:c { g__hook_#1_declared_tl }
    \cs_undefine:c { __hook~#1 }
  }
\prg_new_conditional:Npnn \__hook_if_disabled:n #1 { p, T, F, TF }
  {
    \bool_lazy_and:nnTF
        { \tl_if_exist_p:c { g__hook_#1_declared_tl } }
        { ! \cs_if_exist_p:c { __hook~#1 } }
      { \prg_return_true: }
      { \prg_return_false: }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_disable_generic:n}
⟨latexrelease⟩                    {Disable~hooks}
⟨latexrelease⟩
⟨latexrelease⟩\cs_new_protected:Npn \hook_disable_generic:n #1 {}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_disable_generic:n`, `\__hook_disable:n`, *and* `\__hook_if_disabled:nTF`. *This function is documented on page* 207.)

\hook_activate_generic:n
__hook_activate_generic:n

The \hook_activate_generic:n declaration declares a new hook if it wasn't declared already, in which case it only checks that the already existing hook is not a reversed hook.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_activate_generic:n}
⟨latexrelease⟩                  {Providing~hooks}
```

```
\cs_new_protected:Npn \hook_activate_generic:n #1
  { \__hook_normalize_hook_args:Nn \__hook_activate_generic:nn {#1} {   } }
```

```
\cs_new_protected:Npn \__hook_activate_generic:nn #1 #2
  {
```

If the hook to be activated was disabled we warn (for now — this may change).

```
    \__hook_if_disabled:nTF {#1}
      { \msg_warning:nnn { hooks } { activate-disabled } {#1} }
```

Otherwise we check if the hook is not declared, and if it isn't, figure out if it's reversed or not, then declare it accordingly.

```
      {
        \__hook_if_declared:nF {#1}
          {
            \tl_new:c { g__hook_#1_declared_tl }
            \__hook_make_usable:nn {#1} { 0 }
            \tl_gset:cx { g__hook_#1_reversed_tl }
              { \__hook_if_generic_reversed:nT {#1} { - } }
```

Reflect that we have activated the generic hook and set its execution code.

```
            \__hook_update_hook_code:n {#1}
          }
      }
  }
```

```
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\hook_activate_generic:n}
⟨latexrelease⟩                  {Providing~hooks}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_activate_generic:nn #1 #2
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_disabled:nTF {#1}
⟨latexrelease⟩      { \msg_warning:nnn { hooks } { activate-disabled } {#1} }
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_if_declared:nF {#1}
⟨latexrelease⟩          {
⟨latexrelease⟩            \tl_new:c { g__hook_#1_declared_tl }
⟨latexrelease⟩            \__hook_make_usable:n {#1}
⟨latexrelease⟩            \tl_gset:cx { g__hook_#1_reversed_tl }
⟨latexrelease⟩              { \__hook_if_generic_reversed:nT {#1} { - } }
⟨latexrelease⟩            \__hook_update_hook_code:n {#1}
⟨latexrelease⟩          }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_activate_generic:n}
⟨latexrelease⟩                  {Providing~hooks}
⟨latexrelease⟩\cs_gset_protected:Npn \hook_activate_generic:n #1 { }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_activate_generic:n` *and* `\__hook_activate_generic:n`*. This function is documented on page* 207.)

4.5 Parsing a label

`\__hook_parse_label_default:nN`

This macro checks if a label was given (not `\c_novalue_tl`), and if so, tries to parse the label looking for a leading `.` to replace by `\__hook_currname_or_default:`. `#2` is a boolean representing if `#1` is a label name.

```
\cs_new:Npn \__hook_parse_label_default:nN #1#2
  {
    \tl_if_novalue:nTF {#1}
      { \__hook_currname_or_default: }
      { \tl_trim_spaces_apply:nN {#1} \__hook_parse_dot_label:nN #2 }
  }
```

(*End of definition for* `\__hook_parse_label_default:nN`.)

`\__hook_parse_dot_label:nN`
`\__hook_parse_dot_label:w`
`\__hook_parse_dot_label_cleanup:w`
`\__hook_parse_dot_label_aux:w`

Start by checking if the label is empty, which raises an error, and uses the fallback value. If not, split the label at a `./`, if any, and check if no tokens are before the `./`, or if the only character is a `.`. If these requirements are fulfilled, the leading `.` is replaced with `\__hook_currname_or_default:`. Otherwise the label is returned unchanged. `#2` is a boolean representing if `#1` is a label name.

```
\cs_new:Npn \__hook_parse_dot_label:nN #1#2
  {
    \tl_if_empty:nTF {#1}
      {
        \bool_if:NTF #2
          { \msg_expandable_error:nn { hooks } { empty-label } }
          { \msg_expandable_error:nn { hooks } { empty-hook } }
        \__hook_currname_or_default:
      }
      {
        \str_if_eq:nnTF {#1} { . }
          { \__hook_currname_or_default: }
          { \__hook_parse_dot_label:w #1 ./ \s__hook_mark }
      }
  }
\cs_new:Npn \__hook_parse_dot_label:w #1 ./ #2 \s__hook_mark
  {
    \tl_if_empty:nTF {#1}
      { \__hook_parse_dot_label_aux:w #2 \s__hook_mark }
      {
        \tl_if_empty:nTF {#2}
          { \__hook_make_name:n {#1} }
          { \__hook_parse_dot_label_cleanup:w #1 ./ #2 \s__hook_mark }
      }
  }
\cs_new:Npn \__hook_parse_dot_label_cleanup:w #1 ./ \s__hook_mark {#1}
\cs_new:Npn \__hook_parse_dot_label_aux:w #1 ./ \s__hook_mark
  { \__hook_currname_or_default: / \__hook_make_name:n {#1} }
```

(*End of definition for* `\__hook_parse_dot_label:nN` *and others.*)

__hook_currname_or_default:

This uses \g__hook_hook_curr_name_tl if it is set, otherwise it tries \@currname. If neither is set, it raises an error and uses the fallback value label-missing.

```
\cs_new:Npn \__hook_currname_or_default:
  {
    \tl_if_empty:NTF \g__hook_hook_curr_name_tl
      {
        \tl_if_empty:NTF \@currname
          {
            \msg_expandable_error:nnn { latex2e } { should-not-happen }
              { Empty~default~label. }
            \__hook_make_name:n { label-missing }
          }
          { \@currname }
      }
      { \g__hook_hook_curr_name_tl }
  }
```

(*End of definition for* __hook_currname_or_default:.)

__hook_make_name:n
__hook_make_name:w

This provides a standard sanitization of a hook's name. It uses \cs:w to build a control sequence out of the hook name, then uses \cs_to_str:N to get the string representation of that, without the escape character. \cs:w-based expansion is used instead of e-based because Unicode characters don't behave well inside \expanded. The macro adds the __hook␣ prefix to the hook name to reuse the hook's code token list to build the csname and avoid leaving "public" control sequences defined (as \relax) in TeX's memory.

```
\cs_new:Npn \__hook_make_name:n #1
  {
    \exp_after:wN \exp_after:wN \exp_after:wN \__hook_make_name:w
    \exp_after:wN \token_to_str:N \cs:w __hook~ #1 \cs_end:
  }
\exp_last_unbraced:NNNNo
\cs_new:Npn \__hook_make_name:w #1 \tl_to_str:n { __hook~ } { }
```

(*End of definition for* __hook_make_name:n *and* __hook_make_name:w.)

__hook_normalize_hook_args:Nn
__hook_normalize_hook_args:Nnn
__hook_normalize_hook_rule_args:Nnnnn
__hook_normalize_hook_args_aux:Nn

This is the standard route for normalizing hook and label arguments. The main macro does the entire operation within a group so that csnames made by __hook_make_name:n are wiped off before continuing. This means that this function cannot be used for \hook_use:n!

```
\cs_new_protected:Npn \__hook_normalize_hook_args_aux:Nn #1 #2
  {
    \group_begin:
    \use:e
      {
        \group_end:
        \exp_not:N #1 #2
      }
  }
\cs_new_protected:Npn \__hook_normalize_hook_args:Nn #1 #2
  {
    \__hook_normalize_hook_args_aux:Nn #1
      { { \__hook_parse_label_default:nN {#2} \c_false_bool } }
  }
```

```
\cs_new_protected:Npn \__hook_normalize_hook_args:Nnn #1 #2 #3
  {
    \__hook_normalize_hook_args_aux:Nn #1
      {
        { \__hook_parse_label_default:nN {#2} \c_false_bool }
        { \__hook_parse_label_default:nN {#3} \c_true_bool }
      }
  }
\cs_new_protected:Npn \__hook_normalize_hook_rule_args:Nnnnn #1 #2 #3 #4 #5
  {
    \__hook_normalize_hook_args_aux:Nn #1
      {
        { \__hook_parse_label_default:nN {#2} \c_false_bool }
        { \__hook_parse_label_default:nN {#3} \c_true_bool }
        { \tl_trim_spaces:n {#4} }
        { \__hook_parse_label_default:nN {#5} \c_true_bool }
      }
  }
```

(*End of definition for* `\__hook_normalize_hook_args:Nn` *and others.*)

`\__hook_curr_name_push:n`
`\__hook_curr_name_push_aux:n`
`\__hook_curr_name_pop:`
`\__hook_end_document_label_check:`

The token list `\g__hook_hook_curr_name_tl` stores the name of the current package/file to be used as the default label in hooks. Providing a consistent interface is tricky because packages can be loaded within packages, and some packages may not use `\SetDefaultHookLabel` to change the default label (in which case `\@currname` is used).

To pull that one off, we keep a stack that contains the default label for each level of input. The bottom of the stack contains the default label for the `top-level` (this stack should never go empty). If we're building the format, set the default label to be `top-level`:

```
\tl_gset:Nn \g__hook_hook_curr_name_tl { top-level }
```

Then, in case we're in latexrelease we push something on the stack to support roll forward. But in some rare cases, latexrelease may be loaded inside another package (notably platexrelease), so we'll first push the `top-level` entry:

```
⟨latexrelease⟩\seq_if_empty:NT \g__hook_name_stack_seq
⟨latexrelease⟩  { \seq_gput_right:Nn \g__hook_name_stack_seq { top-level } }
```

then we dissect the `\@currnamestack`, adding `\@currname` to the stack:

```
⟨latexrelease⟩\cs_set_protected:Npn \__hook_tmp:w #1 #2 #3
⟨latexrelease⟩  {
⟨latexrelease⟩    \quark_if_recursion_tail_stop:n {#1}
⟨latexrelease⟩    \seq_gput_right:Nn \g__hook_name_stack_seq {#1}
⟨latexrelease⟩    \__hook_tmp:w
⟨latexrelease⟩  }
⟨latexrelease⟩\exp_after:wN \__hook_tmp:w \@currnamestack
⟨latexrelease⟩  \q_recursion_tail \q_recursion_tail
⟨latexrelease⟩  \q_recursion_tail \q_recursion_stop
```

and finally set the default label to be the `\@currname`:

```
⟨latexrelease⟩\tl_gset:Nx \g__hook_hook_curr_name_tl { \@currname }
⟨latexrelease⟩\seq_gpop_right:NN \g__hook_name_stack_seq \l__hook_tmpa_tl
```

Two commands keep track of the stack: when a file is input, `\__hook_curr_name_push:n` pushes the current default label onto the stack and sets the new default label (all in one go):

```
\cs_new_protected:Npn \__hook_curr_name_push:n #1
  { \exp_args:Nx \__hook_curr_name_push_aux:n { \__hook_make_name:n {#1} } }
\cs_new_protected:Npn \__hook_curr_name_push_aux:n #1
  {
    \tl_if_blank:nTF {#1}
      { \msg_error:nn { hooks } { no-default-label } }
      {
        \str_if_eq:nnTF {#1} { top-level }
          {
            \msg_error:nnnnn { hooks } { set-top-level }
              { to } { PushDefaultHookLabel } {#1}
          }
          {
            \seq_gpush:NV \g__hook_name_stack_seq \g__hook_hook_curr_name_tl
            \tl_gset:Nn \g__hook_hook_curr_name_tl {#1}
          }
      }
  }
```

and when an input is over, the topmost item of the stack is popped, since that label will not be used again, and `\g__hook_hook_curr_name_tl` is updated to equal the now topmost item of the stack:

```
\cs_new_protected:Npn \__hook_curr_name_pop:
  {
    \seq_gpop:NNTF \g__hook_name_stack_seq \l__hook_return_tl
      { \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl }
      { \msg_error:nn { hooks } { extra-pop-label } }
  }
```

At the end of the document we want to check if there was no `\__hook_curr_name_push:n` without a matching `\__hook_curr_name_pop:` (not a critical error, but it might indicate that something else is not quite right):

```
\tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
  { \__hook_end_document_label_check: }
\cs_new_protected:Npn \__hook_end_document_label_check:
  {
    \seq_gpop:NNT \g__hook_name_stack_seq \l__hook_return_tl
      {
        \msg_error:nnx { hooks } { missing-pop-label }
          { \g__hook_hook_curr_name_tl }
        \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl
        \__hook_end_document_label_check:
      }
  }
```

The token list `\g__hook_hook_curr_name_tl` is but a mirror of the top of the stack.

Now define a wrapper that replaces the top of the stack with the argument, and updates `\g__hook_hook_curr_name_tl` accordingly.

```
\cs_new_protected:Npn \__hook_set_default_hook_label:n #1
  {
    \seq_if_empty:NTF \g__hook_name_stack_seq
      {
        \msg_error:nnnnn { hooks } { set-top-level }
          { for } { SetDefaultHookLabel } {#1}
```

`\__hook_set_default_hook_label:n`
`\__hook_set_default_label:n`

```
      }
      { \exp_args:Nx
        \__hook_set_default_label:n { \__hook_make_name:n {#1} } }
  }
\cs_new_protected:Npn \__hook_set_default_label:n #1
  {
    \str_if_eq:nnTF {#1} { top-level }
      {
        \msg_error:nnnnn { hooks } { set-top-level }
          { to } { SetDefaultHookLabel } {#1}
      }
      { \tl_gset:Nn \g__hook_hook_curr_name_tl {#1} }
  }
```

(*End of definition for* `\__hook_curr_name_push:n` *and others.*)

4.6 Adding or removing hook code

`\hook_gput_code:nnn`
`\hook_gput_code_with_args:nnn`
`\__hook_gput_code:nnn`
`\__hook_gput_code_store:nnn`
`\__hook_hook_gput_code_do:nnn`
`\__hook_prop_gput_labeled_cleanup:nnn`
`\__hook_prop_gput_labeled_do:Nnnn`

With `\hook_gput_code:nnn{`⟨*hook*⟩`}{`⟨*label*⟩`}{`⟨*code*⟩`}` a chunk of ⟨*code*⟩ is added to an existing ⟨*hook*⟩ labeled with ⟨*label*⟩.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_gput_code:nnn}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \hook_gput_code:nnn #1 #2 #3
  {
    \__hook_replacing_args_false:
    \__hook_normalize_hook_args:Nnn \__hook_gput_code:nnn {#1} {#2} {#3}
    \__hook_replacing_args_reset:
  }
\cs_new_protected:Npn \hook_gput_code_with_args:nnn #1 #2 #3
  {
    \__hook_replacing_args_true:
    \__hook_normalize_hook_args:Nnn \__hook_gput_code:nnn {#1} {#2} {#3}
    \__hook_replacing_args_reset:
  }
```

If `\AddToHookWithArguments` was used, do some sanity checking, and if it's not possible to use arguments at this point, fall back to regular `\AddToHook` by using `\__hook_replacing_args_false:`.

```
\cs_new_protected:Npn \__hook_gput_code:nnn #1 #2 #3
  {
    \__hook_chk_args_allowed:nn {#1} { AddToHook }
```

Then check if the code should be executed immediately, rather than stored:

```
    \__hook_if_execute_immediately:nTF {#1}
      {
```

`\AddToHookWithArguments` can't be used on one-time hooks (that were already used).

```
        \__hook_if_replacing_args:TF
          {
            \msg_error:nnnn { hooks } { one-time-args }
              {#1} { AddToHook }
          }
          { }
        \use:n
      }
```

```
      { \__hook_gput_code_store:nnn {#1} {#2} }
          {#3}
  }
\cs_new_protected:Npn \__hook_gput_code_store:nnn #1 #2 #3
  {
```

Then check if the hook is usable.

```
    \__hook_if_usable:nTF {#1}
```

If so we simply add (or append) the new code to the property list holding different chunks for the hook. At `\begin{document}` this is then sorted into a token list for fast execution.

```
      {
        \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}
```

However, if there is an update within the document we need to alter this execution code which is done by `\__hook_update_hook_code:n`. In the preamble this does nothing.

```
        \__hook_update_hook_code:n {#1}
      }
```

If the hook is not usable, before giving up, check if it's not disabled and otherwise try to declare it as a generic hook, if its name matches one of the valid patterns.

```
      {
        \__hook_if_disabled:nTF {#1}
          { \msg_error:nnn { hooks } { hook-disabled } {#1} }
          { \__hook_try_declaring_generic_hook:nnn {#1} {#2} {#3} }
      }
  }
```

This macro will unconditionally add a chunk of code to the given hook.

```
\cs_new_protected:Npn \__hook_hook_gput_code_do:nnn #1 #2 #3
  {
```

However, first some debugging info if debugging is enabled:

```
    \__hook_debug:n{\iow_term:x{****~ Add~ to~
                        \__hook_if_usable:nF {#1} { undeclared~ }
                        hook~ #1~ (#2)
                        \on@line\space <-~ \tl_to_str:n{#3}} }
```

Then try to get the code chunk labeled `#2` from the hook. If there's code already there, then append `#3` to that, otherwise just put `#3`. If the current label is `top-level`, the code is added to a dedicated token list `\__hook_toplevel`␣⟨*hook*⟩ that goes at the end of the hook (or at the beginning, for a reversed hook), just before `\__hook_next`␣⟨*hook*⟩.

```
    \str_if_eq:nnTF {#2} { top-level }
      {
        \str_if_eq:eeTF { top-level } { \__hook_currname_or_default: }
          {
```

If the hook's basic structure does not exist, we need to declare it with `\__hook_init_structure:n`.

```
            \__hook_init_structure:n {#1}
```

Then append to the `_toplevel` container for the hook.

```
            \__hook_cs_gput_right:nnn { _toplevel } {#1} {#3}
          }
          { \msg_error:nnn { hooks } { misused-top-level } {#1} }
      }
      {
```

When adding to the code pool, we have to double hashes if `\AddToHook` was used (`replacing_args` is false), so that later it is turned into a single parameter token, rather than a parameter to the hook macro.

```
        \exp_args:Nx \__hook_prop_gput_labeled_cleanup:nnn
          {
            \__hook_if_replacing_args:TF
              { \exp_not:n }
              { \__hook_double_hashes:n }
                {#3}
          }
          {#1} {#2}
      }
  }
```

Adds code to a hook's code pool.

```
\cs_new_protected:Npn \__hook_prop_gput_labeled_cleanup:nnn #1 #2 #3
  {
    \tl_set:Nn \l__hook_return_tl {#1}
    \__hook_if_replacing_args:TF
      {
        \__hook_if_usable:nT {#2}
          {
            \__hook_set_normalise_fn:nn {#2}
              { Invalid~code~added~\msg_line_context: }
            \__hook_normalise_fn:nn {#3} {#1}
            \prop_get:NnN \l__hook_work_prop {#3} \l__hook_return_tl
          }
      }
      { }
    \exp_args:NcV \__hook_prop_gput_labeled_do:Nnn
      { g__hook_#2_code_prop } \l__hook_return_tl {#3}
  }
\cs_new_protected:Npn \__hook_prop_gput_labeled_do:Nnn #1 #2 #3
  {
    \prop_get:NnNTF #1 {#3} \l__hook_return_tl
      { \prop_gput:Nno #1 {#3} { \l__hook_return_tl #2 } }
      { \prop_gput:Nnn #1 {#3} {#2} }
  }
```

```
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_gput_code:nnn}
⟨latexrelease⟩                  {Providing~hooks}
⟨latexrelease⟩\cs_gset_protected:Npn \hook_gput_code:nnn #1 #2
⟨latexrelease⟩  { \__hook_normalize_hook_args:Nnn
⟨latexrelease⟩        \__hook_gput_code:nnn {#1} {#2} }
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_gput_code:nnn #1 #2 #3
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_execute_immediately:nTF {#1}
⟨latexrelease⟩      {#3}
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_if_usable:nTF {#1}
⟨latexrelease⟩          {
⟨latexrelease⟩            \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}
⟨latexrelease⟩            \__hook_update_hook_code:n {#1}
```

```
⟨latexrelease⟩          }
⟨latexrelease⟩          {
⟨latexrelease⟩            \__hook_if_disabled:nTF {#1}
⟨latexrelease⟩              { \msg_error:nnn { hooks } { hook-disabled } {#1} }
⟨latexrelease⟩              { \__hook_try_declaring_generic_hook:nnn
⟨latexrelease⟩                  {#1} {#2} {#3} }
⟨latexrelease⟩          }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_hook_gput_code_do:nnn #1 #2 #3
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_debug:n{\iow_term:x{****~ Add~ to~
⟨latexrelease⟩                      \__hook_if_usable:nF {#1} { undeclared~ }
⟨latexrelease⟩                      hook~ #1~ (#2)
⟨latexrelease⟩                      \on@line\space <-~ \tl_to_str:n{#3}} }
⟨latexrelease⟩    \str_if_eq:nnTF {#2} { top-level }
⟨latexrelease⟩      {
⟨latexrelease⟩        \str_if_eq:eeTF { top-level }
⟨latexrelease⟩                        { \__hook_currname_or_default: }
⟨latexrelease⟩          {
⟨latexrelease⟩            \__hook_init_structure:n {#1}
⟨latexrelease⟩            \__hook_tl_gput_right:cn { __hook_toplevel~#1 } {#3}
⟨latexrelease⟩          }
⟨latexrelease⟩          { \msg_error:nnn { hooks } { misused-top-level } {#1} }
⟨latexrelease⟩      }
⟨latexrelease⟩      {
⟨latexrelease⟩        \prop_get:cnNTF
⟨latexrelease⟩          { g__hook_#1_code_prop } {#2} \l__hook_return_tl
⟨latexrelease⟩          {
⟨latexrelease⟩            \prop_gput:cno { g__hook_#1_code_prop } {#2}
⟨latexrelease⟩              { \l__hook_return_tl #3 }
⟨latexrelease⟩          }
⟨latexrelease⟩          { \prop_gput:cnn { g__hook_#1_code_prop } {#2} {#3} }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_gset_protected:Npn \hook_gput_code_with_args:nnn #1#2#3 { }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_gput_code:nnn` *and others. These functions are documented on page* 208.)

`\__hook_chk_args_allowed:nn` This macro checks if it is possible to add code with references to a hook's arguments for hook `#1`. It only does something if the function being run is `replacing_args`. This macro will error if the hook is declared and takes no arguments, then it will set `\__hook_replacing_args_false:` so that the macro which called it will add the code normally.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_chk_args_allowed:nn}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_chk_args_allowed:nn #1 #2
  {
    \__hook_if_replacing_args:TF
      {
        \__hook_if_declared:nT {#1}
          { \tl_if_empty:cT { c__hook_#1_parameter_tl } { \use_ii:nn } }
        \use_none:n
```

```
          {
            \msg_error:nnnn { hooks } { without-args } {#1} {#2}
            \__hook_replacing_args_false:
          }
      }
      { }
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_chk_args_allowed:nn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_chk_args_allowed:nn
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_chk_args_allowed:nn`.)

`\__hook_gput_undeclared_hook:nnn`

Often it may happen that a package A defines a hook `foo`, but package B, that adds code to that hook, is loaded before A. In such case we need to add code to the hook before its declared. An implicitly declared hook doesn't have arguments (in principle), so use `\c_false_bool` here.

```
\cs_new_protected:Npn \__hook_gput_undeclared_hook:nnn #1 #2 #3
  {
    \__hook_init_structure:n {#1}
    \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}
  }
```

(*End of definition for* `\__hook_gput_undeclared_hook:nnn`.)

`\__hook_try_declaring_generic_hook:nnn`
`\__hook_try_declaring_generic_next_hook:nn`

These entry-level macros just pass the arguments along to the common `\__hook_try_declaring_generic_hook:nNNnn` with the right functions to execute when some action is to be taken.

The wrapper `\__hook_try_declaring_generic_hook:nnn` then defers `\hook_gput_code:nnn` if the generic hook was declared, or to `\__hook_gput_undeclared_hook:nnn` otherwise (the hook was tested for existence before, so at this point if it isn't generic, it doesn't exist).

The wrapper `\__hook_try_declaring_generic_next_hook:nn` for next-execution hooks does the same: it defers the code to `\hook_gput_next_code:nn` if the generic hook was declared, or to `\__hook_gput_next_do:nn` otherwise.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}
⟨latexrelease⟩                  {\__hook_try_declaring_generic_hook:nnn}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_try_declaring_generic_hook:nnn #1
  {
    \__hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop: {#1}
      \__hook_gput_code:nnn
      \__hook_gput_undeclared_hook:nnn
        {#1}
  }
\cs_new_protected:Npn \__hook_try_declaring_generic_next_hook:nn #1
  {
    \__hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop: {#1}
      \__hook_gput_next_code:nn
      \__hook_gput_next_do:nn
        {#1}
```

```
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/11/15}
⟨latexrelease⟩                  {\__hook_try_declaring_generic_hook:nnn}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_try_declaring_generic_hook:nnn #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop:
⟨latexrelease⟩      {#1}
⟨latexrelease⟩      \hook_gput_code:nnn
⟨latexrelease⟩      \__hook_gput_undeclared_hook:nnn
⟨latexrelease⟩        {#1}
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_gset_protected:Npn
⟨latexrelease⟩  \__hook_try_declaring_generic_next_hook:nn #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop:
⟨latexrelease⟩      {#1}
⟨latexrelease⟩      \hook_gput_next_code:nn
⟨latexrelease⟩      \__hook_gput_next_do:nn
⟨latexrelease⟩        {#1}
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}
⟨latexrelease⟩                  {\__hook_try_declaring_generic_hook:nnn}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\cs_new_protected:Npn
⟨latexrelease⟩  \__hook_try_declaring_generic_hook:nnn #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_try_declaring_generic_hook:nNNnn {#1}
⟨latexrelease⟩      \hook_gput_code:nnn \__hook_gput_undeclared_hook:nnn
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new_protected:Npn
⟨latexrelease⟩  \__hook_try_declaring_generic_next_hook:nn #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_try_declaring_generic_hook:nNNnn {#1}
⟨latexrelease⟩      \hook_gput_next_code:nn \__hook_gput_next_do:nn
⟨latexrelease⟩  }
```

(*End of definition for* `\__hook_try_declaring_generic_hook:nnn` *and* `\__hook_try_declaring_generic_next_hook:nn`.)

`\__hook_try_declaring_generic_hook:nNNnn`
`\__hook_try_declaring_generic_hook_split:nNNnn`

`\__hook_try_declaring_generic_hook:nNNnn` now splits the hook name at the first `/` (if any) and first checks if it is a file-specific hook (they require some normalization) using `\__hook_if_file_hook:wTF`. If not then check it is one of a predefined set for generic names. We also split off the second component to see if we have to make a reversed hook. In either case the function returns ⟨*true*⟩ for a generic hook and ⟨*false*⟩ in other cases.

```
⟨latexrelease⟩\cs_new_protected:Npn \__hook_try_declaring_generic_hook:nNNnn #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_file_hook:wTF #1 / / \s__hook_mark
⟨latexrelease⟩      {
⟨latexrelease⟩        \exp_args:Ne
⟨latexrelease⟩        \__hook_try_declaring_generic_hook_split:nNNnn
⟨latexrelease⟩          { \exp_args:Ne \__hook_file_hook_normalize:n {#1} }
```

```
⟨latexrelease⟩       }
⟨latexrelease⟩       { \__hook_try_declaring_generic_hook_split:nNNnn {#1} }
⟨latexrelease⟩   }
⟨latexrelease⟩\cs_new_protected:Npn
⟨latexrelease⟩     \__hook_try_declaring_generic_hook_split:nNNnn #1 #2 #3
⟨latexrelease⟩   {
⟨latexrelease⟩     \__hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop:
⟨latexrelease⟩       {#1}
⟨latexrelease⟩       { #2 }
⟨latexrelease⟩       { #3 } {#1}
⟨latexrelease⟩   }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_try_declaring_generic_hook:nNNnn` *and* `\__hook_try_declaring_generic_hook_split:nNNnn`.)

`\__hook_try_declaring_generic_hook:wnTF`

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}
⟨latexrelease⟩                  {\__hook_try_declaring_generic_hook:wn}
⟨latexrelease⟩                  {Hooks~with~args}
\prg_new_protected_conditional:Npnn
    \__hook_try_declaring_generic_hook:wn
    #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
  {
    \__hook_if_generic:nTF {#5}
      {
        \__hook_if_usable:nF {#5}
          {
```

If the hook doesn't exist yet we check if it is a `cmd` hook and if so we attempt patching the command in addition to declaring the hook.

For some commands this will not be possible, in which case `\__hook_patch_cmd_-or_delay:Nnn` (defined in `ltcmdhooks`) will generate an appropriate error message.

```
            \str_if_eq:nnT {#1} { cmd }
              {
                \__hook_try_put_cmd_hook:n {#5}
                \__hook_make_usable:nn {#5} { 9 }
                \use_none:nnn
              }
```

Declare the hook always even if it can't really be used (error message generated elsewhere).

Here we use `\__hook_make_usable:nn`, so that a `\hook_new:n` is still possible later. Generic hooks (except `cmd` hooks) take no arguments, so use zero as the second argument.

```
            \__hook_make_usable:nn {#5} { 0 }
          }
        \__hook_if_generic_reversed:nT {#5}
          { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
        \prg_return_true:
      }
      {
```

Generic hooks are all named ⟨*type*⟩/⟨*name*⟩/⟨*place*⟩, where ⟨*type*⟩ and ⟨*place*⟩ are predefined (`\c__hook_generic_`⟨*type*⟩`/./`⟨*place*⟩`_tl`), and ⟨*name*⟩ is the variable

component. Older releases had some hooks with the ⟨*name*⟩ in the third part, so the code below supports that syntax for a while, with a warning.

The `\exp_after:wN ... \exp:w` trick is there to remove the conditional structure inserted by `\__hook_try_declaring_generic_hook:wnTF` and thus allow access to the tokens that follow it, as is needed to keep things going.

When the deprecation cycle ends, the lines below should all be replaced by `\prg_return_false:`.

```
        \__hook_if_deprecated_generic:nTF {#5}
          {
            \__hook_deprecated_generic_warn:n {#5}
            \exp_after:wN \__hook_declare_deprecated_generic:NNn
            \exp:w % \exp_end:
          }
          { \prg_return_false: }
      }
  }
```

`\__hook_deprecated_generic_warn:n` will issue a deprecation warning for a given hook, and mark that hook such that the warning will not be issued again (multiple warnings can be issued, but only once per hook).

__hook_deprecated_generic_warn:Nn
__hook_deprecated_generic_warn:Nw

```
\cs_new_protected:Npn \__hook_deprecated_generic_warn:n #1
  { \__hook_deprecated_generic_warn:w #1 \s__hook_mark }
\cs_new_protected:Npn \__hook_deprecated_generic_warn:w
    #1 / #2 / #3 \s__hook_mark
  {
    \if_cs_exist:w __hook~#1/#2/#3 \cs_end: \else:
      \msg_warning:nnnnn { hooks } { generic-deprecated } {#1} {#2} {#3}
    \fi:
    \cs_gset_eq:cN { __hook~#1/#2/#3 } \scan_stop:
  }
```

Now that the user has been told about the deprecation, we proceed by swapping ⟨*name*⟩ and ⟨*place*⟩ and adding the code to the correct hook.

__hook_do_deprecated_generic:Nn
__hook_do_deprecated_generic:Nw
__hook_declare_deprecated_generic:NNw
__hook_declare_deprecated_generic:NNw

```
\cs_new_protected:Npn \__hook_do_deprecated_generic:Nn #1 #2
  { \__hook_do_deprecated_generic:Nw #1 #2 \s__hook_mark }
\cs_new_protected:Npn \__hook_do_deprecated_generic:Nw #1
         #2 / #3 / #4 \s__hook_mark
  { #1 { #2 / #4 / #3 } }
\cs_new_protected:Npn \__hook_declare_deprecated_generic:NNn #1 #2 #3
  { \__hook_declare_deprecated_generic:NNw #1 #2 #3 \s__hook_mark }
\cs_new_protected:Npn \__hook_declare_deprecated_generic:NNw #1 #2
    #3 / #4 / #5 \s__hook_mark
  {
    \__hook_try_declaring_generic_hook:wnTF #3 / #5 / #4 / \scan_stop:
        { #3 / #5 / #4 }
      #1 #2 { #3 / #5 / #4 }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}
⟨latexrelease⟩                  {\__hook_try_declaring_generic_hook:wn}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\prg_new_protected_conditional:Npnn
```

```
⟨latexrelease⟩    \__hook_try_declaring_generic_hook:wn
⟨latexrelease⟩    #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_generic:nTF {#5}
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_if_usable:nF {#5}
⟨latexrelease⟩          {
⟨latexrelease⟩            \str_if_eq:nnT {#1} { cmd }
⟨latexrelease⟩              { \__hook_try_put_cmd_hook:n {#5} }
⟨latexrelease⟩            \__hook_make_usable:n {#5}
⟨latexrelease⟩          }
⟨latexrelease⟩        \__hook_if_generic_reversed:nT {#5}
⟨latexrelease⟩          { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
⟨latexrelease⟩        \prg_return_true:
⟨latexrelease⟩      }
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_if_deprecated_generic:nTF {#5}
⟨latexrelease⟩          {
⟨latexrelease⟩            \__hook_deprecated_generic_warn:n {#5}
⟨latexrelease⟩            \exp_after:wN \__hook_declare_deprecated_generic:NNn
⟨latexrelease⟩            \exp:w % \exp_end:
⟨latexrelease⟩          }
⟨latexrelease⟩          { \prg_return_false: }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2021/06/01}
⟨latexrelease⟩                  {\__hook_try_declaring_generic_hook:wn}
⟨latexrelease⟩                  {Support~cmd~hooks}
⟨latexrelease⟩\prg_new_protected_conditional:Npnn
⟨latexrelease⟩    \__hook_try_declaring_generic_hook:wn
⟨latexrelease⟩    #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_if_empty:nTF {#2}
⟨latexrelease⟩      { \prg_return_false: }
⟨latexrelease⟩      {
⟨latexrelease⟩        \prop_if_in:NnTF \c__hook_generics_prop {#1}
⟨latexrelease⟩          {
⟨latexrelease⟩            \__hook_if_usable:nF {#5}
⟨latexrelease⟩              {
⟨latexrelease⟩                \str_if_eq:nnT {#1} { cmd }
⟨latexrelease⟩                  { \__hook_try_put_cmd_hook:n {#5} }
⟨latexrelease⟩                \__hook_make_usable:n {#5}
⟨latexrelease⟩              }
⟨latexrelease⟩            \prop_if_in:NnTF
⟨latexrelease⟩              \c__hook_generics_reversed_ii_prop {#2}
⟨latexrelease⟩              { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
⟨latexrelease⟩              {
⟨latexrelease⟩                \prop_if_in:NnT
⟨latexrelease⟩                  \c__hook_generics_reversed_iii_prop {#3}
⟨latexrelease⟩                  { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
⟨latexrelease⟩              }
⟨latexrelease⟩            \prg_return_true:
⟨latexrelease⟩          }
```

```
⟨latexrelease⟩          { \prg_return_false: }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}
⟨latexrelease⟩                 {\__hook_try_declaring_generic_hook:wn}
⟨latexrelease⟩                 {Support~cmd~hooks}
⟨latexrelease⟩\prg_new_protected_conditional:Npnn
⟨latexrelease⟩    \__hook_try_declaring_generic_hook:wn
⟨latexrelease⟩    #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_if_empty:nTF {#2}
⟨latexrelease⟩      { \prg_return_false: }
⟨latexrelease⟩      {
⟨latexrelease⟩        \prop_if_in:NnTF \c__hook_generics_prop {#1}
⟨latexrelease⟩          {
⟨latexrelease⟩            \__hook_if_declared:nF {#5} { \hook_new:n {#5} }
⟨latexrelease⟩            \prop_if_in:NnTF
⟨latexrelease⟩              \c__hook_generics_reversed_ii_prop {#2}
⟨latexrelease⟩              { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
⟨latexrelease⟩              {
⟨latexrelease⟩                \prop_if_in:NnT
⟨latexrelease⟩                  \c__hook_generics_reversed_iii_prop {#3}
⟨latexrelease⟩                  { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
⟨latexrelease⟩              }
⟨latexrelease⟩            \prg_return_true:
⟨latexrelease⟩          }
⟨latexrelease⟩          { \prg_return_false: }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_try_declaring_generic_hook:wnTF` *and others.*)

`\__hook_if_file_hook_p:w`
`\__hook_if_file_hook:wTF`

`\__hook_if_file_hook:wTF` checks if the argument is a valid file-specific hook (not, for example, `file/before`, but `file/foo.tex/before`). If it is a file-specific hook, then it executes the ⟨*true*⟩ branch, otherwise ⟨*false*⟩.

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__hook_if_file_hook:w}
⟨latexrelease⟩                 {Standardise~generic~hook~names}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_if_file_hook:w}
⟨latexrelease⟩                 {Standardise~generic~hook~names}
⟨latexrelease⟩\prg_new_conditional:Npnn \__hook_if_file_hook:w
⟨latexrelease⟩    #1 / #2 / #3 \s__hook_mark { TF }
⟨latexrelease⟩  {
⟨latexrelease⟩    \str_if_eq:nnTF {#1} { file }
⟨latexrelease⟩      {
⟨latexrelease⟩        \bool_lazy_or:nnTF
⟨latexrelease⟩            { \tl_if_empty_p:n {#3} }
⟨latexrelease⟩            { \str_if_eq_p:nn {#3} { / } }
⟨latexrelease⟩          { \prg_return_false: }
⟨latexrelease⟩          {
⟨latexrelease⟩            \prop_if_in:NnTF \c__hook_generics_file_prop {#2}
⟨latexrelease⟩              { \prg_return_true: }
```

```
⟨latexrelease⟩                { \prg_return_false: }
⟨latexrelease⟩            }
⟨latexrelease⟩        }
⟨latexrelease⟩        { \prg_return_false: }
⟨latexrelease⟩   }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_if_file_hook:wTF`.)

`\__hook_file_hook_normalize:n`
`\__hook_strip_double_slash:n`
`\__hook_strip_double_slash:w`

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__hook_file_hook_normalize:n}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\EndIncludeInRelease
```

When a file-specific hook is found, before being declared it is lightly normalized by `\__hook_file_hook_normalize:n`. The current implementation just replaces two consecutive slashes (`//`) by a single one, to cope with simple cases where the user did something like `\def\input@path{{./mypath/}}`, in which case a hook would have to be `\AddToHook{file/./mypath//file.tex/after}`.

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_file_hook_normalize:n}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\cs_new:Npn \__hook_file_hook_normalize:n #1
⟨latexrelease⟩   { \__hook_strip_double_slash:n {#1} }
⟨latexrelease⟩\cs_new:Npn \__hook_strip_double_slash:n #1
⟨latexrelease⟩   { \__hook_strip_double_slash:w #1 // \s__hook_mark }
```

This function is always called after testing if the argument is a file hook with `\__hook_if_file_hook:wTF`, so we can assume it has three parts (it is either `file/.../before` or `file/.../after`), so we use `#1/#2/#3 //` instead of just `#1 //` to prevent losing a slash if the file name is empty.

```
⟨latexrelease⟩\cs_new:Npn \__hook_strip_double_slash:w #1/#2/#3//#4\s__hook_mark
⟨latexrelease⟩   {
⟨latexrelease⟩     \tl_if_empty:nTF {#4}
⟨latexrelease⟩       { #1/#2/#3 }
⟨latexrelease⟩       { \__hook_strip_double_slash:w #1/#2/#3 /#4\s__hook_mark }
⟨latexrelease⟩   }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_file_hook_normalize:n`, `\__hook_strip_double_slash:n`, *and* `\__hook_strip_double_slash:w`.)

`\c__hook_generic_cmd/./before_tl`
`\c__hook_generic_cmd/./after_tl`
`\c__hook_generic_env/./before_tl`
`\c__hook_generic_env/./after_tl`
`\c__hook_generic_file/./before_tl`
`\c__hook_generic_file/./after_tl`
`\c__hook_generic_package/./before_tl`
`\c__hook_generic_package/./after_tl`
`\c__hook_generic_class/./before_tl`
`\c__hook_generic_class/./after_tl`
`\c__hook_generic_include/./before_tl`
`\c__hook_generic_include/./after_tl`
`\c__hook_generic_env/./begin_tl`
`\c__hook_generic_env/./end_tl`
`\c__hook_generic_include/./end_tl`

Token lists defining the possible generic hooks. We don't provide any user interface to this as this is meant to be static.

`cmd` The generic hooks used for commands.

`env` The generic hooks used in `\begin` and `\end`.

`file, package, class, include` The generic hooks used when loading a file

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\c__hook_generics_prop}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
\clist_map_inline:nn { cmd , env , file , package , class , include }
  {
    \tl_const:cn { c__hook_generic_#1/./before_tl } { + }
    \tl_const:cn { c__hook_generic_#1/./after_tl  } { - }
```

```
  }
\tl_const:cn { c__hook_generic_env/./begin_tl } { + }
\tl_const:cn { c__hook_generic_env/./end_tl   } { + }

\tl_const:cn { c__hook_generic_include/./end_tl } { - }
\tl_const:cn { c__hook_generic_include/./excluded_tl } { + }
```

Deprecated generic hooks:

```
\clist_map_inline:nn { file , package , class , include }
  {
    \tl_const:cn { c__hook_deprecated_#1/./before_tl } { }
    \tl_const:cn { c__hook_deprecated_#1/./after_tl  } { }
  }
\tl_const:cn { c__hook_deprecated_include/./end_tl } { }
⟨latexrelease⟩\EndIncludeInRelease

⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\c__hook_generics_prop}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\prop_const_from_keyval:Nn \c__hook_generics_prop
⟨latexrelease⟩      {cmd=,env=,file=,package=,class=,include=}
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\c__hook_generic_cmd/./before_tl` *and others.*)

`\c__hook_generics_reversed_ii_prop`
`\c__hook_generics_reversed_iii_prop`
`\c__hook_generics_file_prop`

The following generic hooks are supposed to use reverse ordering (the `ii` and `iii` names are kept for the deprecation cycle):

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\c__hook_generics_reversed_ii_prop}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\EndIncludeInRelease

⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\c__hook_generics_reversed_ii_prop}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\prop_const_from_keyval:Nn
⟨latexrelease⟩     \c__hook_generics_reversed_ii_prop {after=,end=}
⟨latexrelease⟩\prop_const_from_keyval:Nn
⟨latexrelease⟩     \c__hook_generics_reversed_iii_prop {after=}
⟨latexrelease⟩\prop_const_from_keyval:Nn
⟨latexrelease⟩     \c__hook_generics_file_prop {before=,after=}
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\c__hook_generics_reversed_ii_prop`, `\c__hook_generics_reversed_iii_prop`, *and* `\c__hook_generics_file_prop`.)

`\c__hook_parameter_cmd/./before_tl`
`\c__hook_parameter_cmd/./after_tl`

Token lists defining the number of arguments for a given type of generic hook.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\c__hook_parameter_cmd/./before_tl}
⟨latexrelease⟩                  {Hooks~with~args}
```

`cmd` hooks are declared with 9 arguments because they have a variable number of arguments (depending on the command they are attached to), so we use the maximum here.

```
\tl_const:cn { c__hook_parameter_cmd/./before_tl } { #1#2#3#4#5#6#7#8#9 }
\tl_const:cn { c__hook_parameter_cmd/./after_tl }  { #1#2#3#4#5#6#7#8#9 }

⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\c__hook_parameter_cmd/./before_tl}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\c__hook_parameter_cmd/./before_tl` *and* `\c__hook_parameter_cmd/./after_tl`.)

`\hook_gremove_code:nn`
`\__hook_gremove_code:nn`

With `\hook_gremove_code:nn{⟨hook⟩}{⟨label⟩}` any code for ⟨*hook*⟩ stored under ⟨*label*⟩ is removed.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_gremove_code:nn}
⟨latexrelease⟩                    {Hooks~with~args}
\cs_new_protected:Npn \hook_gremove_code:nn #1 #2
  { \__hook_normalize_hook_args:Nnn \__hook_gremove_code:nn {#1} {#2} }
\cs_new_protected:Npn \__hook_gremove_code:nn #1 #2
  {
```

First check that the hook code pool exists. `\__hook_if_usable:nTF` isn't used here because it should be possible to remove code from a hook before its defined (see section 2.1.8).

```
    \__hook_if_structure_exist:nTF {#1}
      {
```

Then remove the chunk and run `\__hook_update_hook_code:n` so that the execution token list reflects the change if we are after `\begin{document}`.

If all code is to be removed, clear the code pool `\g__hook_`⟨*hook*⟩`_code_prop`, the top-level code `\__hook_toplevel␣`⟨*hook*⟩, and the next-execution code `\__hook_next␣`⟨*hook*⟩.

```
        \str_if_eq:nnTF {#2} {*}
          {
            \prop_gclear:c { g__hook_#1_code_prop }
            \__hook_toplevel_gset:nn {#1} { }
            \__hook_next_gset:nn {#1} { }
          }
          {
```

If the label is `top-level` then clear the token list, as all code there is under the same label.

```
            \str_if_eq:nnTF {#2} { top-level }
              { \__hook_toplevel_gset:nn {#1} { } }
              {
                \prop_gpop:cnNF { g__hook_#1_code_prop }
                  {#2} \l__hook_return_tl
                  { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
              }
          }
```

Finally update the code, if the hook exists.

```
        \__hook_if_usable:nT {#1}
          { \__hook_update_hook_code:n {#1} }
      }
```

If the code pool for this hook doesn't exist, show a warning:

```
      {
        \__hook_if_deprecated_generic:nTF {#1}
          {
            \__hook_deprecated_generic_warn:n {#1}
            \__hook_do_deprecated_generic:Nn
                \__hook_gremove_code:nn {#1} {#2}
          }
```

```
          { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
      }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_gremove_code:nn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_new_protected:Npn \__hook_gremove_code:nn #1 #2
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_structure_exist:nTF {#1}
⟨latexrelease⟩      {
⟨latexrelease⟩        \str_if_eq:nnTF {#2} {*}
⟨latexrelease⟩          {
⟨latexrelease⟩            \prop_gclear:c { g__hook_#1_code_prop }
⟨latexrelease⟩            \__hook_tl_gclear:c { __hook_toplevel~#1 }
⟨latexrelease⟩            \__hook_tl_gclear:c { __hook_next~#1 }
⟨latexrelease⟩          }
⟨latexrelease⟩          {
⟨latexrelease⟩            \str_if_eq:nnTF {#2} { top-level }
⟨latexrelease⟩              { \__hook_tl_gclear:c { __hook_toplevel~#1 } }
⟨latexrelease⟩              {
⟨latexrelease⟩                \prop_gpop:cnNF { g__hook_#1_code_prop }
⟨latexrelease⟩                  {#2} \l__hook_return_tl
⟨latexrelease⟩                  { \msg_warning:nnnn { hooks } { cannot-remove }
⟨latexrelease⟩                                      {#1} {#2} }
⟨latexrelease⟩              }
⟨latexrelease⟩          }
⟨latexrelease⟩        \__hook_if_usable:nT {#1}
⟨latexrelease⟩          { \__hook_update_hook_code:n {#1} }
⟨latexrelease⟩      }
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_if_deprecated_generic:nTF {#1}
⟨latexrelease⟩          {
⟨latexrelease⟩            \__hook_deprecated_generic_warn:n {#1}
⟨latexrelease⟩            \__hook_do_deprecated_generic:Nn
⟨latexrelease⟩                \__hook_gremove_code:nn {#1} {#2}
⟨latexrelease⟩          }
⟨latexrelease⟩          { \msg_warning:nnnn { hooks } { cannot-remove }
⟨latexrelease⟩                              {#1} {#2} }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_gremove_code:nn` *and* `\__hook_gremove_code:nn`*. This function is documented on page 208.*)

`\__hook_cs_gput_right:nnn`
`\__hook_cs_gput_right_fast:nnn`
`\__hook_cs_gput_right_slow:nnn`
`\__hook_code_gset_auxi:nnnn`
`\__hook_code_gset_auxi:eeen`

This macro is used to append code to the `toplevel` and `next` token lists, trating them correctly depending on their number of arguments, and depending if the code being added should have parameter tokens understood as parameters, or doubled to be stored as parameter tokens.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_cs_gput_right:nnn}
⟨latexrelease⟩                  {Hooks~with~args}
```

Check if the current hook is declared and takes no arguments. In this case, we short-circuit and use the simpler and much faster approach that doesn't require hash-doubling.

```
\cs_new_protected:Npn \__hook_cs_gput_right:nnn #1 #2
  {
    \if:w T
        \__hook_if_declared:nF {#2} { F }
        \tl_if_empty:cF { c__hook_#2_parameter_tl } { F }
          T
      \exp_after:wN \__hook_cs_gput_right_fast:nnn
    \else:
      \exp_after:wN \__hook_cs_gput_right_slow:nnn
    \fi:
        {#1} {#2}
  }
\cs_new_protected:Npn \__hook_cs_gput_right_fast:nnn #1 #2 #3
  { \cs_gset:cpx { __hook#1~#2 }
      { \exp_not:v { __hook#1~#2 } \exp_not:n {#3} } }
\cs_new_protected:Npn \__hook_cs_gput_right_slow:nnn #1 #2 #3
  {
```

The auxiliary `\__hook_code_gset_auxi:eeen` just does the assignment at the end. Its first argument is the parameter text of the macro, which is chosen here depending if `\c___hook_`⟨*hook*⟩`_parameter_tl` exists, if the hook is declared, and if it's a generic hook.

```
    \cs_if_exist:cF { __hook#1~#2 }
      { \__hook_code_gset_aux:nnn {#1} {#2} { } }
    \__hook_code_gset_auxi:eeen
      {
        \__hook_if_declared:nTF {#2}
          { \tl_use:c { c__hook_#2_parameter_tl } }
          {
            \__hook_if_generic:nTF {#2}
              { \__hook_generic_parameter:n {#2} }
              { \c__hook_nine_parameters_tl }
          }
      }
```

Here we take the existing code in the macro, expand it with as many arguments as it takes, then double the hashes so the code can be reused.

PhO: Maybe can be improved. The case of adding to an empty cs can be optimised by quickly checking `\cs_replacement_spec`.

```
      {
        \exp_args:NNo \exp_args:No \__hook_double_hashes:n
          {
            \cs:w __hook#1~#2 \exp_last_unbraced:Ne \cs_end:
              { \__hook_braced_cs_parameter:n { __hook#1~#2 } }
          }
      }
```

Now the new code: if we are replacing arguments, then hashes are left untouched, otherwise they are doubled.

```
      {
        \__hook_if_replacing_args:TF
          { \exp_not:n }
          { \__hook_double_hashes:n }
            {#3}
      }
```

And finally, the csname which we'll define with all the above.

```
      { __hook#1~#2 }
  }
```

And as promised, the auxiliary that does the definition.

```
\cs_new_protected:Npn \__hook_code_gset_auxi:nnnn #1 #2 #3 #4
  { \cs_gset:cpn {#4} #1 { #2 #3 } }
\cs_generate_variant:Nn \__hook_code_gset_auxi:nnnn { eeen }
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_cs_gput_right:nnn}
⟨latexrelease⟩                 {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_cs_gput_right:nnn
⟨latexrelease⟩\cs_undefine:N \__hook_cs_gput_right_fast:nnn
⟨latexrelease⟩\cs_undefine:N \__hook_cs_gput_right_slow:nnn
⟨latexrelease⟩\cs_undefine:N \__hook_code_gset_auxi:nnnn
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_cs_gput_right:nnn` *and others.*)

`\__hook_code_gset:nn`
`\__hook_code_gset:ne`
`\__hook_toplevel_gset:nn`
`\__hook_next_gset:nn`
`\__hook_code_gset_aux:nnn`

These macros define `\__hook`⟨*type*⟩␣⟨*hook*⟩ (with ⟨*type*⟩ being `_next`, `_toplevel`, or empty) with the given code and the parameters stored in `\c__hook_`⟨*hook*⟩`_parameter_tl` (or none, if that doesn't exist).

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_code_gset:nn}
⟨latexrelease⟩                 {Hooks~with~args}
\cs_new_protected:Npn \__hook_code_gset:nn
  { \__hook_code_gset_aux:nnn { } }
\cs_new_protected:Npn \__hook_toplevel_gset:nn
  { \__hook_code_gset_aux:nnn { _toplevel } }
\cs_new_protected:Npn \__hook_next_gset:nn
  { \__hook_code_gset_aux:nnn { _next } }
\cs_new_protected:Npn \__hook_code_gset_aux:nnn #1 #2 #3
  {
    \cs_gset:cpn { __hook#1~#2 \exp_last_unbraced:Ne }
      { \__hook_parameter:n {#2} }
      {#3}
  }
\cs_generate_variant:Nn \__hook_code_gset:nn { ne }
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_code_gset:nn}
⟨latexrelease⟩                 {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_code_gset:nn
⟨latexrelease⟩\cs_undefine:N \__hook_toplevel_gset:nn
⟨latexrelease⟩\cs_undefine:N \__hook_next_gset:nn
⟨latexrelease⟩\cs_undefine:N \__hook_code_gset_aux:nnn
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_code_gset:nn` *and others.*)

`\__hook_normalise_cs_args:nn`

This macro normalises the parameters of the macros `\__hook`⟨*type*⟩␣⟨*hook*⟩ to take the right number of arguments after a hook is declared. At this point we know `\c__hook_`⟨*hook*⟩`_parameter_tl` exists, so use that to count the arguments and use that as ⟨*parameter text*⟩ for the newly (re)defined macro.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_normalise_cs_args:nn}
⟨latexrelease⟩                 {Hooks~with~args}
\cs_new_protected:Npn \__hook_normalise_cs_args:nn #1 #2
  {
    \cs_if_exist:cT { __hook#1~#2 }
```

```
      {
        \__hook_code_gset_auxi:eeen
          { \tl_use:c { c__hook_#2_parameter_tl } }
          {
            \exp_args:NNo \exp_args:No \__hook_double_hashes:n
              {
                \cs:w __hook#1~#2 \exp_last_unbraced:Ne \cs_end:
                  { \__hook_braced_cs_parameter:n { __hook#1~#2 } }
              }
          }
          { }
          { __hook#1~#2 }
      }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_normalise_cs_args:nn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_normalise_cs_args:nn
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_normalise_cs_args:nn`.)

`\__hook_normalise_code_pool:n`
`\__hook_set_normalise_fn:nn`

This one's a bit of a hack. It takes a hook, and iterates over its code pool (`\g__hook_-⟨hook⟩_code_prop`), redefining each code label to use only valid arguments. This is used when, for example, a code is added referencing arguments `#1` and `#2`, but the hook has only `#1`. In this example, every reference to `#2` is changed to `##2`. This is done because otherwise TEX will throw a low-level error every time some change happens to the hook (code is added, a rule is set, etc), which can get quite repetitive for no good reason.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_normalise_code_pool:n}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_normalise_code_pool:n #1
  {
```

First, call `\__hook_set_normalise_fn:nn` with the hook name to set everything up, then we'll loop over the hook's code pool applying the normalisation above. After that's done, copy the temporary property list back to the hook's.

```
    \__hook_set_normalise_fn:nn {#1} { Offending~label:~'##1' }
    \prop_clear:N \l__hook_work_prop
    \prop_map_function:cN { g__hook_#1_code_prop } \__hook_normalise_fn:nn
    \prop_gset_eq:cN { g__hook_#1_code_prop } \l__hook_work_prop
  }
```

The sole purpose of this function is to define `\__hook_normalise_fn:nn`, which will then do the correcting of the code being added to the hook.

```
\cs_new_protected:Npn \__hook_set_normalise_fn:nn #1 #2
  {
```

To start, we define two auxiliary token lists. `\l__hook_tmpb_tl` contains:

```
{\c__hook_hashes_tl 1}
{\c__hook_hashes_tl 2}
...
{\c__hook_hashes_tl 9}
```

```
    \cs_set:Npn \__hook_tmp:w ##1##2##3##4##5##6##7##8##9 { }
    \tl_set:Ne \l__hook_tmpb_tl
      { \__hook_braced_cs_parameter:n { __hook_tmp:w } }
    \group_begin:
      \__hook_tl_set:cn { c__hook_hash_tl } { \exp_not:N \c__hook_hashes_tl }
      \use:e
        {
    \group_end:
    \tl_set:Nn \exp_not:N \l__hook_tmpb_tl { \l__hook_tmpb_tl }
        }
```

And `\l__hook_tmpa_tl` contains:

```
{\c__hook_hash_tl 1}
{\c__hook_hash_tl 2}
...
{\c__hook_hash_tl <n>}
```

with $\langle n \rangle$ being the number of arguments declared for the hook.

```
    \exp_last_unbraced:NNf
    \cs_set:Npn \__hook_tmp:w { \__hook_parameter:n {#1} } { }
    \tl_set:Ne \l__hook_tmpa_tl
      { \__hook_braced_cs_parameter:n { __hook_tmp:w } }
```

Now this function does the fun part. It is meant to be used with `\prop_map_function:NN`, taking a label name in `##1` and the code stored in that label in `##2`.

```
    \cs_gset_protected:Npx \__hook_normalise_fn:nn ##1 ##2
      {
```

Here we'll define two auxiliary macros: the first one throws an error when it detects an invalid argument reference. It piggybacks on TEX's low-level "Illegal parameter number" error, but it defines a weirdly-named control sequence so that the error comes out nicely formatted. For example, if the label "badpkg" adds some code that references argument `#3` in the hook "foo", which takes only two arguments, the error will be:

```
! Illegal parameter number in definition of hook 'foo'.
(hooks)             Offending label: 'badpkg'.
<to be read again>
                    3
```

At the point of this definition, the error is raised if the code happens to reference an invalid argument. If it was possible to detect that this definition raised no error, the next step would be unnecessary. We'll do all this in a group so this weird definition doesn't leak out, and set `\tex_escapechar:D` to -1 so this hack shows up extra nice in the case of an error.

```
        \group_begin:
          \int_set:Nn \tex_escapechar:D { -1 }
          \cs_set:cpn
              {
                hook~'#1'. ^^J
                (hooks) \prg_replicate:nn { 13 } { ~ }
                #2 % more message text
              }
              \exp_not:v { c__hook_#1_parameter_tl }
            {##2}
        \group_end:
```

This next macro, with a much less fabulous name, takes always nine arguments, and it just transfers the code ##2 under the label ##1 to the temporary property list. The first ⟨*n*⟩ arguments are taken from \l__hook_tmpa_tl, and the other 9−⟨*n*⟩ taken from \l__hook_tmpb_tl (which contains twice as many # tokens as the former). Then, __hook_double_hashes:n is used to double non-argument hashes, and expand the \c__hook_hash_tl and \c__hook_hashes_tl to the actual parameter tokens.

```
        \cs_set:Npn \exp_not:N \__hook_tmp:w
            \exp_not:V \c__hook_nine_parameters_tl
          {
            \prop_put:Nne \exp_not:N \l__hook_work_prop
              {##1} { \exp_not:N \__hook_double_hashes:n {##2} }
          }
```

This next macro, with a much less fabulous name, takes always nine arguments, and it just transfers the code ##2 under the label ##1 to the temporary property list. The first ⟨*n*⟩ arguments are taken from \l__hook_tmpa_tl, and the other 9−⟨*n*⟩ taken from \l__hook_tmpb_tl (which contains twice as many # tokens as the former). Then, __hook_double_hashes:n is used to double non-argument hashes, and expand the \c__hook_hash_tl and \c__hook_hashes_tl to the actual parameter tokens.

```
        \exp_not:N \__hook_tmp:w
          \exp_not:V \l__hook_tmpa_tl
          \exp_args:No \exp_not:o
            { \exp_after:wN \__hook_tmp:w \l__hook_tmpb_tl }
      }
  }
\cs_new_eq:NN \__hook_normalise_fn:nn ?
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_normalise_code_pool:n}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_normalise_code_pool:n
⟨latexrelease⟩\EndIncludeInRelease
```

Check if the expansion of a control sequence is empty by looking at its replacement text.

__hook_cs_if_empty_p:c
__hook_cs_if_empty:c*TF*

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_cs_if_empty:c}
⟨latexrelease⟩                  {Hooks~with~args}
\prg_new_conditional:Npnn \__hook_cs_if_empty:c #1 { p, T, F, TF }
  {
    \if:w \scan_stop: \__hook_replacement_spec:c {#1} \scan_stop:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
\cs_new:Npn \__hook_replacement_spec:c #1
  {
    \exp_args:Nc \token_if_macro:NT {#1}
      { \cs_replacement_spec:c {#1} }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_cs_if_empty:c}
⟨latexrelease⟩                  {Hooks~with~args}
```

```
⟨latexrelease⟩\cs_undefine:N \__hook_cs_if_empty:c
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_normalise_code_pool:n`, `\__hook_set_normalise_fn:nn`, *and* `\__hook_cs_if_empty:cTF`.)

`\__hook_braced_cs_parameter:n`
`\__hook_braced_hidden_loop:w`
`\__hook_cs_parameter_count:N`
`\__hook_cs_parameter_count:w`
`\__hook_cs_end:w`

Looks at the ⟨*parameter text*⟩ of a control sequence, and returns a run of "hidden" braced parameters for that macro. This works as long as the macros take a simple run of zero to nine arguments. The parameters are "hidden" because the parameter tokens are returned inside `\c__hook_hash_tl` instead of explicitly, so that `\__hook_double_hashes:n` won't touch these.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_braced_cs_parameter:n}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new:Npn \__hook_braced_cs_parameter:n #1
  {
    \exp_last_unbraced:Ne \__hook_braced_hidden_loop:w
      { \exp_args:Nc \__hook_cs_parameter_count:N {#1} } ? \s__hook_mark
  }
\cs_new:Npn \__hook_braced_hidden_loop:w #1
  {
    \if:w ? #1
      \__hook_use_i_delimit_by_s_mark:nw
    \fi:
    { \exp_not:N \c__hook_hash_tl #1 }
    \__hook_braced_hidden_loop:w
  }
\cs_new:Npn \__hook_cs_parameter_count:N #1
  {
    \exp_last_unbraced:Nf \__hook_cs_parameter_count:w
      { \token_if_macro:NT #1 { \cs_parameter_spec:N #1 } }
      ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
      ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
      ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
      \s__hook_mark
  }
\cs_new:Npn \__hook_cs_parameter_count:w #1#2 #3#4 #5#6 #7#8
  { #2 #4 #6 #8 \__hook_cs_parameter_count:w }
\cs_new:Npn \__hook_cs_end:w #1 \s__hook_mark { }
⟨latexrelease⟩\EndIncludeInRelease
```

This function can't be undefined when rolling back because it's used at the end of this module to adequate the hook data structures to previous versions.

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_braced_cs_parameter:n}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_braced_cs_parameter:n` *and others.*)

`\__hook_braced_parameter:n`
`\__hook_braced_real_loop:w`

This one is used in simpler cases, where no special handling of hashes is required. This is used only inside `\__hook_initialize_hook_code:n`, so it assumes `\c__hook_`⟨*hook*⟩`_parameter_tl` is defined, but should work otherwise.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_braced_parameter:n}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new:Npn \__hook_braced_parameter:n #1
```

```
  {
    \if_case:w
      \int_eval:n
        { \exp_args:Nv \str_count:n { c__hook_#1_parameter_tl } / 3 }
      \exp_stop_f:
    \or: {##1}
    \or: {##1} {##2}
    \or: {##1} {##2} {##3}
    \or: {##1} {##2} {##3} {##4}
    \or: {##1} {##2} {##3} {##4} {##5}
    \or: {##1} {##2} {##3} {##4} {##5} {##6}
    \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7}
    \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7} {##8}
    \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7} {##8} {##9}
    \else:
      \msg_expandable_error:nnn { latex2e } { should-not-happen }
        { Invalid~parameter~spec. }
    \fi:
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_braced_parameter:n}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_braced_parameter:n
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_braced_parameter:n` *and* `\__hook_braced_real_loop:w`.)

`\__hook_parameter:n` This is just a shortcut to `e`- or `f`-expand to the ⟨*parameter text*⟩ of the hook.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_parameter:n}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new:Npn \__hook_parameter:n #1
  {
    \cs:w c__hook_
    \tl_if_exist:cTF { c__hook_#1_parameter_tl }
      { #1_parameter } { empty }
    _tl \cs_end:
  }
\cs_new:Npn \__hook_generic_parameter:n #1
  { \__hook_generic_parameter:w #1 / / / \s__hook_mark }
\cs_new:Npn \__hook_generic_parameter:w #1 / #2 / #3 / #4 \s__hook_mark
  {
    \cs_if_exist_use:cF { c__hook_parameter_#1/./#3_tl }
      { \c__hook_empty_tl }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_parameter:n}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_parameter:n
⟨latexrelease⟩\cs_undefine:N \__hook_generic_parameter:n
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_parameter:n`.)

4.7 Setting rules for hooks code

\g__hook_??_code_prop
__hook~??
\g__hook_??_reversed_tl
\c__hook_??_parameter_tl

Initially these variables simply used an empty "label" name (not two question marks). This was a bit unfortunate, because then l3doc complains about __ in the middle of a command name when trying to typeset the documentation. However using a "normal" name such as default has the disadvantage of that being not really distinguishable from a real hook name. I now have settled for ?? which needs some gymnastics to get it into the csname, but since this is used a lot, the code should be fast, so this is not done with c expansion in the code later on.

__hook␣?? isn't used, but it has to be defined to trick the code into thinking that ?? is actually a hook.

```
\prop_new:c { g__hook_??_code_prop }
\prop_new:c { __hook~?? }
```

Default rules are always given in normal ordering (never in reversed ordering). If such a rule is applied to a reversed hook it behaves as if the rule is reversed (e.g., after becomes before) because those rules are applied first and then the order is reversed.

```
\tl_new:c { g__hook_??_reversed_tl }
```

The parameter text for the "default" hook is empty.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\c__hook_??_parameter_tl}
⟨latexrelease⟩                  {Hooks~with~args}
\tl_const:cn { c__hook_??_parameter_tl } { }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\c__hook_??_parameter_tl}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:c { c__hook_??_parameter_tl }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* \g__hook_??_code_prop *and others.*)

\hook_gset_rule:nnnn
__hook_gset_rule:nnnn

With \hook_gset_rule:nnnn{⟨*hook*⟩}{⟨*label1*⟩}{⟨*relation*⟩}{⟨*label2*⟩} a relation is defined between the two code labels for the given ⟨*hook*⟩. The special hook ?? stands for *any* hook, which sets a default rule (to be used if no other relation between the two hooks exist).

```
\cs_new_protected:Npn \hook_gset_rule:nnnn #1#2#3#4
  {
    \__hook_normalize_hook_rule_args:Nnnnn \__hook_gset_rule:nnnn
      {#1} {#2} {#3} {#4}
  }
```

```
⟨latexrelease⟩\IncludeInRelease{2022/06/01}{\__hook_gset_rule:nnnn}
⟨latexrelease⟩                  {Refuse~setting~rule~for~one-time~hooks}
```

```
\cs_new_protected:Npn \__hook_gset_rule:nnnn #1#2#3#4
  {
    \__hook_if_deprecated_generic:nT {#1}
      {
        \__hook_deprecated_generic_warn:n {#1}
        \__hook_do_deprecated_generic:Nn \__hook_gset_rule:nnnn {#1}
          {#2} {#3} {#4}
        \__hook_use_none_delimit_by_s_mark:w
      }
    \__hook_if_execute_immediately:nT {#1}
      {
```

```
        \msg_error:nnnnnn { hooks } { rule-too-late }
          {#1} {#2} {#3} {#4}
        \__hook_use_none_delimit_by_s_mark:w
      }
```

First we ensure the basic data structure of the hook exists:

```
    \__hook_init_structure:n {#1}
```

Then we clear any previous relationship between both labels.

```
    \__hook_rule_gclear:nnn {#1} {#2} {#4}
```

Then we call the function to handle the given rule. Throw an error if the rule is invalid.

```
    \cs_if_exist_use:cTF { __hook_rule_#3_gset:nnn }
      {
          {#1} {#2} {#4}
        \__hook_update_hook_code:n {#1}
      }
      {
        \msg_error:nnnnnn { hooks } { unknown-rule }
          {#1} {#2} {#3} {#4}
      }
    \s__hook_mark
  }
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_gset_rule:nnnn}
⟨latexrelease⟩                  {Refuse~setting~rule~for~one-time~hooks}
⟨latexrelease⟩\cs_new_protected:Npn \__hook_gset_rule:nnnn #1#2#3#4
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_deprecated_generic:nT {#1}
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_deprecated_generic_warn:n {#1}
⟨latexrelease⟩        \__hook_do_deprecated_generic:Nn \__hook_gset_rule:nnnn
⟨latexrelease⟩          {#1} {#2} {#3} {#4}
⟨latexrelease⟩        \exp_after:wN \use_none:nnnnnnnnn \use_none:n
⟨latexrelease⟩      }
⟨latexrelease⟩    \__hook_init_structure:n {#1}
⟨latexrelease⟩    \__hook_rule_gclear:nnn {#1} {#2} {#4}
⟨latexrelease⟩    \cs_if_exist_use:cTF { __hook_rule_#3_gset:nnn }
⟨latexrelease⟩      {
⟨latexrelease⟩          {#1} {#2} {#4}
⟨latexrelease⟩        \__hook_update_hook_code:n {#1}
⟨latexrelease⟩      }
⟨latexrelease⟩      {
⟨latexrelease⟩        \msg_error:nnnnnn { hooks } { unknown-rule }
⟨latexrelease⟩          {#1} {#2} {#3} {#4}
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_gset_rule:nnnn` *and* `\__hook_gset_rule:nnnn`*. This function is documented on page* 209.)

`\__hook_rule_before_gset:nnn`
`\__hook_rule_after_gset:nnn`
`\__hook_rule_<_gset:nnn`
`\__hook_rule_>_gset:nnn`

Then we add the new rule. We need to normalize the rules here to allow for faster processing later. Given a pair of labels l_A and l_B, the rule $l_A > l_B$ is the same as $l_B < l_A$ only

presented differently. But by normalizing the forms of the rule to a single representation, say, $l_B < l_A$, reduces the time spent looking for the rules later considerably.

Here we do that normalization by using `\(pdf)strcmp` to lexically sort labels l_A and l_B to a fixed order. This order is then enforced every time these two labels are used together.

Here we use `\__hook_label_pair:nn {`$\langle hook \rangle$`} {`$\langle l_A \rangle$`} {`$\langle l_B \rangle$`}` to build a string l_B | l_A with a fixed order, and use `\__hook_label_ordered:nnTF` to apply the correct rule to the pair of labels, depending if it was sorted or not.

```
\cs_new_protected:Npn \__hook_rule_before_gset:nnn #1#2#3
  {
    \__hook_tl_gset:cx
      { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
      { \__hook_label_ordered:nnTF {#2} {#3} { < } { > } }
  }
\cs_new_eq:cN { __hook_rule_<_gset:nnn } \__hook_rule_before_gset:nnn
\cs_new_protected:Npn \__hook_rule_after_gset:nnn #1#2#3
  {
    \__hook_tl_gset:cx
      { g__hook_#1_rule_ \__hook_label_pair:nn {#3} {#2} _tl }
      { \__hook_label_ordered:nnTF {#3} {#2} { < } { > } }
  }
\cs_new_eq:cN { __hook_rule_>_gset:nnn } \__hook_rule_after_gset:nnn
```

(*End of definition for* `\__hook_rule_before_gset:nnn` *and others.*)

`\__hook_rule_voids_gset:nnn`

This rule removes (clears, actually) the code from label `#3` if label `#2` is in the hook `#1`.

```
\cs_new_protected:Npn \__hook_rule_voids_gset:nnn #1#2#3
  {
    \__hook_tl_gset:cx
      { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
      { \__hook_label_ordered:nnTF {#2} {#3} { -> } { <- } }
  }
```

(*End of definition for* `\__hook_rule_voids_gset:nnn`.)

`\__hook_rule_incompatible-error_gset:nnn`
`\__hook_rule_incompatible-warning_gset:nnn`

These relations make an error/warning if labels `#2` and `#3` appear together in hook `#1`.

```
\cs_new_protected:cpn { __hook_rule_incompatible-error_gset:nnn } #1#2#3
  { \__hook_tl_gset:cn
      { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
      { xE }
  }
\cs_new_protected:cpn { __hook_rule_incompatible-warning_gset:nnn } #1#2#3
  { \__hook_tl_gset:cn
      { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
      { xW }
  }
```

(*End of definition for* `\__hook_rule_incompatible-error_gset:nnn` *and* `\__hook_rule_incompatible-warning_gset:nnn`.)

`\__hook_rule_unrelated_gset:nnn`
`\__hook_rule_gclear:nnn`

Undo a setting. `\__hook_rule_unrelated_gset:nnn` doesn't need to do anything, since we use `\__hook_rule_gclear:nnn` before setting any rule.

```
\cs_new_protected:Npn \__hook_rule_unrelated_gset:nnn #1#2#3 { }
\cs_new_protected:Npn \__hook_rule_gclear:nnn #1#2#3
  { \cs_undefine:c { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl } }
```

(*End of definition for* `\__hook_rule_unrelated_gset:nnn` *and* `\__hook_rule_gclear:nnn`.)

`\__hook_label_pair:nn` Ensure that the lexically greater label comes first.

```
\cs_new:Npn \__hook_label_pair:nn #1#2
  {
    \if_case:w \__hook_str_compare:nn {#1} {#2} \exp_stop_f:
           #1 | #1 %  0
    \or:   #1 | #2 % +1
    \else: #2 | #1 % -1
    \fi:
  }
```

(*End of definition for* `\__hook_label_pair:nn`.)

`\__hook_label_ordered_p:nn` `\__hook_label_ordered:nnTF` Check that labels `#1` and `#2` are in the correct order (as returned by `\__hook_label_pair:nn`) and if so return true, else return false.

```
\prg_new_conditional:Npnn \__hook_label_ordered:nn #1#2 { TF }
  {
    \if_int_compare:w \__hook_str_compare:nn {#1} {#2} > 0 \exp_stop_f:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
```

(*End of definition for* `\__hook_label_ordered:nnTF`.)

`\__hook_if_label_case:nnnnn` To avoid doing the string comparison twice in `\__hook_initialize_single:NNn` (once with `\str_if_eq:nn` and again with `\__hook_label_ordered:nn`), we use a three-way branching macro that will compare `#1` and `#2` and expand to `\use_i:nnn` if they are equal, `\use_ii:nn` if `#1` is lexically greater, and `\use_iii:nn` otherwise.

```
\cs_new:Npn \__hook_if_label_case:nnnnn #1#2
  {
    \cs:w use_
      \if_case:w \__hook_str_compare:nn {#1} {#2}
         i \or: ii \else: iii \fi: :nnn
    \cs_end:
  }
```

(*End of definition for* `\__hook_if_label_case:nnnnn`.)

`\__hook_update_hook_code:n` Before `\begin{document}` this does nothing, in the body it reinitializes the hook code using the altered data.

```
\cs_new_eq:NN \__hook_update_hook_code:n \use_none:n
```

(*End of definition for* `\__hook_update_hook_code:n`.)

`\__hook_initialize_all:` Initialize all known hooks (at `\begin{document}`), i.e., update the fast execution token lists to hold the necessary code in the right order.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_initialize_all:}
⟨latexrelease⟩                    {Hooks~with~args}
\cs_new_protected:Npn \__hook_initialize_all:
  {
```

First we change `\__hook_update_hook_code:n` which so far was a no-op to now initialize one hook. This way any later updates to the hook will run that code and also update the execution token list.

```
    \cs_gset_eq:NN \__hook_update_hook_code:n \__hook_initialize_hook_code:n
```

Now we loop over all hooks that have been defined and update each of them. Here we have to determine if the hook has arguments so that auxiliaries know what to do with hashes. We look at `\c__hook_`⟨*hook*⟩`_parameter_tl`, if it has any parameters, and set `replacing_args` accordingly.

```
    \__hook_debug:n { \prop_gclear:N \g__hook_used_prop }
    \seq_map_inline:Nn \g__hook_all_seq
      {
        \tl_if_empty:cTF { c__hook_##1_parameter_tl }
          { \__hook_replacing_args_false: }
          { \__hook_replacing_args_true: }
        \__hook_update_hook_code:n {##1}
        \__hook_replacing_args_reset:
      }
```

If we are debugging we show results hook by hook for all hooks that have data.

```
    \__hook_debug:n
      {
        \iow_term:x { ^^J All~initialized~(non-empty)~hooks: }
        \prop_map_inline:Nn \g__hook_used_prop
          {
            \iow_term:x
              { ^^J ~ ##1 ~ -> ~ \cs_replacement_spec:c { __hook~##1 } ~ }
          }
      }
```

After all hooks are initialized we change the "use" to just call the hook code and not initialize it (as this was already done in the preamble.

```
    \__hook_post_initialization_defs:
  }
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_initialize_all:}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_initialize_all:
⟨latexrelease⟩  {
⟨latexrelease⟩    \cs_gset_eq:NN \__hook_update_hook_code:n
⟨latexrelease⟩                   \__hook_initialize_hook_code:n
⟨latexrelease⟩    \__hook_debug:n { \prop_gclear:N \g__hook_used_prop }
⟨latexrelease⟩    \seq_map_inline:Nn \g__hook_all_seq
⟨latexrelease⟩      { \__hook_update_hook_code:n {##1} }
⟨latexrelease⟩    \__hook_debug:n
⟨latexrelease⟩      {
⟨latexrelease⟩        \iow_term:x{^^JAll~ initialized~ (non-empty)~ hooks:}
⟨latexrelease⟩        \prop_map_inline:Nn \g__hook_used_prop
⟨latexrelease⟩          {
⟨latexrelease⟩            \iow_term:x
⟨latexrelease⟩              { ^^J ~ ##1 ~ -> ~
⟨latexrelease⟩                \cs_replacement_spec:c { __hook~##1 } ~ }
⟨latexrelease⟩          }
⟨latexrelease⟩      }
```

```
⟨latexrelease⟩     \cs_gset_eq:NN \hook_use:n \__hook_use_initialized:n
⟨latexrelease⟩     \cs_gset_eq:NN \__hook_preamble_hook:n \use_none:n
⟨latexrelease⟩   }
⟨@@=⟩
⟨latexrelease⟩\cs_gset_eq:NN \@expl@@@initialize@all@@
⟨latexrelease⟩                \__hook_initialize_all:
⟨@@=hook⟩
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_initialize_all:`.)

`\__hook_initialize_hook_code:n` Initializing or reinitializing the fast execution hook code. In the preamble this is selectively done in case a hook gets used and at `\begin{document}` this is done for all hooks and afterwards only if the hook code changes.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_initialize_hook_code:n}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_initialize_hook_code:n #1
  {
    \__hook_debug:n
      { \iow_term:x { ^^J Update~code~for~hook~'#1' \on@line :^^J } }
```

This does the sorting and the updates. First thing we do is to check if a legacy hook macro exists and if so we add it to the hook under the label `legacy`. This might make the hook non-empty so we have to do this before the then following test.

```
    \__hook_include_legacy_code_chunk:n {#1}
```

If there aren't any code chunks for the current hook, there is no point in even starting the sorting routine so we make a quick test for that and in that case just update `\__hook␣`⟨*hook*⟩ to hold the `top-level` and `next` code chunks. If there are code chunks we call `\__hook_initialize_single:NNn` and pass to it ready made csnames as they are needed several times inside. This way we save a bit on processing time if we do that up front.

```
    \__hook_if_usable:nT {#1}
      {
        \prop_if_empty:cTF { g__hook_#1_code_prop }
          {
            \__hook_code_gset:ne {#1}
              {
```

The hook may take arguments, so we add a run of braced parameters after the `_next` and `_toplevel` macros, so that the arguments passed to the hook are forwarded to them.

```
                \exp_not:c { __hook_toplevel~#1 }
                \__hook_braced_parameter:n {#1}
                \exp_not:c { __hook_next~#1 }
                \__hook_braced_parameter:n {#1}
              }
          }
          {
```

By default the algorithm sorts the code chunks and then saves the result in a token list for fast execution; this is done by adding the code chunks one after another, using `\tl_gput_right:NV`. When we sort code for a reversed hook, all we have to do is to add the code chunks in the opposite order into the token list. So all we have to do in preparation is to change two definitions that are used later on.

```
            \__hook_if_reversed:nTF {#1}
```

```
              { \cs_set_eq:NN \__hook_tl_gput:Nn    \__hook_tl_gput_left:Nn
                \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_left:NV  }
              { \cs_set_eq:NN \__hook_tl_gput:Nn    \__hook_tl_gput_right:Nn
                \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_right:NV }
```

When sorting, some relations (namely `voids`) need to act destructively on the code property lists to remove code that shouldn't appear in the sorted hook token list, so we make a copy of the code property list that we can safely work on without changing the main one.

```
            \prop_set_eq:Nc \l__hook_work_prop { g__hook_#1_code_prop }
            \__hook_initialize_single:ccn
              { __hook~#1 } { g__hook_#1_labels_clist } {#1}
```

For debug display we want to keep track of those hooks that actually got code added to them, so we record that in plist. We use a plist to ensure that we record each hook name only once, i.e., we are only interested in storing the keys and the value is arbitrary.

```
            \__hook_debug:n
              { \exp_args:NNx \prop_gput:Nnn \g__hook_used_prop {#1} { } }
          }
      }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_initialize_hook_code:n}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_initialize_hook_code:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_debug:n
⟨latexrelease⟩      { \iow_term:x { ^^J Update~code~for~hook~'#1'
⟨latexrelease⟩                      \on@line :^^J } }
⟨latexrelease⟩    \__hook_include_legacy_code_chunk:n {#1}
⟨latexrelease⟩    \__hook_if_usable:nT {#1}
⟨latexrelease⟩      {
⟨latexrelease⟩        \prop_if_empty:cTF { g__hook_#1_code_prop }
⟨latexrelease⟩          {
⟨latexrelease⟩            \__hook_tl_gset:co { __hook~#1 }
⟨latexrelease⟩              {
⟨latexrelease⟩                \cs:w __hook_toplevel~#1 \exp_after:wN \cs_end:
⟨latexrelease⟩                \cs:w __hook_next~#1 \cs_end:
⟨latexrelease⟩              }
⟨latexrelease⟩          }
⟨latexrelease⟩          {
⟨latexrelease⟩            \__hook_if_reversed:nTF {#1}
⟨latexrelease⟩              { \cs_set_eq:NN \__hook_tl_gput:Nn
⟨latexrelease⟩                              \__hook_tl_gput_left:Nn
⟨latexrelease⟩                \cs_set_eq:NN \__hook_clist_gput:NV
⟨latexrelease⟩                              \clist_gput_left:NV  }
⟨latexrelease⟩              { \cs_set_eq:NN \__hook_tl_gput:Nn
⟨latexrelease⟩                              \__hook_tl_gput_right:Nn
⟨latexrelease⟩                \cs_set_eq:NN \__hook_clist_gput:NV
⟨latexrelease⟩                              \clist_gput_right:NV }
⟨latexrelease⟩            \prop_set_eq:Nc \l__hook_work_prop
⟨latexrelease⟩              { g__hook_#1_code_prop }
⟨latexrelease⟩            \__hook_initialize_single:ccn
⟨latexrelease⟩              { __hook~#1 } { g__hook_#1_labels_clist } {#1}
```

```
⟨latexrelease⟩            \__hook_debug:n
⟨latexrelease⟩              { \exp_args:NNx \prop_gput:Nnn \g__hook_used_prop
⟨latexrelease⟩                    {#1} { } }
⟨latexrelease⟩          }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_initialize_hook_code:n`.)

`\__hook_tl_csname:n`
`\__hook_seq_csname:n`

It is faster to pass a single token and expand it when necessary than to pass a bunch of character tokens around.

FMi: note to myself: verify

```
\cs_new:Npn \__hook_tl_csname:n #1 { l__hook_label_#1_tl }
\cs_new:Npn \__hook_seq_csname:n #1 { l__hook_label_#1_seq }
```

(*End of definition for* `\__hook_tl_csname:n` *and* `\__hook_seq_csname:n`.)

`\l__hook_labels_seq`
`\l__hook_labels_int`
`\l__hook_front_tl`
`\l__hook_rear_tl`
`\l__hook_label_0_tl`

For the sorting I am basically implementing Knuth's algorithm for topological sorting as given in TAOCP volume 1 pages 263–266. For this algorithm we need a number of local variables:

- List of labels used in the current hook to label code chunks:

```
\seq_new:N \l__hook_labels_seq
```

- Number of labels used in the current hook. In Knuth's algorithm this is called N:

```
\int_new:N \l__hook_labels_int
```

- The sorted code list to be build is managed using two pointers one to the front of the queue and one to the rear. We model this using token list pointers. Knuth calls them F and R:

```
\tl_new:N \l__hook_front_tl
\tl_new:N \l__hook_rear_tl
```

- The data for the start of the queue is kept in this token list, it corresponds to what Don calls `QLINK[0]` but since we aren't manipulating individual words in memory it is slightly differently done:

```
\tl_new:c { \__hook_tl_csname:n { 0 } }
```

(*End of definition for* `\l__hook_labels_seq` *and others.*)

`\__hook_initialize_single:NNn`
`\__hook_initialize_single:ccn`

`\__hook_initialize_single:NNn` implements the sorting of the code chunks for a hook and saves the result in the token list for fast execution (`#4`). The arguments are ⟨*hook-code-plist*⟩, ⟨*hook-code-tl*⟩, ⟨*hook-top-level-code-tl*⟩, ⟨*hook-next-code-tl*⟩, ⟨*hook-ordered-labels-clist*⟩ and ⟨*hook-name*⟩ (the latter is only used for debugging—the ⟨*hook-rule-plist*⟩ is accessed using the ⟨*hook-name*⟩).

The additional complexity compared to Don's algorithm is that we do not use simple positive integers but have arbitrary alphanumeric labels. As usual Don's data structures are chosen in a way that one can omit a lot of tests and I have mimicked that as far as possible. The result is a restriction I do not test for at the moment: a label can't be equal to the number 0!

FMi: Needs checking for, just in case … maybe

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_initialize_single:NNn}
⟨latexrelease⟩                    {Hooks~with~args}
\cs_new_protected:Npn \__hook_initialize_single:NNn #1#2#3
  {
```

Step T1: Initialize the data structure …

```
    \seq_clear:N \l__hook_labels_seq
    \int_zero:N  \l__hook_labels_int
```

Store the name of the hook:

```
    \tl_set:Nn \l__hook_cur_hook_tl {#3}
```

We loop over the property list holding the code and record all the labels listed there. Only the rules for those labels are of interest to us. While we are at it we count them (which gives us the N in Knuth's algorithm). The prefix `label_` is added to the variables to ensure that labels named `front`, `rear`, `labels`, or `return` don't interact with our code.

```
    \prop_map_inline:Nn \l__hook_work_prop
       {
         \int_incr:N \l__hook_labels_int
         \seq_put_right:Nn \l__hook_labels_seq {##1}
         \__hook_tl_set:cn { \__hook_tl_csname:n {##1} } { 0 }
         \seq_clear_new:c { \__hook_seq_csname:n {##1} }
       }
```

Steps T2 and T3: Here we sort the relevant rules into the data structure…

This loop constitutes a square matrix of the labels in `\l__hook_work_prop` in the vertical and the horizontal directions. However, since the rule $l_A\langle\texttt{rel}\rangle l_B$ is the same as $l_B\langle\texttt{rel}\rangle^{-1} l_A$ we can cut the loop short at the diagonal of the matrix (*i.e.*, when both labels are equal), saving a good amount of time. The way the rules were set up (see the implementation of `\__hook_rule_before_gset:nnn` above) ensures that we have no rule in the ignored side of the matrix, and all rules are seen. The rules are applied in `\__hook_apply_label_pair:nnn`, which takes the properly-ordered pair of labels as argument.

```
    \prop_map_inline:Nn \l__hook_work_prop
      {
        \prop_map_inline:Nn \l__hook_work_prop
          {
            \__hook_if_label_case:nnnnn {##1} {####1}
              { \prop_map_break: }
              { \__hook_apply_label_pair:nnn {##1} {####1} }
              { \__hook_apply_label_pair:nnn {####1} {##1} }
                  {#3}
          }
      }
```

Now take a breath, and look at the data structures that have been set up:

```
    \__hook_debug:n { \__hook_debug_label_data:N \l__hook_work_prop }
```

Step T4:

```
    \tl_set:Nn \l__hook_rear_tl { 0 }
    \tl_set:cn { \__hook_tl_csname:n { 0 } } { 0 }
    \seq_map_inline:Nn \l__hook_labels_seq
      {
```

```
        \int_compare:nNnT { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
            {
              \tl_set:cn { \__hook_tl_csname:n { \l__hook_rear_tl } }{##1}
              \tl_set:Nn \l__hook_rear_tl {##1}
            }
      }
    \tl_set_eq:Nc \l__hook_front_tl { \__hook_tl_csname:n { 0 } }
    \__hook_tl_gclear:N #1
    \clist_gclear:N #2
```

The whole loop gets combined in steps T5–T7:

```
    \bool_while_do:nn { ! \str_if_eq_p:Vn \l__hook_front_tl { 0 } }
      {
```

This part is step T5:

```
        \int_decr:N \l__hook_labels_int
        \prop_get:NVN \l__hook_work_prop \l__hook_front_tl \l__hook_return_tl
        \exp_args:NNV \__hook_tl_gput:Nn #1 \l__hook_return_tl
        \__hook_clist_gput:NV #2 \l__hook_front_tl
        \__hook_debug:n{ \iow_term:x{Handled~ code~ for~ \l__hook_front_tl} }
```

This is step T6, except that we don't use a pointer P to move through the successors, but instead use `##1` of the mapping function.

```
        \seq_map_inline:cn { \__hook_seq_csname:n { \l__hook_front_tl } }
          {
            \tl_set:cx { \__hook_tl_csname:n {##1} }
                       { \int_eval:n
                           { \cs:w \__hook_tl_csname:n {##1} \cs_end: - 1 }
                       }
            \int_compare:nNnT
                { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
                {
                  \tl_set:cn
                    { \__hook_tl_csname:n { \l__hook_rear_tl } } {##1}
                  \tl_set:Nn \l__hook_rear_tl            {##1}
                }
          }
```

and here is step T7:

```
        \tl_set_eq:Nc \l__hook_front_tl
                      { \__hook_tl_csname:n { \l__hook_front_tl } }
```

This is step T8: If we haven't moved the code for all labels (i.e., if `\l__hook_labels_int` is still greater than zero) we have a loop and our partial order can't be flattened out.

```
      }
    \int_compare:nNnF \l__hook_labels_int = 0
      {
        \iow_term:x{====================}
        \iow_term:x{Error:~ label~ rules~ are~ incompatible:}
```

This is not really the information one needs in the error case but it will do for now ...

FMi: improve output on a rainy day

```
        \__hook_debug_label_data:N \l__hook_work_prop
        \iow_term:x{====================}
      }
```

After we have added all hook code to `#1`, we finish it off by adding extra code for the `top-level` (`#2`) and for one time execution (`#3`). These should normally be empty. The `top-level` code is added with `\__hook_tl_gput:Nn` as that might change for a reversed hook (then `top-level` is the very first code chunk added). The `next` code is always added last (to the right). The hook may take arguments, so we add a run of braced parameters after the `_next` and `_toplevel` macros, so that the arguments passed to the hook are forwarded to them.

```
    \exp_args:NNe \__hook_tl_gput:Nn #1
      { \exp_not:c { __hook_toplevel~#3 } \__hook_braced_parameter:n {#3} }
    \__hook_tl_gput_right:Ne #1
      { \exp_not:c { __hook_next~#3 } \__hook_braced_parameter:n {#3} }
    \use:e
      {
        \cs_gset:cpn { __hook~#3 } \use:c { c__hook_#3_parameter_tl }
          { \exp_not:V #1 }
      }
  }
```

```
\cs_generate_variant:Nn \__hook_initialize_single:NNn { cc }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_initialize_single:NNn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_new_protected:Npn \__hook_initialize_single:NNn #1#2#3
⟨latexrelease⟩  {
⟨latexrelease⟩    \seq_clear:N \l__hook_labels_seq
⟨latexrelease⟩    \int_zero:N  \l__hook_labels_int
⟨latexrelease⟩    \tl_set:Nn \l__hook_cur_hook_tl {#3}
⟨latexrelease⟩    \prop_map_inline:Nn \l__hook_work_prop
⟨latexrelease⟩       {
⟨latexrelease⟩         \int_incr:N \l__hook_labels_int
⟨latexrelease⟩         \seq_put_right:Nn \l__hook_labels_seq {##1}
⟨latexrelease⟩         \__hook_tl_set:cn { \__hook_tl_csname:n {##1} } { 0 }
⟨latexrelease⟩         \seq_clear_new:c { \__hook_seq_csname:n {##1} }
⟨latexrelease⟩       }
⟨latexrelease⟩    \prop_map_inline:Nn \l__hook_work_prop
⟨latexrelease⟩      {
⟨latexrelease⟩        \prop_map_inline:Nn \l__hook_work_prop
⟨latexrelease⟩          {
⟨latexrelease⟩            \__hook_if_label_case:nnnnn {##1} {####1}
⟨latexrelease⟩              { \prop_map_break: }
⟨latexrelease⟩              { \__hook_apply_label_pair:nnn {##1} {####1} }
⟨latexrelease⟩              { \__hook_apply_label_pair:nnn {####1} {##1} }
⟨latexrelease⟩                  {#3}
⟨latexrelease⟩          }
⟨latexrelease⟩      }
⟨latexrelease⟩    \__hook_debug:n
⟨latexrelease⟩      { \__hook_debug_label_data:N \l__hook_work_prop }
⟨latexrelease⟩    \tl_set:Nn \l__hook_rear_tl { 0 }
⟨latexrelease⟩    \tl_set:cn { \__hook_tl_csname:n { 0 } } { 0 }
⟨latexrelease⟩    \seq_map_inline:Nn \l__hook_labels_seq
```

```
⟨latexrelease⟩      {
⟨latexrelease⟩        \int_compare:nNnT
⟨latexrelease⟩          { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
⟨latexrelease⟩          {
⟨latexrelease⟩            \tl_set:cn { \__hook_tl_csname:n
⟨latexrelease⟩                           { \l__hook_rear_tl } } {##1}
⟨latexrelease⟩            \tl_set:Nn \l__hook_rear_tl           {##1}
⟨latexrelease⟩          }
⟨latexrelease⟩      }
⟨latexrelease⟩    \tl_set_eq:Nc \l__hook_front_tl { \__hook_tl_csname:n { 0 } }
⟨latexrelease⟩    \__hook_tl_gclear:N #1
⟨latexrelease⟩    \clist_gclear:N #2
⟨latexrelease⟩    \bool_while_do:nn
⟨latexrelease⟩      { ! \str_if_eq_p:Vn \l__hook_front_tl { 0 } }
⟨latexrelease⟩      {
⟨latexrelease⟩        \int_decr:N \l__hook_labels_int
⟨latexrelease⟩        \prop_get:NVN \l__hook_work_prop
⟨latexrelease⟩            \l__hook_front_tl \l__hook_return_tl
⟨latexrelease⟩        \exp_args:NNV \__hook_tl_gput:Nn #1 \l__hook_return_tl
⟨latexrelease⟩        \__hook_clist_gput:NV #2 \l__hook_front_tl
⟨latexrelease⟩        \__hook_debug:n{ \iow_term:x
⟨latexrelease⟩              {Handled~ code~ for~ \l__hook_front_tl} }
⟨latexrelease⟩        \seq_map_inline:cn
⟨latexrelease⟩              { \__hook_seq_csname:n { \l__hook_front_tl } }
⟨latexrelease⟩          {
⟨latexrelease⟩            \tl_set:cx { \__hook_tl_csname:n {##1} }
⟨latexrelease⟩              { \int_eval:n
⟨latexrelease⟩                { \cs:w \__hook_tl_csname:n {##1} \cs_end: - 1 }
⟨latexrelease⟩              }
⟨latexrelease⟩            \int_compare:nNnT
⟨latexrelease⟩              { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
⟨latexrelease⟩              {
⟨latexrelease⟩                \tl_set:cn { \__hook_tl_csname:n
⟨latexrelease⟩                               { \l__hook_rear_tl } } {##1}
⟨latexrelease⟩                \tl_set:Nn \l__hook_rear_tl           {##1}
⟨latexrelease⟩              }
⟨latexrelease⟩          }
⟨latexrelease⟩        \tl_set_eq:Nc \l__hook_front_tl
⟨latexrelease⟩          { \__hook_tl_csname:n { \l__hook_front_tl } }
⟨latexrelease⟩      }
⟨latexrelease⟩    \int_compare:nNnF \l__hook_labels_int = 0
⟨latexrelease⟩      {
⟨latexrelease⟩        \iow_term:x{====================}
⟨latexrelease⟩        \iow_term:x{Error:~ label~ rules~ are~ incompatible:}
⟨latexrelease⟩        \__hook_debug_label_data:N \l__hook_work_prop
⟨latexrelease⟩        \iow_term:x{====================}
⟨latexrelease⟩      }
⟨latexrelease⟩    \exp_args:NNo \__hook_tl_gput:Nn #1
⟨latexrelease⟩                    { \cs:w __hook_toplevel~#3 \cs_end: }
⟨latexrelease⟩    \__hook_tl_gput_right:No #1 { \cs:w __hook_next~#3 \cs_end: }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_generate_variant:Nn \__hook_tl_gput_right:Nn { No }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_initialize_single:NNn`.)

__hook_tl_gput:Nn
__hook_clist_gput:NV

These append either on the right (normal hook) or on the left (reversed hook). This is setup up in __hook_initialize_hook_code:n, elsewhere their behavior is undefined.

```
\cs_new:Npn \__hook_tl_gput:Nn    { \ERROR }
\cs_new:Npn \__hook_clist_gput:NV { \ERROR }
```

(*End of definition for* __hook_tl_gput:Nn *and* __hook_clist_gput:NV.)

__hook_apply_label_pair:nnn
__hook_label_if_exist_apply:nnnF

This is the payload of steps T2 and T3 executed in the loop described above. This macro assumes #1 and #2 are ordered, which means that any rule pertaining the pair #1 and #2 is \g__hook_⟨*hook*⟩_rule_#1|#2_tl, and not \g__hook_⟨*hook*⟩_rule_#2|#1_tl. This also saves a great deal of time since we only need to check the order of the labels once.

The arguments here are ⟨*label1*⟩, ⟨*label2*⟩, ⟨*hook*⟩, and ⟨*hook-code-plist*⟩. We are about to apply the next rule and enter it into the data structure. __hook_apply_label_pair:nnn will just call __hook_label_if_exist_apply:nnnF for the ⟨*hook*⟩, and if no rule is found, also try the ⟨*hook*⟩ name ?? denoting a default hook rule.

__hook_label_if_exist_apply:nnnF will check if the rule exists for the given hook, and if so call __hook_apply_rule:nnn.

```
\cs_new_protected:Npn \__hook_apply_label_pair:nnn #1#2#3
  {
```

Extra complication: as we use default rules and local hook specific rules we first have to check if there is a local rule and if that exist use it. Otherwise check if there is a default rule and use that.

```
    \__hook_label_if_exist_apply:nnnF {#1} {#2} {#3}
      {
```

If there is no hook-specific rule we check for a default one and use that if it exists.

```
        \__hook_label_if_exist_apply:nnnF {#1} {#2} { ?? } { }
      }
  }
\cs_new_protected:Npn \__hook_label_if_exist_apply:nnnF #1#2#3
  {
    \if_cs_exist:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end:
```

What to do precisely depends on the type of rule we have encountered. If it is a before rule it will be handled by the algorithm but other types need to be managed differently. All this is done in __hook_apply_rule:nnnN.

```
      \__hook_apply_rule:nnn {#1} {#2} {#3}
      \exp_after:wN \use_none:n
    \else:
      \use:nn
    \fi:
  }
```

(*End of definition for* __hook_apply_label_pair:nnn *and* __hook_label_if_exist_apply:nnnF.)

__hook_apply_rule:nnn

This is the code executed in steps T2 and T3 while looping through the matrix This is part of step T3. We are about to apply the next rule and enter it into the data structure. The arguments are ⟨*label1*⟩, ⟨*label2*⟩, ⟨*hook-name*⟩, and ⟨*hook-code-plist*⟩.

```
\cs_new_protected:Npn \__hook_apply_rule:nnn #1#2#3
  {
    \cs:w __hook_apply_
      \cs:w g__hook_#3_reversed_tl \cs_end: rule_
        \cs:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end: :nnn \cs_end:
```

```
      {#1} {#2} {#3}
  }
```

(*End of definition for* `\__hook_apply_rule:nnn`.)

`\__hook_apply_rule_<:nnn`
`\__hook_apply_rule_>:nnn`

The most common cases are `<` and `>` so we handle that first. They are relations $\prec$ and $\succ$ in TAOCP, and they dictate sorting.

```
\cs_new_protected:cpn { __hook_apply_rule_<:nnn } #1#2#3
  {
    \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
    \tl_set:cx { \__hook_tl_csname:n {#2} }
       { \int_eval:n{ \cs:w \__hook_tl_csname:n {#2} \cs_end: + 1 } }
    \seq_put_right:cn{ \__hook_seq_csname:n {#1} }{#2}
  }
\cs_new_protected:cpn { __hook_apply_rule_>:nnn } #1#2#3
  {
    \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
    \tl_set:cx { \__hook_tl_csname:n {#1} }
       { \int_eval:n{ \cs:w \__hook_tl_csname:n {#1} \cs_end: + 1 } }
    \seq_put_right:cn{ \__hook_seq_csname:n {#2} }{#1}
  }
```

(*End of definition for* `\__hook_apply_rule_<:nnn` *and* `\__hook_apply_rule_>:nnn`.)

`\__hook_apply_rule_xE:nnn`
`\__hook_apply_rule_xW:nnn`

These relations make two labels incompatible within a hook. `xE` makes raises an error if the labels are found in the same hook, and `xW` makes it a warning.

```
\cs_new_protected:cpn { __hook_apply_rule_xE:nnn } #1#2#3
  {
    \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
    \msg_error:nnnnnn { hooks } { labels-incompatible }
      {#1} {#2} {#3} { 1 }
    \use:c { __hook_apply_rule_->:nnn } {#1} {#2} {#3}
    \use:c { __hook_apply_rule_<-:nnn } {#1} {#2} {#3}
  }
\cs_new_protected:cpn { __hook_apply_rule_xW:nnn } #1#2#3
  {
    \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
    \msg_warning:nnnnnn { hooks } { labels-incompatible }
      {#1} {#2} {#3} { 0 }
  }
```

(*End of definition for* `\__hook_apply_rule_xE:nnn` *and* `\__hook_apply_rule_xW:nnn`.)

`\__hook_apply_rule_->:nnn`
`\__hook_apply_rule_<-:nnn`

If we see `->` we have to drop code for label `#3` and carry on. We could do a little better and drop everything for that label since it doesn't matter where we put such empty code. However that would complicate the algorithm a lot with little gain.[12] So we still unnecessarily try to sort it in and depending on the rules that might result in a loop that is otherwise resolved. If that turns out to be a real issue, we can improve the code.

Here the code is removed from `\l__hook_cur_hook_tl` rather than `#3` because the latter may be `??`, and the default hook doesn't store any code. Removing it instead from `\l__hook_cur_hook_tl` makes the default rules `->` and `<-` work properly.

[12] This also has the advantage that the result of the sorting doesn't change, as it might otherwise do (for unrelated chunks) if we aren't careful.

```
\cs_new_protected:cpn { __hook_apply_rule_->:nnn } #1#2#3
  {
    \__hook_debug:n
       {
         \__hook_msg_pair_found:nnn {#1} {#2} {#3}
         \iow_term:x{--->~ Drop~ '#2'~ code~ from~
           \iow_char:N \\ g__hook_ \l__hook_cur_hook_tl _code_prop ~
           because~ of~ '#1' }
       }
    \prop_put:Nnn \l__hook_work_prop {#2} { }
  }
\cs_new_protected:cpn { __hook_apply_rule_<-:nnn } #1#2#3
  {
    \__hook_debug:n
       {
         \__hook_msg_pair_found:nnn {#1} {#2} {#3}
         \iow_term:x{--->~ Drop~ '#1'~ code~ from~
           \iow_char:N \\ g__hook_ \l__hook_cur_hook_tl _code_prop ~
           because~ of~ '#2' }
       }
    \prop_put:Nnn \l__hook_work_prop {#1} { }
  }
```

(*End of definition for* `\__hook_apply_rule_->:nnn` *and* `\__hook_apply_rule_<-:nnn`.)

`\__hook_apply_-rule_<:nnn`
`\__hook_apply_-rule_>:nnn`
`\__hook_apply_-rule_<-:nnn`
`\__hook_apply_-rule_->:nnn`
`\__hook_apply_-rule_xW:nnn`
`\__hook_apply_-rule_xE:nnn`

Reversed rules.

```
\cs_new_eq:cc { __hook_apply_-rule_<:nnn } { __hook_apply_rule_>:nnn }
\cs_new_eq:cc { __hook_apply_-rule_>:nnn } { __hook_apply_rule_<:nnn }
\cs_new_eq:cc { __hook_apply_-rule_<-:nnn } { __hook_apply_rule_<-:nnn }
\cs_new_eq:cc { __hook_apply_-rule_->:nnn } { __hook_apply_rule_->:nnn }
\cs_new_eq:cc { __hook_apply_-rule_xE:nnn } { __hook_apply_rule_xE:nnn }
\cs_new_eq:cc { __hook_apply_-rule_xW:nnn } { __hook_apply_rule_xW:nnn }
```

(*End of definition for* `\__hook_apply_-rule_<:nnn` *and others.*)

`\__hook_msg_pair_found:nnn`

A macro to avoid moving this many tokens around.

```
\cs_new_protected:Npn \__hook_msg_pair_found:nnn #1#2#3
  {
    \iow_term:x{~ \str_if_eq:nnTF {#3} {??} {default} {~normal} ~
        rule~ \__hook_label_pair:nn {#1} {#2}:~
        \use:c { g__hook_#3_rule_ \__hook_label_pair:nn {#1} {#2} _tl } ~
        found}
  }
```

(*End of definition for* `\__hook_msg_pair_found:nnn`.)

`\__hook_debug_label_data:N`

```
\cs_new_protected:Npn \__hook_debug_label_data:N #1 {
  \iow_term:x{Code~ labels~ for~ sorting:}
  \iow_term:x{~ \seq_use:Nnnn\l__hook_labels_seq {~and~}{,~}{~and~} }
  \iow_term:x{^^J Data~ structure~ for~ label~ rules:}
  \prop_map_inline:Nn #1
       {
         \iow_term:x{~ ##1~ =~ \tl_use:c{ \__hook_tl_csname:n {##1} }~ ->~
           \seq_use:cnnn{ \__hook_seq_csname:n {##1} }{~->~}{~->~}{~->~}
```

```
          }
        }
  \iow_term:x{}
}
```

(*End of definition for* `\__hook_debug_label_data:N`.)

`\hook_show:n`
`\hook_log:n`
`\__hook_log_line:x`
`\__hook_log_line_indent:x`
`\__hook_log:nN`

This writes out information about the hook given in its argument onto the `.log` file and the terminal, if `\show_hook:n` is used. Internally both share the same structure, except that at the end, `\hook_show:n` triggers TEX's prompt.

```
\cs_new_protected:Npn \hook_log:n #1
  {
    \cs_set_eq:NN \__hook_log_cmd:x \iow_log:x
    \__hook_normalize_hook_args:Nn \__hook_log:nN {#1} \tl_log:x
  }
\cs_new_protected:Npn \hook_show:n #1
  {
    \cs_set_eq:NN \__hook_log_cmd:x \iow_term:x
    \__hook_normalize_hook_args:Nn \__hook_log:nN {#1} \tl_show:x
  }
\cs_new_protected:Npn \__hook_log_line:x #1
  { \__hook_log_cmd:x { >~#1 } }
\cs_new_protected:Npn \__hook_log_line_indent:x #1
  { \__hook_log_cmd:x { >~\@spaces #1 } }
```

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_log:nN}
⟨latexrelease⟩                    {Hooks~with~args}
\cs_new_protected:Npn \__hook_log:nN #1 #2
  {
    \__hook_if_deprecated_generic:nT {#1}
      {
        \__hook_deprecated_generic_warn:n {#1}
        \__hook_do_deprecated_generic:Nn \__hook_log:nN {#1} #2
        \exp_after:wN \use_none:nnnnnnnnn \use_none:nnnnn
      }
    \__hook_preamble_hook:n {#1}
    \__hook_log_cmd:x
      {
        ^^J ->~The~
        \__hook_if_generic:nT {#1} { generic~ }
        hook~'#1'
        \__hook_if_disabled:nF {#1}
          {
            \exp_args:Nne \__hook_print_args:nn {#1}
              {
                \int_eval:n
                  { \str_count:e { \__hook_parameter:n {#1} } / 3 }
              }
          }
        :
      }

    \__hook_if_usable:nF {#1}
      { \__hook_log_line:x { The~hook~is~not~declared. } }
    \__hook_if_disabled:nT {#1}
```

```
    { \__hook_log_line:x { The~hook~is~disabled. } }
  \hook_if_empty:nTF {#1}
    { #2 { The~hook~is~empty } }
    {
      \__hook_log_line:x { Code~chunks: }
      \bool_lazy_or:nnTF
        { ! \prop_if_exist_p:c { g__hook_#1_code_prop } }
        { \prop_if_empty_p:c { g__hook_#1_code_prop } }
        { \__hook_log_line_indent:x { --- } }
        {
          \prop_map_inline:cn { g__hook_#1_code_prop }
            {
              \exp_after:wN \cs_set:Npn \exp_after:wN \__hook_tmp:w
                \c__hook_nine_parameters_tl {##2}
              \__hook_log_line_indent:x
                { ##1~->~\cs_replacement_spec:N \__hook_tmp:w }
            }
        }
```

If there is code in the `top-level` token list, print it:

```
      \__hook_log_line:x
        {
          Document-level~(top-level)~code
          \__hook_if_usable:nT {#1}
            { ~(executed~\__hook_if_reversed:nTF {#1} {first} {last} ) } :
        }
      \__hook_log_line_indent:x
        {
          \__hook_cs_if_empty:cTF { __hook_toplevel~#1 }
            { --- }
            { -> ~ \cs_replacement_spec:c { __hook_toplevel~#1 } }
        }
```

```
      \__hook_log_line:x { Extra~code~for~next~invocation: }
      \__hook_log_line_indent:x
        {
          \__hook_cs_if_empty:cTF { __hook_next~#1 }
            { --- }
```

If the token list is not empty we want to display it but without the first tokens (the code to clear itself) so we call a helper command to get rid of them.

```
            {
              -> ~ \exp_last_unbraced:Nf \__hook_log_next_code:w
                { \cs_replacement_spec:c { __hook_next~#1 } }
            }
        }
```

Loop through the rules in a hook and for every rule found, print it. If no rule is there, print `---`. The boolean `\l__hook_tmpa_bool` here indicates if the hook has no rules.

```
      \__hook_log_line:x { Rules: }
      \bool_set_true:N \l__hook_tmpa_bool
      \__hook_list_rules:nn {#1}
        {
          \bool_set_false:N \l__hook_tmpa_bool
```

```
            \__hook_log_line_indent:x
              {
                ##2~ with~
                \str_if_eq:nnT {##3} {??} { default~ }
                relation~ ##1
              }
          }
        \bool_if:NT \l__hook_tmpa_bool
          { \__hook_log_line_indent:x { --- } }
```

When the hook is declared (that is, the sorting algorithm is applied to that hook) and not empty

```
        \bool_lazy_and:nnTF
            { \__hook_if_usable_p:n {#1} }
            { ! \hook_if_empty_p:n {#1} }
          {
            \__hook_log_line:x
              {
                Execution~order
                \bool_if:NTF \l__hook_tmpa_bool
                  { \__hook_if_reversed:nT {#1} { ~(after~reversal) } }
                  { ~(after~
                    \__hook_if_reversed:nT {#1} { reversal~and~ }
                    applying~rules)
                  } :
              }
            #2 % \tl_show:n
              {
                \@spaces
                \clist_if_empty:cTF { g__hook_#1_labels_clist }
                  { --- }
                  { \clist_use:cn { g__hook_#1_labels_clist } { ,~ } }
              }
          }
          {
            \__hook_log_line:x { Execution~order: }
            #2
              {
                \@spaces Not~set~because~the~hook~ \__hook_if_usable:nTF {#1}
                  { code~pool~is~empty }
                  { is~\__hook_if_disabled:nTF {#1} {disabled} {undeclared} }
              }
          }
      }
  }
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_log:nN}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_new_protected:Npn \__hook_log:nN #1 #2
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_deprecated_generic:nT {#1}
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_deprecated_generic_warn:n {#1}
```

```
⟨latexrelease⟩        \__hook_do_deprecated_generic:Nn \__hook_log:nN {#1} #2
⟨latexrelease⟩        \exp_after:wN \use_none:nnnnnnnnn \use_none:nnnnn
⟨latexrelease⟩      }
⟨latexrelease⟩    \__hook_preamble_hook:n {#1}
⟨latexrelease⟩    \__hook_log_cmd:x
⟨latexrelease⟩      { ^^J ->~The~ \__hook_if_generic:nT
⟨latexrelease⟩                      {#1} { generic~ } hook~'#1': }
⟨latexrelease⟩    \__hook_if_usable:nF {#1}
⟨latexrelease⟩      { \__hook_log_line:x { The~hook~is~not~declared. } }
⟨latexrelease⟩    \__hook_if_disabled:nT {#1}
⟨latexrelease⟩      { \__hook_log_line:x { The~hook~is~disabled. } }
⟨latexrelease⟩    \hook_if_empty:nTF {#1}
⟨latexrelease⟩      { #2 { The~hook~is~empty } }
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_log_line:x { Code~chunks: }
⟨latexrelease⟩        \prop_if_empty:cTF { g__hook_#1_code_prop }
⟨latexrelease⟩          { \__hook_log_line_indent:x { --- } }
⟨latexrelease⟩          {
⟨latexrelease⟩            \prop_map_inline:cn { g__hook_#1_code_prop }
⟨latexrelease⟩              { \__hook_log_line_indent:x
⟨latexrelease⟩                  { ##1~->~\tl_to_str:n {##2} } }
⟨latexrelease⟩          }
⟨latexrelease⟩        \__hook_log_line:x
⟨latexrelease⟩          {
⟨latexrelease⟩            Document-level~(top-level)~code
⟨latexrelease⟩            \__hook_if_usable:nT {#1}
⟨latexrelease⟩              { ~(executed~
⟨latexrelease⟩                \__hook_if_reversed:nTF {#1} {first} {last} ) } :
⟨latexrelease⟩          }
⟨latexrelease⟩        \__hook_log_line_indent:x
⟨latexrelease⟩          {
⟨latexrelease⟩            \tl_if_empty:cTF { __hook_toplevel~#1 }
⟨latexrelease⟩              { --- }
⟨latexrelease⟩              { -> ~ \exp_args:Nv \tl_to_str:n
⟨latexrelease⟩                                { __hook_toplevel~#1 } }
⟨latexrelease⟩          }
⟨latexrelease⟩        \__hook_log_line:x { Extra~code~for~next~invocation: }
⟨latexrelease⟩        \__hook_log_line_indent:x
⟨latexrelease⟩          {
⟨latexrelease⟩            \tl_if_empty:cTF { __hook_next~#1 }
⟨latexrelease⟩              { --- }
⟨latexrelease⟩              { ->~ \exp_args:Nv \__hook_log_next_code:n
⟨latexrelease⟩                                 { __hook_next~#1 } }
⟨latexrelease⟩          }
⟨latexrelease⟩        \__hook_log_line:x { Rules: }
⟨latexrelease⟩        \bool_set_true:N \l__hook_tmpa_bool
⟨latexrelease⟩        \__hook_list_rules:nn {#1}
⟨latexrelease⟩          {
⟨latexrelease⟩            \bool_set_false:N \l__hook_tmpa_bool
⟨latexrelease⟩            \__hook_log_line_indent:x
⟨latexrelease⟩              {
⟨latexrelease⟩                ##2~ with~
⟨latexrelease⟩                \str_if_eq:nnT {##3} {??} { default~ }
⟨latexrelease⟩                relation~ ##1
```

```
⟨latexrelease⟩                }
⟨latexrelease⟩            }
⟨latexrelease⟩          \bool_if:NT \l__hook_tmpa_bool
⟨latexrelease⟩            { \__hook_log_line_indent:x { --- } }
⟨latexrelease⟩          \bool_lazy_and:nnTF
⟨latexrelease⟩              { \__hook_if_usable_p:n {#1} }
⟨latexrelease⟩              { ! \hook_if_empty_p:n {#1} }
⟨latexrelease⟩            {
⟨latexrelease⟩              \__hook_log_line:x
⟨latexrelease⟩                {
⟨latexrelease⟩                  Execution~order
⟨latexrelease⟩                  \bool_if:NTF \l__hook_tmpa_bool
⟨latexrelease⟩                    { \__hook_if_reversed:nT
⟨latexrelease⟩                         {#1}{ ~(after~reversal) } }
⟨latexrelease⟩                    { ~(after~
⟨latexrelease⟩                      \__hook_if_reversed:nT {#1} { reversal~and~ }
⟨latexrelease⟩                      applying~rules)
⟨latexrelease⟩                    } :
⟨latexrelease⟩                }
⟨latexrelease⟩              #2 % \tl_show:n
⟨latexrelease⟩                {
⟨latexrelease⟩                  \@spaces
⟨latexrelease⟩                  \clist_if_empty:cTF { g__hook_#1_labels_clist }
⟨latexrelease⟩                    { --- }
⟨latexrelease⟩                    { \clist_use:cn
⟨latexrelease⟩                        { g__hook_#1_labels_clist } { ,~ } }
⟨latexrelease⟩                }
⟨latexrelease⟩            }
⟨latexrelease⟩            {
⟨latexrelease⟩              \__hook_log_line:x { Execution~order: }
⟨latexrelease⟩              #2
⟨latexrelease⟩                {
⟨latexrelease⟩                  \@spaces Not~set~because~the~hook~
⟨latexrelease⟩                      \__hook_if_usable:nTF {#1}
⟨latexrelease⟩                    { code~pool~is~empty }
⟨latexrelease⟩                    { is~\__hook_if_disabled:nTF
⟨latexrelease⟩                        {#1} {disabled} {undeclared} }
⟨latexrelease⟩                }
⟨latexrelease⟩            }
⟨latexrelease⟩        }
⟨latexrelease⟩    }
⟨latexrelease⟩\EndIncludeInRelease
```

To display the code for next invocation only (i.e., from `\AddToHookNext` we have to remove the string `\__hook_clear_next:n{`⟨*hook*⟩`}`, so the simplest is to use a macro delimited by a `}`$_1$2.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_log_next_code:n}
⟨latexrelease⟩                  {Hooks~with~args}
\__hook_log_next_code:n \exp_last_unbraced:NNNNo
\cs_new:Npn \__hook_log_next_code:w #1 \c_right_brace_str { }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_log_next_code:n}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset:Npn \__hook_log_next_code:n #1
```

```
⟨latexrelease⟩  { \exp_args:No \tl_to_str:n { \use_none:nn #1 } }
⟨latexrelease⟩\EndIncludeInRelease
```

Pretty-prints the number of arguments of a hook.

__hook_print_args:n

```
\cs_new:Npn \__hook_print_args:nn #1 #2
  {
    \int_compare:nNnT {#2} > { 0 }
      {
        \__hook_if_declared:nT {#1} { \use_none:nnn }
        \__hook_if_cmd_hook:nT {#1}
          { \use_i:nnn { ~ (unknown ~ } }
        \use:n { ~ (#2 ~ }
        argument \int_compare:nNnT {#2} > { 1 } { s } )
      }
  }
```

(*End of definition for* `\hook_show:n` *and others. These functions are documented on page* 209.)

__hook_list_rules:nn
__hook_list_one_rule:nnn
__hook_list_if_rule_exists:nnnF

This macro takes a ⟨*hook*⟩ and an ⟨*inline function*⟩ and loops through each pair of ⟨*labels*⟩ in the ⟨*hook*⟩, and if there is a relation between this pair of ⟨*labels*⟩, the ⟨*inline function*⟩ is executed with `#1` = ⟨*relation*⟩, `#2` = ⟨*label*$_1$⟩`|`⟨*label*$_2$⟩, and `#3` = ⟨*hook*⟩ (the latter may be the argument `#1` to `\__hook_list_rules:nn`, or `??` if it is a default rule).

```
\cs_new_protected:Npn \__hook_list_rules:nn #1 #2
  {
    \prop_if_exist:cT { g__hook_#1_code_prop }
      {
        \cs_set_protected:Npn \__hook_tmp:w ##1 ##2 ##3 {#2}
        \prop_map_inline:cn { g__hook_#1_code_prop }
          {
            \prop_map_inline:cn { g__hook_#1_code_prop }
              {
                \__hook_if_label_case:nnnnn {##1} {####1}
                  { \prop_map_break: }
                  { \__hook_list_one_rule:nnn {##1} {####1} }
                  { \__hook_list_one_rule:nnn {####1} {##1} }
                      {#1}
              }
          }
      }
  }
```

These two are quite similar to `\__hook_apply_label_pair:nnn` and `\__hook_label_if_exist_apply:nnnF`, respectively, but rather than applying the rule, they pass it to the ⟨*inline function*⟩.

```
\cs_new_protected:Npn \__hook_list_one_rule:nnn #1#2#3
  {
    \__hook_list_if_rule_exists:nnnF {#1} {#2} {#3}
      { \__hook_list_if_rule_exists:nnnF {#1} {#2} { ?? } { } }
  }
\cs_new_protected:Npn \__hook_list_if_rule_exists:nnnF #1#2#3
  {
    \if_cs_exist:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end:
      \exp_args:Nv \__hook_tmp:w
```

```
        { g__hook_ #3 _rule_ #1 | #2 _tl } { #1 | #2 } {#3}
      \exp_after:wN \use_none:nn
    \fi:
    \use:n
  }
```

(*End of definition for* `\__hook_list_rules:nn`, `\__hook_list_one_rule:nnn`, *and* `\__hook_list_if_rule_exists:nnnF`.)

`\__hook_debug_print_rules:n` A shorthand for debugging that prints similar to `\prop_show:N`.

```
\cs_new_protected:Npn \__hook_debug_print_rules:n #1
  {
    \iow_term:n { The~hook~#1~contains~the~rules: }
    \cs_set_protected:Npn \__hook_tmp:w ##1
      {
        \__hook_list_rules:nn {#1}
          {
            \iow_term:x
              {
                > ##1 {####2} ##1 => ##1 {####1}
                \str_if_eq:nnT {####3} {??} { ~(default) }
              }
          }
      }
    \exp_args:No \__hook_tmp:w { \use:nn { ~ } { ~ } }
  }
```

(*End of definition for* `\__hook_debug_print_rules:n`.)

4.8 Specifying code for next invocation

`\hook_gput_next_code:nn`

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_gput_next_code:nn}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \hook_gput_next_code:nn #1 #2
  {
    \__hook_replacing_args_false:
    \__hook_normalize_hook_args:Nn \__hook_gput_next_code:nn {#1} {#2}
    \__hook_replacing_args_reset:
  }
\cs_new_protected:Npn \hook_gput_next_code_with_args:nn #1 #2
  {
    \__hook_replacing_args_true:
    \__hook_normalize_hook_args:Nn \__hook_gput_next_code:nn {#1} {#2}
    \__hook_replacing_args_reset:
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_gput_next_code:nn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \hook_gput_next_code:nn #1
⟨latexrelease⟩  { \__hook_normalize_hook_args:Nn
⟨latexrelease⟩        \__hook_gput_next_code:nn {#1} }
⟨latexrelease⟩\cs_gset_protected:Npn \hook_gput_next_code_with_args:nn #1 #2 { }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_gput_next_code:nn`. *This function is documented on page* 208.)

`\__hook_gput_next_code:nn`

```
\cs_new_protected:Npn \__hook_gput_next_code:nn #1 #2
  {
    \__hook_if_disabled:nTF {#1}
      { \msg_error:nnn { hooks } { hook-disabled } {#1} }
      {
        \__hook_if_structure_exist:nTF {#1}
          { \__hook_gput_next_do:nn }
          { \__hook_try_declaring_generic_next_hook:nn }
              {#1} {#2}
      }
  }
```

(*End of definition for* `\__hook_gput_next_code:nn`.)

`\__hook_gput_next_do:nn` Start by sanity-checking with `\__hook_chk_args_allowed:nn`. Then check if the "next code" token list is empty: if so we need to add a `\tl_gclear:c` to clear it, so the code lasts for one usage only. The token list is cleared early so that nested usages don't get lost. `\tl_gclear:c` is used instead of `\tl_gclear:N` in case the hook is used in an expansion-only context, so the token list doesn't expand before `\tl_gclear:N`: that would make an infinite loop. Also in case the main code token list is empty, the hook code has to be updated to add the next execution token list.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_gput_next_do:nn}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_gput_next_do:nn #1
  {
    \__hook_init_structure:n {#1}
    \__hook_chk_args_allowed:nn {#1} { AddToHookNext }
    \__hook_cs_if_empty:cT { __hook~#1 }
      { \__hook_update_hook_code:n {#1} }
    \__hook_cs_if_empty:cT { __hook_next~#1 }
      { \__hook_next_gset:nn {#1} { \__hook_clear_next:n {#1} } }
    \__hook_cs_gput_right:nnn { _next } {#1}
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_gput_next_do:nn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_gput_next_do:nn #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \exp_args:Nc \__hook_gput_next_do:Nnn
⟨latexrelease⟩      { __hook_next~#1 } {#1}
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_gput_next_do:Nnn #1 #2
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_if_empty:cT { __hook~#2 }
⟨latexrelease⟩      { \__hook_update_hook_code:n {#2} }
⟨latexrelease⟩    \tl_if_empty:NT #1
⟨latexrelease⟩      { \__hook_tl_gset:Nn #1 { \__hook_clear_next:n {#2} } }
⟨latexrelease⟩    \__hook_tl_gput_right:Nn #1
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_gput_next_do:nn`.)

`\hook_gclear_next_code:n` Discard anything set up for next invocation of the hook.

```
\cs_new_protected:Npn \hook_gclear_next_code:n #1
  { \__hook_normalize_hook_args:Nn \__hook_clear_next:n {#1} }
```

(*End of definition for* `\hook_gclear_next_code:n`. *This function is documented on page* 208.)

`\__hook_clear_next:n`

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_clear_next:n}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_clear_next:n #1
  { \__hook_next_gset:nn {#1} { } }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_clear_next:n}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_clear_next:n #1
⟨latexrelease⟩  { \cs_gset_eq:cN { __hook_next~#1 } \c_empty_tl }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_clear_next:n`.)

4.9 Using the hook

`\hook_use:n`
`\__hook_use_initialized:n`
`\__hook_preamble_hook:n`

`\hook_use:n` as defined here is used in the preamble, where hooks aren't initialized by default. `\__hook_use_initialized:n` is also defined, which is the non-`\protected` version for use within the document. Their definition is identical, except for the `\__hook_preamble_hook:n` (which wouldn't hurt in the expandable version, but it would be an unnecessary extra expansion).

`\__hook_use_initialized:n` holds the expandable definition while in the preamble. `\__hook_preamble_hook:n` initializes the hook in the preamble, and is redefined to `\use_none:n` at `\begin{document}`.

Both versions do the same thing internally: they check that the hook exists as given, and if so they use it as quickly as possible.

At `\begin{document}`, all hooks are initialized, and any change in them causes an update, so `\hook_use:n` can be made expandable. This one is better not protected so that it can expand into nothing if containing no code. Also important in case of generic hooks that we do not generate a `\relax` as a side effect of checking for a csname. In contrast to the TEX low-level `\csname ...\endcsname` construct `\tl_if_exist:c` is careful to avoid this.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_use:n}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \hook_use:n #1
  {
    \__hook_preamble_hook:n {#1}
    \__hook_use_initialized:n {#1}
  }
\cs_new:Npn \__hook_use_initialized:n #1
  {
    \if_cs_exist:w __hook~#1 \cs_end:
      \cs:w __hook~#1 \use_i:nn
    \fi:
    \use_none:n
```

```
    \cs_end:
  }
\cs_new_protected:Npn \__hook_preamble_hook:n #1
  {
    \if_cs_exist:w __hook~#1 \cs_end:
      \__hook_initialize_hook_code:n {#1}
    \fi:
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\hook_use:n}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\cs_new_protected:Npn \hook_use:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_if_exist:cT { __hook~#1 }
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_preamble_hook:n {#1}
⟨latexrelease⟩        \cs:w __hook~#1 \cs_end:
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new:Npn \__hook_use_initialized:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \if_cs_exist:w __hook~#1 \cs_end:
⟨latexrelease⟩      \cs:w __hook~#1 \exp_after:wN \cs_end:
⟨latexrelease⟩    \fi:
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new_protected:Npn \__hook_preamble_hook:n #1
⟨latexrelease⟩  { \__hook_initialize_hook_code:n {#1} }
⟨latexrelease⟩\cs_new:Npn \hook_use:nnw #1 { }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_use:n}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\cs_new_protected:Npn \hook_use:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_if_exist:cTF { __hook~#1 }
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_preamble_hook:n {#1}
⟨latexrelease⟩        \cs:w __hook~#1 \cs_end:
⟨latexrelease⟩      }
⟨latexrelease⟩      { \__hook_use:wn #1 / \s__hook_mark {#1} }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new:Npn \__hook_use_initialized:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \if_cs_exist:w __hook~#1 \cs_end:
⟨latexrelease⟩    \else:
⟨latexrelease⟩      \__hook_use_undefined:w
⟨latexrelease⟩    \fi:
⟨latexrelease⟩    \cs:w __hook~#1 \__hook_use_end:
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new:Npn \__hook_use_undefined:w
⟨latexrelease⟩    #1 #2 __hook~#3 \__hook_use_end:
⟨latexrelease⟩  {
⟨latexrelease⟩    #1 % fi
⟨latexrelease⟩    \__hook_use:wn #3 / \s__hook_mark {#3}
```

```
⟨latexrelease⟩   }
⟨latexrelease⟩\cs_new_protected:Npn \__hook_preamble_hook:n #1
⟨latexrelease⟩   { \__hook_initialize_hook_code:n {#1} }
⟨latexrelease⟩\cs_new_eq:NN \__hook_use_end: \cs_end:
⟨latexrelease⟩\cs_new:Npn \hook_use:nnw #1 { }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_use:n`, `\__hook_use_initialized:n`, *and* `\__hook_preamble_hook:n`. *This function is documented on page* 207.)

`\hook_use:nnw`
`\__hook_use_initialized:nnw`

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_use:nnw}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \hook_use:nnw #1
  {
    \__hook_preamble_hook:n {#1}
    \__hook_use_initialized:nnw {#1}
  }
\cs_new:Npn \__hook_use_initialized:nnw #1 #2
  {
    \cs:w
      \if_cs_exist:w __hook~#1 \cs_end:
        __hook~#1
      \else:
        use_none: \prg_replicate:nn {#2} { n }
      \fi:
    \cs_end:
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_use:nnw}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset:Npn \hook_use:nnw #1 #2
⟨latexrelease⟩   { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_use:nnw` *and* `\__hook_use_initialized:nnw`. *This function is documented on page* 207.)

`\__hook_post_initialization_defs:`

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_post_initialization_defs:}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_post_initialization_defs:
  {
    \cs_gset_eq:NN \hook_use:n \__hook_use_initialized:n
    \cs_gset_eq:NN \hook_use:nnw \__hook_use_initialized:nnw
    \cs_gset_eq:NN \__hook_preamble_hook:n \use_none:n
    \cs_gset_eq:NN \__hook_post_initialization_defs: \prg_do_nothing:
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_post_initialization_defs:}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_post_initialization_defs:
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_post_initialization_defs:`.)

__hook_use:wn
__hook_try_file_hook:n
__hook_if_usable_use:n

__hook_use:wn does a quick check to test if the current hook is a file hook: those need a special treatment. If it is not, the hook does not exist. If it is, then __hook_try_file_hook:n is called, and checks that the current hook is a file-specific hook using __hook_if_file_hook:wTF. If it's not, then it's a generic file/ hook and is used if it exist.

If it is a file-specific hook, it passes through the same normalization as during declaration, and then it is used if defined. __hook_if_usable_use:n checks if the hook exist, and calls __hook_preamble_hook:n if so, then uses the hook.

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__hook_use:wn}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_use:wn}
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\cs_new:Npn \__hook_use:wn #1 / #2 \s__hook_mark #3
⟨latexrelease⟩  {
⟨latexrelease⟩    \str_if_eq:nnTF {#1} { file }
⟨latexrelease⟩      { \__hook_try_file_hook:n {#3} }
⟨latexrelease⟩      { } % Hook doesn't exist
⟨latexrelease⟩  }

⟨latexrelease⟩\cs_new_protected:Npn \__hook_try_file_hook:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_file_hook:wTF #1 / / \s__hook_mark
⟨latexrelease⟩      {
⟨latexrelease⟩        \exp_args:Ne \__hook_if_usable_use:n
⟨latexrelease⟩          { \exp_args:Ne \__hook_file_hook_normalize:n {#1} }
⟨latexrelease⟩      }
⟨latexrelease⟩      { \__hook_if_usable_use:n {#1} }
⟨latexrelease⟩            % file/ generic hook (e.g. file/before)
⟨latexrelease⟩  }

⟨latexrelease⟩\cs_new_protected:Npn \__hook_if_usable_use:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_if_exist:cT { __hook~#1 }
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_preamble_hook:n {#1}
⟨latexrelease⟩        \cs:w __hook~#1 \cs_end:
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* __hook_use:wn, __hook_try_file_hook:n, *and* __hook_if_usable_use:n.)

\hook_use_once:n
\hook_use_once:nnw

For hooks that can and should be used only once we have a special use command that further inhibits the hook from getting more code added to it. This has the effect that any further code added to the hook is executed immediately rather than stored in the hook.

The code needs some gymnastics to prevent space trimming from the hook name, since \hook_use:n and \hook_use_once:n are documented to not trim spaces.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_use_once:nnw}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \hook_use_once:n #1
  {
    \__hook_if_execute_immediately:nF {#1}
```

```
      { \__hook_normalize_hook_args:Nn \__hook_use_once:nn
          { \use:n {#1} } { 0 } }
  }
\cs_new_protected:Npn \hook_use_once:nnw #1 #2
  {
    \__hook_if_execute_immediately:nF {#1}
      { \__hook_normalize_hook_args:Nn \__hook_use_once:nn
          { \use:n {#1} } {#2} }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_use_once:n` *and* `\hook_use_once:nnw`*. These functions are documented on page 207.*)

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_use_once:nnw}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \hook_use_once:n #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_execute_immediately:nF {#1}
⟨latexrelease⟩      { \__hook_normalize_hook_args:Nn \__hook_use_once:n
⟨latexrelease⟩          { \use:n {#1} } }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_gset:Npn \hook_use_once:nnw #1 #2
⟨latexrelease⟩  { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
⟨latexrelease⟩\EndIncludeInRelease
```

`\__hook_use_once:nn`

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_use_once:nn}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_use_once:nn #1 #2
  {
    \__hook_preamble_hook:n {#1}
    \__hook_use_once_set:n {#1}
```

When a hook has arguments, the call to `\__hook_use_initialized:n`, should be the very last thing to happen, otherwise the arguments grabbed will be wrong. So, to clean up after the hook we need to cheat a bit and sneak the cleanup code at the end of the hook, along with the next execution code.

```
    \__hook_replacing_args_false:
    \__hook_cs_gput_right:nnn { _next } {#1}
      { \__hook_use_once_clear:n {#1} }
    \__hook_replacing_args_reset:
    \__hook_if_usable:nTF {#1}
      { \__hook_use_initialized:n {#1} }
      {
        \int_compare:nNnT {#2} > { 0 }
          { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
      }
  }
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_use_once:nn}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_use_once:n #1
⟨latexrelease⟩  {
```

```
⟨latexrelease⟩     \__hook_preamble_hook:n {#1}
⟨latexrelease⟩     \__hook_use_once_set:n {#1}
⟨latexrelease⟩     \__hook_use_initialized:n {#1}
⟨latexrelease⟩     \__hook_use_once_clear:n {#1}
⟨latexrelease⟩   }
⟨latexrelease⟩\cs_undefine:N \__hook_use_once:nn
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_use_once:nn`.)

`\__hook_use_once_set:n`
`\__hook_use_once_clear:n`

`\__hook_use_once_set:n` is used before the actual hook code is executed so that any usage of `\AddToHook` inside the hook causes the code to execute immediately. Setting `\g__hook_`⟨*hook*⟩`_reversed_tl` to `I` prevents further code from being added to the hook. `\__hook_use_once_clear:n` then clears the hook so that any further call to `\hook_use:n` or `\hook_use_once:n` will expand to nothing.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_use_once_clear:n}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new_protected:Npn \__hook_use_once_set:n #1
  { \__hook_tl_gset:cn { g__hook_#1_reversed_tl } { I } }
\cs_new_protected:Npn \__hook_use_once_clear:n #1
  {
    \__hook_code_gset:nn {#1} { }
    \__hook_next_gset:nn {#1} { }
    \__hook_toplevel_gset:nn {#1} { }
    \prop_gclear_new:c { g__hook_#1_code_prop }
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_use_once_clear:n}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_new_protected:Npn \__hook_use_once_clear:n #1
⟨latexrelease⟩   {
⟨latexrelease⟩     \__hook_tl_gclear:c { __hook~#1 }
⟨latexrelease⟩     \__hook_tl_gclear:c { __hook_next~#1 }
⟨latexrelease⟩     \__hook_tl_gclear:c { __hook_toplevel~#1 }
⟨latexrelease⟩     \prop_gclear_new:c { g__hook_#1_code_prop }
⟨latexrelease⟩   }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_use_once_set:n` *and* `\__hook_use_once_clear:n`.)

`\__hook_if_execute_immediately_p:n`
`\__hook_if_execute_immediately:n`*TF*

To check whether the code being added should be executed immediately (that is, if the hook is a one-time hook), we check if `\g__hook_`⟨*hook*⟩`_reversed_tl` is `I`. The gymnastics around `\if:w` is there to allow the `reversed` token list to be empty.

```
\prg_new_conditional:Npnn \__hook_if_execute_immediately:n #1 { T, F, TF }
  {
    \exp_after:wN \__hook_use_none_delimit_by_s_mark:w
    \if:w I
        \if_cs_exist:w g__hook_#1_reversed_tl \cs_end:
          \cs:w g__hook_#1_reversed_tl \exp_after:wN \cs_end:
        \fi:
        X
      \s__hook_mark \prg_return_true:
    \else:
      \s__hook_mark \prg_return_false:
```

```
    \fi:
  }
```

(*End of definition for* `\__hook_if_execute_immediately:nTF`.)

4.10 Querying a hook

Simpler data types, like token lists, have three possible states; they can exist and be empty, exist and be non-empty, and they may not exist, in which case emptiness doesn't apply (though `\tl_if_empty:N` returns false in this case).

Hooks are a bit more complicated: they have several other states as discussed in 4.4.2. A hook may exist or not, and either way it may or may not be empty (even a hook that doesn't exist may be non-empty) or may be disabled.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its "next" token list. The hook doesn't need to be declared to have code added to its code pool (it may happen that a package A defines a hook `foo`, but it's loaded after package B, which adds some code to that hook. In this case it is important that the code added by package B is remembered until package A is loaded).

All other states can only be queried with internal tests as the different states are irrelevant for package code.

`\hook_if_empty_p:n`
`\hook_if_empty:nTF`

Test if a hook is empty (that is, no code was added to that hook). A ⟨*hook*⟩ being empty means that all three of its `\g__hook_`⟨*hook*⟩`_code_prop`, its `\__hook_toplevel␣`⟨*hook*⟩ and its `\__hook_next␣`⟨*hook*⟩ are empty.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_if_empty:n}
⟨latexrelease⟩                    {Hooks~with~args}
\prg_new_conditional:Npnn \hook_if_empty:n #1 { p , T , F , TF }
  {
    \if:w
        T
        \prop_if_exist:cT { g__hook_#1_code_prop }
          { \prop_if_empty:cF { g__hook_#1_code_prop } { F } }
        \__hook_cs_if_empty:cF { __hook_toplevel~#1 } { F }
        \__hook_cs_if_empty:cF { __hook_next~#1 } { F }
        T
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_if_empty:n}
⟨latexrelease⟩                    {Hooks~with~args}
⟨latexrelease⟩\prg_new_conditional:Npnn \hook_if_empty:n #1 { p , T , F , TF }
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_if_structure_exist:nTF {#1}
⟨latexrelease⟩      {
⟨latexrelease⟩        \bool_lazy_and:nnTF
⟨latexrelease⟩            { \prop_if_empty_p:c { g__hook_#1_code_prop } }
⟨latexrelease⟩            {
⟨latexrelease⟩              \bool_lazy_and_p:nn
⟨latexrelease⟩                { \tl_if_empty_p:c { __hook_toplevel~#1 } }
```

```
⟨latexrelease⟩                { \tl_if_empty_p:c { __hook_next~#1 } }
⟨latexrelease⟩            }
⟨latexrelease⟩          { \prg_return_true: }
⟨latexrelease⟩          { \prg_return_false: }
⟨latexrelease⟩      }
⟨latexrelease⟩      { \prg_return_true: }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\hook_if_empty:nTF`. *This function is documented on page* 209.)

`\__hook_if_usable_p:n`
`\__hook_if_usable:nTF`

A hook is usable if the token list that stores the sorted code for that hook, `\__hook␣⟨hook⟩`, exists. The property list `\g__hook_⟨hook⟩_code_prop` cannot be used here because often it is necessary to add code to a hook without knowing if such hook was already declared, or even if it will ever be (for example, in case the package that defines it isn't loaded).

```
\prg_new_conditional:Npnn \__hook_if_usable:n #1 { p , T , F , TF }
  {
    \cs_if_exist:cTF { __hook~#1 }
      { \prg_return_true: }
      { \prg_return_false: }
  }
```

(*End of definition for* `\__hook_if_usable:nTF`.)

`\__hook_if_structure_exist_p:n`
`__hook_if_structure_exist:nTF`

An internal check if the hook has already its basic internal structure set up with `\__hook_init_structure:n`. This means that the hook was already used somehow (a code chunk or rule was added to it), but it still wasn't declared with `\hook_new:n`.

```
\prg_new_conditional:Npnn \__hook_if_structure_exist:n #1 { p , T , F , TF }
  {
    \prop_if_exist:cTF { g__hook_#1_code_prop }
      { \prg_return_true: }
      { \prg_return_false: }
  }
```

(*End of definition for* `\__hook_if_structure_exist:nTF`.)

`\__hook_if_declared_p:n`
`\__hook_if_declared:nTF`

Internal test to check if the hook was officially declared with `\hook_new:n` or a variant.

```
\prg_new_conditional:Npnn \__hook_if_declared:n #1 { p, T, F, TF }
  {
    \tl_if_exist:cTF { g__hook_#1_declared_tl }
      { \prg_return_true: }
      { \prg_return_false: }
  }
```

(*End of definition for* `\__hook_if_declared:nTF`.)

`\__hook_if_reversed_p:n`
`\__hook_if_reversed:nTF`

An internal conditional that checks if a hook is reversed.

```
\prg_new_conditional:Npnn \__hook_if_reversed:n #1 { p , T , F , TF }
  {
    \exp_after:wN \__hook_use_none_delimit_by_s_mark:w
    \if:w - \cs:w g__hook_#1_reversed_tl \cs_end:
      \s__hook_mark \prg_return_true:
    \else:
```

```
      \s__hook_mark \prg_return_false:
    \fi:
  }
```

(*End of definition for* `\__hook_if_reversed:nTF`.)

`\__hook_if_generic_p:n`
`\__hook_if_generic:nTF`
`\__hook_if_deprecated_generic_p:n`
`\__hook_if_deprecated_generic:nTF`

An internal conditional that checks if a name belongs to a generic hook. The deprecated version needs to check if `#3` is empty to avoid returning true on `file/before`, for example.

```
\prg_new_conditional:Npnn \__hook_if_generic:n #1 { T, TF }
  { \__hook_if_generic:w #1 / / / \s__hook_mark }
\cs_new:Npn \__hook_if_generic:w #1 / #2 / #3 / #4 \s__hook_mark
  {
    \cs_if_exist:cTF { c__hook_generic_#1/./#3_tl }
      { \prg_return_true: }
      { \prg_return_false: }
  }
\prg_new_conditional:Npnn \__hook_if_deprecated_generic:n #1 { T, TF }
  { \__hook_if_deprecated_generic:w #1 / / / \s__hook_mark }
\cs_new:Npn \__hook_if_deprecated_generic:w #1 / #2 / #3 / #4 \s__hook_mark
  {
    \cs_if_exist:cTF { c__hook_deprecated_#1/./#2_tl }
      {
        \tl_if_empty:nTF {#3}
          { \prg_return_false: }
          { \prg_return_true: }
      }
      { \prg_return_false: }
  }
```

(*End of definition for* `\__hook_if_generic:nTF` *and* `\__hook_if_deprecated_generic:nTF`.)

`\__hook_if_cmd_hook_p:n`
`\__hook_if_cmd_hook:nTF`
`\__hook_if_cmd_hook_p:w`
`\__hook_if_cmd_hook:wTF`

An internal conditional that checks if a given hook is a valid generic `cmd` hook.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_if_cmd_hook:n}
⟨latexrelease⟩                  {Hooks~with~args}
\prg_new_conditional:Npnn \__hook_if_cmd_hook:n #1 { T }
  { \__hook_if_cmd_hook:w #1 / / / \s__hook_mark }
\cs_new:Npn \__hook_if_cmd_hook:w #1 / #2 / #3 / #4 \s__hook_mark
  {
    \if:w Y
          \str_if_eq:nnF {#1} { cmd } { N }
          \tl_if_exist:cF { c__hook_generic_#1/./#3_tl } { N }
          Y
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_if_cmd_hook:n}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_if_cmd_hook:nT
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_if_cmd_hook:nTF` *and* `\__hook_if_cmd_hook:wTF`.)

__hook_if_generic_reversed_p:n
__hook_if_generic_reversed:nTF

An internal conditional that checks if a name belongs to a generic reversed hook.

```
\prg_new_conditional:Npnn \__hook_if_generic_reversed:n #1 { T }
  { \__hook_if_generic_reversed:w #1 / / / \scan_stop: }
\cs_new:Npn \__hook_if_generic_reversed:w #1 / #2 / #3 / #4 \scan_stop:
  {
    \if_charcode:w - \cs:w c__hook_generic_#1/./#3_tl \cs_end:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
```

(*End of definition for* `\__hook_if_generic_reversed:nTF`.)

__hook_if_replacing_args:TF
__hook_misused_if_replacing_args:nn
__hook_replacing_args_true:
__hook_replacing_args_false:
__hook_replacing_args_reset:
\g__hook_replacing_stack_seq

An internal conditional that checks if the code being added to the hook contains arguments.

```
\seq_new:N \g__hook_replacing_stack_seq
\cs_new:Npn \__hook_misused_if_replacing_args:nn #1 #2
  {
    \msg_expandable_error:nnn { latex2e } { should-not-happen }
      { Misused~\__hook_if_replacing_args:. }
  }
\cs_new:Npn \__hook_if_replacing_args:TF
  { \__hook_misused_if_replacing_args:nn }
\cs_new_protected:Npn \__hook_replacing_args_true:
  {
    \seq_gpush:No \g__hook_replacing_stack_seq
      { \__hook_if_replacing_args:TF }
    \cs_set:Npn \__hook_if_replacing_args:TF { \use_i:nn }
  }
\cs_new_protected:Npn \__hook_replacing_args_false:
  {
    \seq_gpush:No \g__hook_replacing_stack_seq
      { \__hook_if_replacing_args:TF }
    \cs_set:Npn \__hook_if_replacing_args:TF { \use_ii:nn }
  }
\cs_new_protected:Npn \__hook_replacing_args_reset:
  {
    \seq_gpop:NN \g__hook_replacing_stack_seq \l__hook_return_tl
    \cs_gset_eq:NN \__hook_if_replacing_args:TF \l__hook_return_tl
  }
```

(*End of definition for* `\__hook_if_replacing_args:TF` *and others.*)

4.11 Messages

Hook errors are LaTeX kernel errors:

```
\prop_gput:Nnn \g_msg_module_type_prop { hooks } { LaTeX }
```

And so are kernel errors (this should move elsewhere eventually).

```
\prop_gput:Nnn \g_msg_module_type_prop { latex2e } { LaTeX }
\prop_gput:Nnn \g_msg_module_name_prop { latex2e } { kernel }
```

```
\msg_new:nnnn { hooks } { labels-incompatible }
  {
    Labels~'#1'~and~'#2'~are~incompatible
    \str_if_eq:nnF {#3} {??} { ~in~hook~'#3' } .~
    \int_compare:nNnTF {#4} = { 1 }
      { The~ code~ for~ both~ labels~ will~ be~ dropped. }
      { You~ may~ see~ errors~ later. }
  }
  { LaTeX~found~two~incompatible~labels~in~the~same~hook.~
    This~indicates~an~incompatibility~between~packages.  }
\msg_new:nnnn { hooks } { exists }
    { Hook~'#1'~ has~ already~ been~ declared. }
    { There~ already~ exists~ a~ hook~ declaration~ with~ this~
      name.\\
      Please~ use~ a~ different~ name~ for~ your~ hook.}
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{too-many-args}
⟨latexrelease⟩                  {Hooks~with~args}
\msg_new:nnnn { hooks } { too-many-args }
  { Too~many~arguments~for~hook~'#1'. }
  {
    You~tried~to~declare~a~hook~with~#2~arguments,~but~a~
    hook~can~only~have~up~to~nine.~LaTeX~will~define~this~
    hook~with~nine~arguments.
  }
\msg_new:nnnn { hooks } { without-args }
  { Hook~'#1'~has~no~arguments. }
  {
    You~tried~to~use~\iow_char:N\\#2WithArguments~
    on~a~hook~that~takes~no~arguments.\\
    Check~the~usage~of~the~hook~or~use~\iow_char:N\\#2~instead.\\
    \\
    LaTeX~will~use~\iow_char:N\\#2.
  }
\msg_new:nnnn { hooks } { one-time-args }
  { You~can't~have~arguments~in~used~one-time~hook~'#1'. }
  {
    You~tried~to~use~\iow_char:N\\#2WithArguments~
    on~a~one-time~hook~that~has~already~been~used.~
    You~have~to~add~the~code~before~the~hook~is~used,~
    or~add~the~code~without~arguments~using~\iow_char:N\\#2~instead.\\
    \\
    LaTeX~will~use~\iow_char:N\\#2.
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{too-many-args}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\EndIncludeInRelease
\msg_new:nnnn { hooks } { hook-disabled }
  { Cannot~add~code~to~disabled~hook~'#1'. }
  {
    The~hook~'#1'~you~tried~to~add~code~to~was~previously~disabled~
```

```
    with~\iow_char:N\\hook_disable_generic:n~or~
    \iow_char:N\\DisableGenericHook,~so~
    it~cannot~have~code~added~to~it.
  }
\msg_new:nnn { hooks } { empty-label }
  {
    Empty~code~label~\msg_line_context:.~
    Using~'\__hook_currname_or_default:'~instead.
  }
\msg_new:nnn { hooks } { empty-hook }
  {
    Empty~hook~name~\msg_line_context:.
  }
\msg_new:nnn { hooks } { no-default-label }
  {
    Missing~(empty)~default~label~\msg_line_context:. \\
    This~command~was~ignored.
  }
\msg_new:nnnn { hooks } { unknown-rule }
  {
    Unknown~ relationship~ '#3'~
    between~ labels~ '#2'~ and~ '#4'~
    \str_if_eq:nnF {#1} {??} { ~in~hook~'#1' }. ~
    Perhaps~ a~ misspelling?
  }
  {
    The~ relation~ used~ not~ known~ to~ the~ system.~ Allowed~ values~ are~
    'before'~ or~ '<',~
    'after'~ or~ '>',~
    'incompatible-warning',~
    'incompatible-error',~
    'voids'~ or~
    'unrelated'.
  }
\msg_new:nnnn { hooks } { rule-too-late }
  {
    Sorting~rule~for~'#1'~hook~applied~too~late.\\
    Try~setting~this~rule~earlier.
  }
  {
    You~tried~to~set~the~ordering~of~hook~'#1'~using\\
    \ \ \iow_char:N\\DeclareHookRule{#1}{#2}{#3}{#4}\\
    but~hook~'#1'~was~already~used~as~a~one-time~hook,~
    thus~sorting~is\\
    no~longer~possible.~Declare~the~rule~
    before~the~hook~is~used.
  }
\msg_new:nnnn { hooks } { misused-top-level }
  {
    Illegal~use~of~\iow_char:N \\AddToHook{#1}[top-level]{...}.\\
    'top-level'~is~reserved~for~the~user's~document.
```

```
  }
  {
    The~'top-level'~label~is~meant~for~user~code~only,~and~should~only~
    be~used~(sparingly)~in~the~main~document.~Use~the~default~label~
    '\__hook_currname_or_default:'~for~this~\@cls@pkg,~or~another~
    suitable~label.
  }
\msg_new:nnn { hooks } { set-top-level }
  {
    You~cannot~change~the~default~label~#1~'top-level'.~Illegal \\
    \use:nn { ~ } { ~ } \iow_char:N \\#2{#3} \\
    \msg_line_context:.
  }
\msg_new:nnn { hooks } { extra-pop-label }
  {
    Extra~\iow_char:N \\PopDefaultHookLabel. \\
    This~command~will~be~ignored.
  }
\msg_new:nnn { hooks } { missing-pop-label }
  {
    Missing~\iow_char:N \\PopDefaultHookLabel. \\
    The~label~'#1'~was~pushed~but~never~popped.~Something~is~wrong.
  }
\msg_new:nnn { latex2e } { should-not-happen }
  {
    This~should~not~happen.~#1 \\
    Please~report~at~https://github.com/latex3/latex2e.
  }
\msg_new:nnn { hooks } { activate-disabled }
  {
    Cannot~ activate~ hook~ '#1'~ because~ it~ is~ disabled!
  }
\msg_new:nnn { hooks } { cannot-remove }
  {
    Cannot~remove~chunk~'#2'~from~hook~'#1'~because~
    \__hook_if_structure_exist:nTF {#1}
      { it~does~not~exist~in~that~hook. }
      { the~hook~does~not~exist. }
  }
\msg_new:nnn { hooks } { generic-deprecated }
  {
    Generic~hook~'#1/#2/#3'~is~deprecated. \\
    Use~hook~'#1/#3/#2'~instead.
  }
```

4.12 LaTeX 2ε package interface commands

\NewHook
\NewReversedHook
\NewMirroredHookPair

Declaring new hooks ...

```
\NewDocumentCommand \NewHook             { m }
  { \hook_new:n {#1} }
\NewDocumentCommand \NewReversedHook     { m }
```

```
  { \hook_new_reversed:n {#1} }
\NewDocumentCommand \NewMirroredHookPair { mm }
  { \hook_new_pair:nn {#1}{#2} }
```

(*End of definition for* `\NewHook`, `\NewReversedHook`, *and* `\NewMirroredHookPair`. *These functions are documented on page* 194.)

`\NewHookWithArguments` `\NewReversedHookWithArguments` `\NewMirroredHookPairWithArguments`

Declaring new hooks with arguments...

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\NewHookWithArguments}
⟨latexrelease⟩                  {Hooks~with~args}
\NewDocumentCommand \NewHookWithArguments             { mm }
  { \hook_new_with_args:nn {#1} {#2} }
\NewDocumentCommand \NewReversedHookWithArguments     { mm }
  { \hook_new_reversed_with_args:nn {#1} {#2} }
\NewDocumentCommand \NewMirroredHookPairWithArguments { mmm }
  { \hook_new_pair_with_args:nnn {#1} {#2} {#3} }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\NewHookWithArguments}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_new_protected:Npn \NewHookWithArguments #1 #2 { }
⟨latexrelease⟩\cs_new_protected:Npn \NewReversedHookWithArguments #1 #2 { }
⟨latexrelease⟩\cs_new_protected:Npn \NewMirroredHookPairWithArguments #1 #2 #3{}
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\NewHookWithArguments`, `\NewReversedHookWithArguments`, *and* `\NewMirroredHookPairWithArguments`. *These functions are documented on page* 195.)

```
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\ActivateGenericHook}
⟨latexrelease⟩                  {Providing~hooks}
```

`\ActivateGenericHook`

Providing new hooks ...

```
\NewDocumentCommand \ActivateGenericHook { m }
  { \hook_activate_generic:n {#1} }
```

(*End of definition for* `\ActivateGenericHook`. *This function is documented on page* 196.)

`\DisableGenericHook`

Disabling a generic hook.

```
\NewDocumentCommand \DisableGenericHook { m }
  { \hook_disable_generic:n {#1} }
```

(*End of definition for* `\DisableGenericHook`. *This function is documented on page* 195.)

```
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\ActivateGenericHook}
⟨latexrelease⟩                  {Providing~hooks}
⟨latexrelease⟩\def \ActivateGenericHook #1 { }
⟨latexrelease⟩\def \DisableGenericHook #1 { }
⟨latexrelease⟩\EndIncludeInRelease
```

`\AddToHook` `\AddToHookWithArguments`

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\AddToHookWithArguments}
⟨latexrelease⟩                  {Hooks~with~args}
\NewDocumentCommand \AddToHook { m o +m }
  { \hook_gput_code:nnn {#1} {#2} {#3} }
\NewDocumentCommand \AddToHookWithArguments { m o +m }
  { \hook_gput_code_with_args:nnn {#1} {#2} {#3} }
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\AddToHookWithArguments}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_new_protected:Npn \AddToHookWithArguments #1 #2 #3 { }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\AddToHook` *and* `\AddToHookWithArguments`*. These functions are documented on page 197.*)

\AddToHookNext
\AddToHookNextWithArguments

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\AddToHookNextWithArguments}
⟨latexrelease⟩                  {Hooks~with~args}
\NewDocumentCommand \AddToHookNext { m +m }
  { \hook_gput_next_code:nn {#1} {#2} }
\NewDocumentCommand \AddToHookNextWithArguments { m +m }
  { \hook_gput_next_code_with_args:nn {#1} {#2} }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\AddToHookNextWithArguments}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_new_protected:Npn \AddToHookNextWithArguments #1 #2 { }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\AddToHookNext` *and* `\AddToHookNextWithArguments`*. These functions are documented on page 199.*)

\ClearHookNext

```
\NewDocumentCommand \ClearHookNext { m }
  { \hook_gclear_next_code:n {#1} }
```

(*End of definition for* `\ClearHookNext`*. This function is documented on page 199.*)

\RemoveFromHook

```
\NewDocumentCommand \RemoveFromHook { m o }
  { \hook_gremove_code:nn {#1} {#2} }
```

(*End of definition for* `\RemoveFromHook`*. This function is documented on page 198.*)

\SetDefaultHookLabel
\PushDefaultHookLabel
\PopDefaultHookLabel

Now define a wrapper that replaces the top of the stack with the argument, and updates `\g__hook_hook_curr_name_tl` accordingly.

```
\NewDocumentCommand \SetDefaultHookLabel { m }
  { \__hook_set_default_hook_label:n {#1} }
```

The label is only automatically updated with `\@onefilewithoptions` (`\usepackage` and `\documentclass`), but some packages, like Ti*k*Z, define package-like interfaces, like `\usetikzlibrary` that are wrappers around `\input`, so they inherit the default label currently in force (usually `top-level`, but it may change if loaded in another package). To provide a package-like behavior also for hooks in these files, we provide high-level access to the default label stack.

```
\NewDocumentCommand \PushDefaultHookLabel { m }
  { \__hook_curr_name_push:n {#1} }
\NewDocumentCommand \PopDefaultHookLabel { }
  { \__hook_curr_name_pop: }
```

The current label stack holds the labels for all files but the current one (more or less like \@currnamestack), and the current label token list, \g__hook_hook_curr_name_tl, holds the label for the current file. However \@pushfilename happens before \@currname is set, so we need to look ahead to get the \@currname for the label. expl3 also requires the current file in \@pushfilename, so here we abuse \@expl@push@filename@aux@@ to do __hook_curr_name_push:n.

```
\cs_gset_protected:Npn \@expl@push@filename@aux@@ #1#2#3
  {
    \__hook_curr_name_push:n {#3}
    \str_gset:Nx \g_file_curr_name_str {#3}
    #1 #2 {#3}
  }
```

(*End of definition for* \SetDefaultHookLabel, \PushDefaultHookLabel, *and* \PopDefaultHookLabel. *These functions are documented on page* 202.)

\UseHook
\UseOneTimeHook
\UseHookWithArguments
\UseOneTimeHookWithArguments

Avoid the overhead of xparse and its protection that we don't want here (since the hook should vanish without trace if empty)!

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\UseHookWithArguments}
⟨latexrelease⟩                  {Hooks~with~args}
\cs_new:Npn \UseHook        { \hook_use:n }
\cs_new:Npn \UseOneTimeHook { \hook_use_once:n }
\cs_new:Npn \UseHookWithArguments        { \hook_use:nnw }
\cs_new:Npn \UseOneTimeHookWithArguments { \hook_use_once:nnw }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\UseHookWithArguments}
⟨latexrelease⟩                  {Hooks~with~args}
⟨latexrelease⟩\cs_new:Npn \UseHookWithArguments #1 #2 { }
⟨latexrelease⟩\cs_new:Npn \UseOneTimeHookWithArguments #1 #2 { }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* \UseHook *and others. These functions are documented on page* 196.)

\ShowHook
\LogHook

```
\cs_new_protected:Npn \ShowHook { \hook_show:n }
\cs_new_protected:Npn \LogHook { \hook_log:n }
```

(*End of definition for* \ShowHook *and* \LogHook. *These functions are documented on page* 205.)

\DebugHooksOn
\DebugHooksOff

```
\cs_new_protected:Npn \DebugHooksOn  { \hook_debug_on:  }
\cs_new_protected:Npn \DebugHooksOff { \hook_debug_off: }
```

(*End of definition for* \DebugHooksOn *and* \DebugHooksOff. *These functions are documented on page* 206.)

\DeclareHookRule

```
\NewDocumentCommand \DeclareHookRule { m m m m }
                    { \hook_gset_rule:nnnn {#1}{#2}{#3}{#4} }
```

(*End of definition for* \DeclareHookRule. *This function is documented on page* 203.)

\DeclareDefaultHookRule

This declaration is only supported before \begin{document}.

```
\NewDocumentCommand \DeclareDefaultHookRule { m m m }
                    { \hook_gset_rule:nnnn {??}{#1}{#2}{#3} }
\@onlypreamble\DeclareDefaultHookRule
```

(*End of definition for* `\DeclareDefaultHookRule`. *This function is documented on page* 204.)

`\ClearHookRule` A special setup rule that removes an existing relation. Basically @@_rule_gclear:nnn plus fixing the property list for debugging.

FMi: Needs perhaps an L3 interface, or maybe it should get dropped?

```
\NewDocumentCommand \ClearHookRule { m m m }
{ \hook_gset_rule:nnnn {#1}{#2}{unrelated}{#3} }
```

(*End of definition for* `\ClearHookRule`. *This function is documented on page* 203.)

`\IfHookEmptyTF` `\IfHookEmptyT` `\IfHookEmptyF` Here we avoid the overhead of xparse, since `\IfHookEmptyTF` is used in `\end` (that is, every LaTeX environment). As a further optimization, use `\let` rather than `\def` to avoid one expansion step.

```
\cs_new_eq:NN \IfHookEmptyTF \hook_if_empty:nTF
\cs_new_eq:NN \IfHookEmptyT \hook_if_empty:nT
\cs_new_eq:NN \IfHookEmptyF \hook_if_empty:nF
```

(*End of definition for* `\IfHookEmptyTF`, `\IfHookEmptyT`, *and* `\IfHookEmptyF`. *These functions are documented on page* 205.)

`\IfHookExistsTF` Marked for removal and no longer documented in the doc section!

PhO: `\IfHookExistsTF` *is used in* `jlreq.cls`, `pxatbegshi.sty`, `pxeverysel.sty`, `pxeveryshi.sty`, *so the public name may be an alias of the internal conditional for a while. Regardless, those packages' use for* `\IfHookExistsTF` *is not really correct and can be changed.*

```
\cs_new_eq:NN \IfHookExistsTF \__hook_if_usable:nTF
```

(*End of definition for* `\IfHookExistsTF`.)

4.13 Deprecated that needs cleanup at some point

`\hook_disable:n` `\hook_provide:n` `\hook_provide_reversed:n` `\hook_provide_pair:nn` `\__hook_activate_generic_reversed:n` `\__hook_activate_generic_pair:nn` Deprecated.

```
\cs_new_protected:Npn \hook_disable:n
  {
    \__hook_deprecated_warn:nn
      { hook_disable:n }
      { hook_disable_generic:n }
    \hook_disable_generic:n
  }
\cs_new_protected:Npn \hook_provide:n
  {
    \__hook_deprecated_warn:nn
      { hook_provide:n }
      { hook_activate_generic:n }
    \hook_activate_generic:n
  }
\cs_new_protected:Npn \hook_provide_reversed:n
  {
    \__hook_deprecated_warn:nn
      { hook_provide_reversed:n }
      { hook_activate_generic:n }
    \__hook_activate_generic_reversed:n
  }
```

```
\cs_new_protected:Npn \hook_provide_pair:nn
  {
    \__hook_deprecated_warn:nn
      { hook_provide_pair:nn }
      { hook_activate_generic:n }
    \__hook_activate_generic_pair:nn
  }
\cs_new_protected:Npn \__hook_activate_generic_reversed:n #1
  { \__hook_normalize_hook_args:Nn \__hook_activate_generic:nn {#1} { - } }
\cs_new_protected:Npn \__hook_activate_generic_pair:nn #1#2
  { \hook_activate_generic:n {#1} \__hook_activate_generic_reversed:n {#2} }
```

(*End of definition for* `\hook_disable:n` *and others.*)

`\DisableHook`
`\ProvideHook`
`\ProvideReversedHook`
`\ProvideMirroredHookPair`

Deprecated.

```
\cs_new_protected:Npn \DisableHook
  {
    \__hook_deprecated_warn:nn
      { DisableHook }
      { DisableGenericHook }
    \hook_disable_generic:n
  }
\cs_new_protected:Npn \ProvideHook
  {
    \__hook_deprecated_warn:nn
      { ProvideHook }
      { ActivateGenericHook }
    \hook_activate_generic:n
  }
\cs_new_protected:Npn \ProvideReversedHook
  {
    \__hook_deprecated_warn:nn
      { ProvideReversedHook }
      { ActivateGenericHook }
    \__hook_activate_generic_reversed:n
  }
\cs_new_protected:Npn \ProvideMirroredHookPair
  {
    \__hook_deprecated_warn:nn
      { ProvideMirroredHookPair }
      { ActivateGenericHook }
    \__hook_activate_generic_pair:nn
  }
```

(*End of definition for* `\DisableHook` *and others.*)

`\__hook_deprecated_warn:nn`

Warns about a deprecation, telling what should be used instead.

```
\cs_new_protected:Npn \__hook_deprecated_warn:nn #1 #2
  { \msg_warning:nnnn { hooks } { deprecated } {#1} {#2} }
\msg_new:nnn { hooks } { deprecated }
  {
    Command~\iow_char:N\\#1~is~deprecated~and~will~be~removed~in~a~
    future~release. \\ \\
    Use~\iow_char:N\\#2~instead.
  }
```

(*End of definition for* `\__hook_deprecated_warn:nn`.)

4.14 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) LaTeX 2_ε names to allow for internal commands to be used outside this module. We have to unset the `@@` since we want double "at" sign in place of double underscores.

```
⟨@@=⟩
```

`\@expl@@@initialize@all@@`
`\@expl@@@hook@curr@name@pop@@`

```
\cs_new_eq:NN \@expl@@@initialize@all@@
              \__hook_initialize_all:
\cs_new_eq:NN \@expl@@@hook@curr@name@pop@@
              \__hook_curr_name_pop:
```

(*End of definition for* `\@expl@@@initialize@all@@` *and* `\@expl@@@hook@curr@name@pop@@`.)

Rolling back here doesn't undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

```
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{lthooks}
⟨latexrelease⟩                 {The~hook~management}%
⟨latexrelease⟩
⟨latexrelease⟩\def \NewHook#1{}
⟨latexrelease⟩\def \NewReversedHook#1{}
⟨latexrelease⟩\def \NewMirroredHookPair#1#2{}
⟨latexrelease⟩
⟨latexrelease⟩\def \DisableGenericHook #1{}
⟨latexrelease⟩
⟨latexrelease⟩\long\def\AddToHookNext#1#2{}
⟨latexrelease⟩
⟨latexrelease⟩\def\AddToHook#1{\@gobble@AddToHook@args}
⟨latexrelease⟩\providecommand\@gobble@AddToHook@args[2][]{}
⟨latexrelease⟩
⟨latexrelease⟩\def\RemoveFromHook#1{\@gobble@RemoveFromHook@arg}
⟨latexrelease⟩\providecommand\@gobble@RemoveFromHook@arg[1][]{}
⟨latexrelease⟩
⟨latexrelease⟩\def \UseHook        #1{}
⟨latexrelease⟩\def \UseOneTimeHook #1{}
⟨latexrelease⟩\def \ShowHook #1{}
⟨latexrelease⟩\let \DebugHooksOn \@empty
⟨latexrelease⟩\let \DebugHooksOff\@empty
⟨latexrelease⟩
⟨latexrelease⟩\def \DeclareHookRule #1#2#3#4{}
⟨latexrelease⟩\def \DeclareDefaultHookRule #1#2#3{}
⟨latexrelease⟩\def \ClearHookRule #1#2#3{}
```

If the hook management is not provided we make the test for existence false and the test for empty true in the hope that this is most of the time reasonable. If not a package would need to guard against running in an old kernel.

```
⟨latexrelease⟩\long\def \IfHookExistsTF #1#2#3{#3}
⟨latexrelease⟩\long\def \IfHookEmptyTF #1#2#3{#2}
⟨latexrelease⟩
⟨latexrelease⟩\EndModuleRelease
```

```
⟨@@=hook⟩

⟨latexrelease⟩\cs:w __hook_rollback_tidying: \cs_end:
⟨latexrelease⟩\bool_lazy_and:nnT
⟨latexrelease⟩    { \int_compare_p:nNn { \sourceLaTeXdate } > { 20230600 } }
⟨latexrelease⟩    { \int_compare_p:nNn { \requestedLaTeXdate } < { 20230601 } }
⟨latexrelease⟩  {
⟨latexrelease⟩    \cs_gset_protected:Npn \__hook_rollback_tidying:
⟨latexrelease⟩      {
⟨latexrelease⟩        \@latex@error { Rollback~code~executed~twice }
⟨latexrelease⟩          {
⟨latexrelease⟩            Something~went~wrong~(unless~this~was~
⟨latexrelease⟩            done~on~purpose~in~a~testing~environment).
⟨latexrelease⟩          }
⟨latexrelease⟩        \use_none:nnnn
⟨latexrelease⟩      }
⟨latexrelease⟩    \cs_set:Npn \__hook_tmp:w #1 #2
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_tl_gset:cx { __hook#1~#2 }
⟨latexrelease⟩          {
⟨latexrelease⟩            \exp_args:No \exp_not:o
⟨latexrelease⟩              {
⟨latexrelease⟩                \cs:w __hook#1~#2 \exp_last_unbraced:Ne \cs_end:
⟨latexrelease⟩                  { \__hook_braced_cs_parameter:n
⟨latexrelease⟩                      { __hook#1~#2 } }
⟨latexrelease⟩              }
⟨latexrelease⟩          }
⟨latexrelease⟩      }
⟨latexrelease⟩    \seq_map_inline:Nn \g__hook_all_seq
⟨latexrelease⟩      {
⟨latexrelease⟩        \exp_after:wN \cs_gset_nopar:Npn
⟨latexrelease⟩          \cs:w g__hook_#1_code_prop \exp_args:NNo \exp_args:No
⟨latexrelease⟩            \cs_end: { \cs:w g__hook_#1_code_prop \cs_end: }
⟨latexrelease⟩        \__hook_tmp:w { _toplevel } {#1}
⟨latexrelease⟩        \__hook_tmp:w { _next } {#1}
⟨latexrelease⟩      }
⟨latexrelease⟩  }
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩

⟨@@=⟩
```

File 09
ltcmdhooks.dtx

1 Introduction

This file implements generic hooks for (arbitrary) commands. In theory every command \⟨*name*⟩ offers now two associated hooks to which code can be added using `\AddToHook`,[13] `\AddToHookNext`, `\AddToHookWithArguments`, and `\AddToHookNextWithArguments`.[14]

However, this is only true "in theory". In practice there are a number of restrictions that makes it impossible to use such generic command hooks in a number of cases, so please read all of section 2 to understand what may prevent you from using them successfully.

The generic command hooks are:

cmd/⟨*name*⟩/before This hook is executed at the very start of the command, right after its arguments (if any) are parsed. The hook ⟨*code*⟩ runs in the command inside a call to `\UseHookWithArguments`. Any code added to this hook using `\AddToHookWithArguments` or `\AddToHookNextWithArguments` can access the command's arguments using `#1`, `#2`, etc., up to the number of arguments of the command. If `\AddToHook` or `\AddToHookNext` are used, the arguments cannot be accessed (see the lthooks documentation[15] on hooks with arguments).

cmd/⟨*name*⟩/after This hook is similar to `cmd/⟨name⟩/before`, but it is executed at the very end of the command body. This hook is implemented as a reversed hook.

The hooks are not physically present before `\begin{document}`[16] (i.e., using a command in the preamble will never execute the hook) and if nobody has declared any code for them, then they are not added to the command code ever. For example, if we have the following definition

```
\newcommand\foo[2]{Code #1 for #2!}
```

then executing `\foo{A}{B}` will simply run `Code␣A␣for␣B!` as it was always the case. However, if somebody, somewhere (e.g., in a package) adds

```
\AddToHook{cmd/foo/before}{<before code>}
```

then, after `\begin{document}` the definition of `\foo` will be:

```
\renewcommand\foo[2]{%
  \UseHookWithArguments{cmd/foo/before}{2}{#1}{#2}%
  Code #1 for #2!}
```

and similarly `\AddToHook{cmd/foo/after}{<after code>}` alters the definition to

[13] In this documentation, when something is being said about `\AddToHook`, the same will be valid for `\AddToHookWithArguments`, unless that particular paragraph is highlighting the differences between both. The same is true for the other hook-related functions and their `...WithArguments` counterparts.

[14] In practice this is not supported for all types of commands, see section 2.2 for the restrictions that apply and what happens if one tries to use this with commands for which this is not supported.

[15] `texdoc lthooks-doc`

[16] More specifically, they are inserted in the commands after the `begindocument` hook, so they are also not present while LaTeX is reading the `.aux` file.

```
\renewcommand\foo[2]{%
  Code #1 for #2!%
  \UseHookWithArguments{cmd/foo/after}{2}{#1}{#2}}
```

In other words, the mechanism is similar to what etoolbox offers with `\pretocmd` and `\apptocmd` with the important differences

- that code can be prepended or appended (i.e., added to the hooks) even if the command itself is not (yet) defined, because the defining package has not been loaded at this point;
- and that by using the hook management interface it is now possible to define how the code chunks added in these places are ordered, if different packages want to add code at these points.

2 Restrictions and Operational details

Adding arbitrary material to commands is tricky because most of the time we do not know what the macro expects as arguments when expanding and TEX doesn't have a reliable way to see that, so some guesswork has to be employed.

We can do this in most cases when commands are defined using `\NewDocumentCommand` or `\newcommand` (with a few exceptions). For commands defined with `\def` the situation is less good. Common cases where the command hooks will not work are:

- Commands that use special catcode settings within their definition. In that case it is usually not possible to augment the definition (see 2.1).
- If a command is defined while `\ExplSyntaxOn` is in force **and** the command contains `~` characters to represent spaces, then it can't be patched to include the command hooks. In fact in some very special circumstances you might even get a low-level error rather than the information that the command can't be patched (see, for example, https://github.com/latex3/latex2e/issues/1430.
- Commands that have arguments as far as the user is concerned (e.g., `\section` or `\caption`), but are defined in a way that these arguments are not read by the user level command but only later during the processing. In that case the `after` hook doesn't work at all. The before hook only works with `\AddToHook` but not with `\AddToHookWithArguments` because the arguments haven't been read at that point where the hook is patched in. See section 2.2.
- Adding a specific generic command hook is only attempted once per command, thus after redefining a command such hooks will no longer be there and will also not being re-added, see section 2.1.1.

All this means that you have to have a good understanding of how commands are defined when you attempt to make use of such hooks and something goes wrong. What can help in that case is to turn on `\DebugHooksOn` in which case you get much more (low-level) details on why something fails and what was tried to enable the hooks.

2.1 Patching

The code here tries to find out if a command was defined with `\newcommand` or `\DeclareRobustCommand` or `\NewDocumentCommand`, and if so it *assumes* that the argument specification of the command is as expected (which is not fail-proof, if someone redefines the internals of these commands in devious ways, but is a reasonable assumption).

If the command is one of the defined types, the code here does a sandboxed expansion of the command such that it can be redefined again exactly as before, but with the hook code added.

If however the command is not a known type (it was defined with `\def`, for example), then the code uses an approach similar to etoolbox's `\patchcmd` to retokenize the command with the hook code in place. This procedure, however, is more likely to fail if the catcode settings are not the same as the ones at the time of command's definition, so not always adding a hook to a command will work.

2.1.1 Timing

When `\AddToHook` (or its expl3 equivalent) is called with a generic `cmd` hook, say, `cmd/foo/before`, for the first time (that is, no code was added to that same hook before), in the preamble of a document, it will store a patch instruction for that command until `\begin{document}`, and only then all the commands which had hooks added will be patched in one go. That means that no command in the preamble will have hooks patched into them.

At `\begin{document}` all the delayed patches will be executed, and if the command doesn't exist the code is still added to the hook, but it will not be executed. After `\begin{document}`, when `\AddToHook` is called with a generic `cmd` hook the first time, the command will be immediately patched to include the hook, and if it doesn't exist or if it can't be patched for any reason, an error is thrown; if `\AddToHook` was already used in the preamble no new patching is attempted.

This has the consequence that a command defined or redefined after `\begin{document}` only uses generic `cmd` hook code if `\AddToHook` is called for the first time after the definition is made, or if the command explicitly uses the generic hook in its definition by declaring it with `\NewHookPair` adding `\UseHook` as part of the code.[17]

2.2 Commands that look ahead

Some commands are defined in different "steps" and they look ahead in the input stream to find more arguments. If you try to add some code to the `cmd/`⟨*name*⟩`/after` hook of such command, it will not work, and it is not possible to detect that programmatically, so the user has to know (or find out) which commands can or cannot have hooks attached to them.

One good example is the `\section` command. You can add something to the `cmd/section/before` hook (but only with `\AddToHook` not `\AddToHookWithArguments`), but if you try to add anything to the `cmd/section/after` hook, `\section` will no longer work at all. That happens because the `\section` macro takes no argument, but instead calls a few internal LaTeX macros to look for the optional and mandatory arguments. By adding code to the `cmd/section/after` hook, you get in the way of that scanning.

[17] We might change this behavior in the main document slightly after gaining some usage experience.

In such a case, where it is known that a specific generic command hook does not work if code is added to it, the package author can add a `\DisableGenericHook`[18] declaration to prevent this from happening in user documents and thereby avoiding obscure errors.

3 Package Author Interface

The `cmd` hooks are, by default, available for all commands that can be patched to add the hooks. For some commands, however, the very beginning or the very end of the code is not the best place to put the hooks, for example, if the command looks ahead for arguments (see section 2.2).

If you are a package author and you want to add the hooks to your own commands in the proper position you can define the command and manually add the `\UseHookWithArguments` calls inside the command in the proper positions, and manually define the hooks with `\NewHookWithArguments` or `\NewReversedHookWithArguments`. When the hooks are explicitly defined, patching is not attempted so you can make sure your command works properly. For example, an (admittedly not really useful) command that typesets its contents in a framed box with width optionally given in parentheses:

```
\newcommand\fancybox{\@ifnextchar({\@fancybox}{\@fancybox(5cm)}}
\def\@fancybox(#1)#2{\fbox{\parbox{#1}{#2}}}
```

If you try that definition, then add some code after it with

```
\AddToHook{cmd/fancybox/after}{<code>}
```

and then use the `\fancybox` command you will see that it will be completely broken, because the hook will get executed in the middle of parsing for optional `(...)` argument.

If, on the other hand, you want to add hooks to your command you can do something like:

```
\newcommand\fancybox{\@ifnextchar({\@fancybox}{\@fancybox(5cm)}}
\def\@fancybox(#1)#2{\fbox{%
              \UseHookWithArguments{cmd/fancybox/before}{2}{#1}{#2}%
              \parbox{#1}{#2}%
              \UseHookWithArguments{cmd/fancybox/after}{2}{#1}{#2}}}
\NewHookWithArguments{cmd/fancybox/before}{2}
\NewReversedHookWithArguments{cmd/fancybox/after}{2}
```

then the hooks will be executed where they should and no patching will be attempted. It is important that the hooks are declared with `\NewHookWithArguments` or `\NewReversedHookWithArguments`, otherwise the command hook code will try to patch the command. Note also that the call to `\UseHookWithArguments{cmd/fancybox/before}` does not need to be in the definition of `\fancybox`, but anywhere it makes sense to insert it (in this case in the internal `\@fancybox`).

Alternatively, if for whatever reason your command does not support the generic hooks provided here, you can disable a hook with `\DisableGenericHook`[19], so that when someone tries to add code to it they will get an error. Or if you don't want the error, you can simply declare the hook with `\NewHook` and never use it.

[18] Please use `\DisableGenericHook` if at all, only on hooks that you "own", i.e., for commands your package or class defines and not second guess whether or not hooks of other packages should get disabled!

[19] Please use `\DisableGenericHook` if at all, only on hooks that you "own", i.e., for commands your package or class defines and not second guess whether or not hooks of other packages should get disabled!

The above approach is useful for really complex commands where for one or the other reason the hooks can't be placed at the very beginning and end of the command body and some hand-crafting is needed. However, in the example above the real (and in fact only) issue is the cascading argument parsing in the style developed long ago in LaTeX 2.09. Thus, a much simpler solution for this case is to replace it with the modern `\NewDocumentCommand` syntax and define the command as follows:

```
\DeclareDocumentCommand\fancybox{D(){5cm}m}{\fbox{\parbox{#1}{#2}}}
```

If you do that then both hooks automatically work and are patched into the right places.

3.1 Arguments and redefining commands

The code in ltcmdhooks does its best to find out how many arguments a given command has, and to insert the appropriate call to `\UseHookWithArguments`, so that the arguments seen by the hook are exactly those grabbed by the command (the hook, after all, is a macro call, so the arguments have to be placed in the right order, or they won't match).

When using the package writer interface, as discussed in section 3, to change the position of the hooks in your commands, you are also free to change how the hook code in your command sees its arguments. When a `cmd` hook is declared with `\NewHook` (or `\NewHookWithArguments` or other variations of that), it loses its "generic" nature and works as a regular hook. This means that you may choose to declare it without arguments regardless if the command takes arguments or not, or declare it with arguments, even if the command takes none.

However, this flexibility should not be abused. When using a nonstandard configuration for the hook arguments, think reasonably: a user will expect that the argument `#1` in the hook corresponds to the argument's first argument, and so on. Any other configuration is likely to cause confusion and, if used, will have to be well documented.

This flexibility, however, allows you to "correct" the arguments for the hooks. For example, LaTeX's `\refstepcounter` has a single argument, the name of the counter. The cleveref package adds an optional argument to `\refstepcounter`, making the name of the counter argument `#2`. If the author of cleveref wanted, for whatever reason, to add hooks to `\refstepcounter`, to preserve compatibility he could write something along the lines of:

```
\NewHookWithArguments{cmd/refstepcounter/before}{1}
\renewcommand\refstepcounter[2][<default>]{%
  \UseHookWithArguments{cmd/refstepcounter/before}{1}{#2}%
  <code for \refstepcounter>}
```

so that the mandatory argument, which is arg `#2` in the definition, would still be seen as `#1` in the hook code.

Another possibility would be to place the optional argument as the second argument for the hook, so that people looking for it would be able to use it. In either case, it would have to be well documented to cause as little confusion as possible.

4 The Implementation

4.1 Execution plan

To add `before` and `after` hooks to a command we will need to peek into the definition of a command, which is always a tricky thing to do. Some cases are easy because we

know how the command was defined, so we can assume how its ⟨*parameter text*⟩ looks like (for example a command defined with `\newcommand` may have an optional argument followed by a run of mandatory arguments), so we can just expand that command and make it grab `#1`, `#2`, etc. as arguments and define it all back with the hooks added.

Life's usually not that easy, so with some commands we can't do that (a `#1` might as well be $\#_{12}1_{12}$ instead of the expected $\#_{6}1_{12}$, for example) so we need to resort to "patching" the command: read its `\meaning`, and tokenize it again with `\scantokens` and hope for the best.

So the overall plan is:

1. Check if a command is of a known type (that is, defined with `\newcommand`[20], `\DeclareRobustCommand`, or `\New(Expandable)DocumentCommand`), and if is, take appropriate action.

2. If the command is not a known type, we'll check if the command can be patched. Two things will prevent a command from being patched: if it was defined in a nonstandard catcode setting, or if it is an internal expl3 command with `__`⟨*module*⟩ in its name, in which case we refuse to patch.

3. If the command was defined in nonstandard catcode settings, we will try a few standard ones to try our best to carry out the pathing. If this doesn't help either, the code will give up and throw an error.

```
⟨@@=hook⟩
```

```
⟨*2ekernel | latexrelease⟩
\ExplSyntaxOn
⟨latexrelease⟩\NewModuleRelease{2021/06/01}{ltcmdhooks}
⟨latexrelease⟩                    {The~hook~management~system~for~commands}
```

4.2 Variables

`\g_hook_patch_action_list_tl`

Pairs of `\if<cmd>..\patch<cmd>` to be used with `\robust@command@act` when looking for a known patching rule. This token list is exposed because we see some future applications (with very specialized packages, such as etoolbox that may want to extend the pairs processed. It is not meant for general use which is why it is not documented in the interface documentation above.

```
\tl_new:N \g_hook_patch_action_list_tl
```

(*End of definition for* `\g_hook_patch_action_list_tl`.)

`\l__hook_patch_num_args_int`

The number of arguments in a macro being patched.

```
\int_new:N \l__hook_patch_num_args_int
```

(*End of definition for* `\l__hook_patch_num_args_int`.)

`\l__hook_patch_prefixes_tl`
`\l__hook_param_text_tl`
`\l__hook_replace_text_tl`

The prefixes and parameters of the definition for the macro being patched.

```
\tl_new:N \l__hook_patch_prefixes_tl
\tl_new:N \l__hook_param_text_tl
\tl_new:N \l__hook_replace_text_tl
```

[20] It's not always possible to reliably detect this case because a command defined with no optional argument is indistinguishable from a `\def`ed command.

(*End of definition for* `\l__hook_patch_prefixes_tl`, `\l__hook_param_text_tl`, *and* `\l__hook_replace_text_tl`.)

`\c__hook_hash_tl`
`\c__hook_hashes_tl`

Two constant token lists that contain one and two parameter tokens.

```
\tl_const:Nn \c__hook_hash_tl { # }
\tl_const:Nn \c__hook_hashes_tl { # # }
```

(*End of definition for* `\c__hook_hash_tl` *and* `\c__hook_hashes_tl`.)

`\__hook_exp_not:NN`
`\__hook_def_cmd:w`

Two temporary macros that change depending on the macro being patched.

```
\cs_new_eq:NN \__hook_exp_not:NN ?
\cs_new_eq:NN \__hook_def_cmd:w ?
```

(*End of definition for* `\__hook_exp_not:NN` *and* `\__hook_def_cmd:w`.)

`\q__hook_recursion_tail`
`\q__hook_recursion_stop`

Internal quarks for recursion: they can't appear in any macro being patched.

```
\quark_new:N \q__hook_recursion_tail
\quark_new:N \q__hook_recursion_stop
```

(*End of definition for* `\q__hook_recursion_tail` *and* `\q__hook_recursion_stop`.)

`\g__hook_delayed_patches_prop`

A list containing the patches delayed to `\begin{document}`, so that patching is not attempted twice.

```
\prop_new:N \g__hook_delayed_patches_prop
```

(*End of definition for* `\g__hook_delayed_patches_prop`.)

`\__hook_patch_debug:x`

A helper for patching debug info.

```
\cs_new_protected:Npn \__hook_patch_debug:x #1
  { \__hook_debug:n { \iow_term:x { [lthooks]~#1 } } }
```

(*End of definition for* `\__hook_patch_debug:x`.)

4.3 Variants

`\tl_rescan:nV`

expl3 function variants used throughout the code.

```
\cs_generate_variant:Nn \tl_rescan:nn { nV }
```

(*End of definition for* `\tl_rescan:nV`.)

4.4 Patching or delaying

Before `\begin{document}` all patching is delayed.

`\__hook_try_put_cmd_hook:n`
`\__hook_try_put_cmd_hook:w`

This function is called from within `\AddToHook`, when code is first added to a generic `cmd` hook. If it is called within in the preamble, it delays the action until `\begin{document}`; otherwise it tries to update the hook.

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__hook_try_put_cmd_hook:n}%
⟨latexrelease⟩                  {Standardise~generic~hook~names}
\cs_new_protected:Npn \__hook_try_put_cmd_hook:n #1
  { \__hook_try_put_cmd_hook:w #1 / / / \s__hook_mark {#1} }
\cs_new_protected:Npn \__hook_try_put_cmd_hook:w
    #1 / #2 / #3 / #4 \s__hook_mark #5
  {
    \__hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'#2'~(#3): } }
```

```
    \exp_args:Nc \__hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3}
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\__hook_try_put_cmd_hook:n}%
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\cs_new_protected:Npn \__hook_try_put_cmd_hook:n #1
⟨latexrelease⟩  { \__hook_try_put_cmd_hook:w #1 / / / \s__hook_mark {#1} }
⟨latexrelease⟩\cs_new_protected:Npn \__hook_try_put_cmd_hook:w
⟨latexrelease⟩    #1 / #2 / #3 / #4 \s__hook_mark #5
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'#2'~(#3): } }
⟨latexrelease⟩    \str_case:nnTF {#3}
⟨latexrelease⟩        { { before } { } { after } { } }
⟨latexrelease⟩      { \exp_args:Nc \__hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3} }
⟨latexrelease⟩      { \msg_error:nnnn { hooks } { wrong-cmd-hook } {#2} {#3} }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_try_put_cmd_hook:n` *and* `\__hook_try_put_cmd_hook:w`.)

`\__hook_patch_cmd_or_delay:Nnn`
`\__hook_cmd_begindocument_code:`

In the preamble, `\__hook_patch_cmd_or_delay:Nnn` just adds the patch instruction to a property list to be executed later.

```
\cs_new_protected:Npn \__hook_patch_cmd_or_delay:Nnn #1 #2 #3
  {
    \__hook_debug:n { \iow_term:n { ->~Add~generic~cmd~hook~for~#2~(#3). } }
    \__hook_debug:n
      { \iow_term:n { !~In~the~preamble:~delaying. } }
    \prop_gput:Nnn \g__hook_delayed_patches_prop { #2 / #3 }
      { \__hook_cmd_try_patch:nn {#2} {#3} }
  }
```

The delayed patches are added to a property list to prevent duplication, and the code stored in the property list for each key is executed. The function `\__hook_patch_cmd_or_delay:Nnn` is also redefined to be `\__hook_patch_command:Nnn` so that no further delaying is attempted.

```
\cs_new_protected:Npn \__hook_cmd_begindocument_code:
  {
    \cs_gset_eq:NN \__hook_patch_cmd_or_delay:Nnn \__hook_patch_command:Nnn
    \prop_map_function:NN \g__hook_delayed_patches_prop { \use_ii:nn }
    \prop_gclear:N \g__hook_delayed_patches_prop
    \cs_undefine:N \__hook_cmd_begindocument_code:
  }
\g@addto@macro \@kernel@after@begindocument
  { \__hook_cmd_begindocument_code: }
```

(*End of definition for* `\__hook_patch_cmd_or_delay:Nnn` *and* `\__hook_cmd_begindocument_code:`.)

`\__hook_cmd_try_patch:nn`

At `\begin{document}` tries patching the command if the hook was not manually created in the meantime. If the document does not exist, no error is raised here as it may hook into a package that wasn't loaded. Hooks added to commands in the document body still raise an error if the command is not defined.

```
\cs_new_protected:Npn \__hook_cmd_try_patch:nn #1 #2
  {
    \__hook_debug:n
```

```
      { \iow_term:x { ->~\string\begin{document}~try~cmd / #1 / #2. } }
    \__hook_if_declared:nTF { cmd / #1 / #2 }
      {
        \__hook_debug:n
          { \iow_term:n { .->~Giving~up:~hook~already~created. } }
      }
      {
        \cs_if_exist:cT {#1}
          { \exp_args:Nc \__hook_patch_command:Nnn {#1} {#1} {#2} }
      }
  }
```

(*End of definition for* `\__hook_cmd_try_patch:nn`.)

4.5 Patching commands

`\__hook_patch_command:Nnn`
`\__hook_patch_check:NNnn`
`\__hook_if_public_command:NTF`
`\__hook_if_public_command:w`

`\__hook_patch_command:Nnn` will do some sanity checks on the argument to detect if it is possible to add hooks to the command, and raises an error otherwise. If the command can contain hooks, then it uses `\robust@command@act` to find out what type is the command, and patch it accordingly.

```
\cs_new_protected:Npn \__hook_patch_command:Nnn #1 #2 #3
  {
    \__hook_patch_debug:x { analyzing~'\token_to_str:N #1' }
    \__hook_patch_debug:x { \token_to_str:N #1 = \token_to_meaning:N #1 }
    \__hook_patch_check:NNnn \cs_if_exist:NTF #1 { undef }
      {
        \__hook_patch_debug:x { ++~control~sequence~is~defined }
        \__hook_patch_check:NNnn \token_if_macro:NTF #1 { macro }
          {
            \__hook_patch_debug:x { ++~control~sequence~is~a~macro }
            \__hook_patch_check:NNnn \__hook_if_public_command:NTF #1 { expl3 }
              {
                \__hook_patch_debug:x { ++~macro~is~not~private }
                \robust@command@act
                  \g_hook_patch_action_list_tl #1
                  \__hook_retokenize_patch:Nnn { #1 {#2} {#3} }
              }
          }
      }
  }
```

And here's the auxiliary used above:

```
\cs_new_protected:Npn \__hook_patch_check:NNnn #1 #2 #3 #4
  {
    #1 #2 {#4}
      {
        \msg_error:nnxx { hooks } { cant-patch }
          { \token_to_str:N #2 } {#3}
      }
  }
```

and a conditional `\__hook_if_public_command:NTF` to check if a command has `__` in its name (no other checking is performed). Primitives with `:D` in their name could be included here, but they are already discarded in the `\token_if_macro:NTF` test above.

```
\use:x
  {
    \prg_new_protected_conditional:Npnn
        \exp_not:N \__hook_if_public_command:N ##1 { TF }
      {
        \exp_not:N \exp_last_unbraced:Nf
          \exp_not:N \__hook_if_public_command:w
            { \exp_not:N \cs_to_str:N ##1 }
          \tl_to_str:n { _ _ } \s__hook_mark
      }
  }
\exp_last_unbraced:NNNNo
\cs_new_protected:Npn \__hook_if_public_command:w
    #1 \tl_to_str:n { _ _ } #2 \s__hook_mark
  {
    \tl_if_empty:nTF {#2}
      { \prg_return_true: }
      { \prg_return_false: }
  }
```

(*End of definition for* `\__hook_patch_command:Nnn` *and others.*)

4.5.1 Patching by expansion and redefinition

`\g_hook_patch_action_list_tl`

This is the list of known command types and the function that patches the command hooks into them. The conditionals are taken from `\ShowCommand`, `\NewCommandCopy` and `\__kernel_cmd_if_xparse:NTF` defined in `ltcmd`.

```
\tl_gset:Nn \g_hook_patch_action_list_tl
  {
    { \@if@DeclareRobustCommand \__hook_patch_DeclareRobustCommand:Nnn }
    { \@if@newcommand \__hook_patch_newcommand:Nnn }
    { \__kernel_cmd_if_xparse:NTF \__hook_cmd_patch_xparse:Nnn }
  }
```

(*End of definition for* `\g_hook_patch_action_list_tl`.)

`\__hook_patch_DeclareRobustCommand:Nnn`

At this point we know that the commands can be patched by expanding then redefining. These are the cases of commands defined with `\newcommand` with an optional argument or with `\DeclareRobustCommand`.

With `\__hook_patch_DeclareRobustCommand:Nnn` we check if the command has an optional argument (with a test counter-intuitively called `\@if@newcommand`; also make sure the command doesn't take args by calling `\robust@command@chk@safe`). If so, we pass the patching action to `\__hook_patch_newcommand:Nnn`, otherwise we call the patching engine `\__hook_patch_expand_redefine:NNnn` with a `\c_false_bool` to indicate that there is no optional argument.

```
\cs_new_protected:Npn \__hook_patch_DeclareRobustCommand:Nnn #1
  {
    \exp_args:Nc \__hook_patch_DeclareRobustCommand_aux:Nnn
      { \cs_to_str:N #1 ~ }
  }
\cs_new_protected:Npn \__hook_patch_DeclareRobustCommand_aux:Nnn #1
  {
    \robust@command@chk@safe #1
```

```
      { \@if@newcommand #1 }
      { \use_ii:nn }
        { \__hook_patch_newcommand:Nnn }
        { \__hook_patch_expand_redefine:NNnn \c_false_bool }
          #1
  }
```

(*End of definition for* `\__hook_patch_DeclareRobustCommand:Nnn`.)

`\__hook_patch_newcommand:Nnn` If the command was defined with `\newcommand` and an optional argument, call the patching engine with a `\c_true_bool` to flag the presence of an optional argument, and with `\\command` to patch the actual code for `\command`.

```
\cs_new_protected:Npn \__hook_patch_newcommand:Nnn #1
  {
    \exp_args:NNc \__hook_patch_expand_redefine:NNnn \c_true_bool
      { \c_backslash_str \cs_to_str:N #1 }
  }
```

(*End of definition for* `\__hook_patch_newcommand:Nnn`.)

`\__hook_cmd_patch_xparse:Nnn` And for commands defined by the xparse commands use this for patching:

```
\cs_new_protected:Npn \__hook_cmd_patch_xparse:Nnn #1
  {
    \exp_args:NNc \__hook_patch_expand_redefine:NNnn \c_false_bool
      { \cs_to_str:N #1 ~ code }
  }
```

(*End of definition for* `\__hook_cmd_patch_xparse:Nnn`.)

`\__hook_patch_expand_redefine:NNnn`
`\__hook_redefine_with_hooks:Nnnn`
`\__hook_make_prefixes:w`

Now the real action begins. Here we have in `#1` a boolean indicating if the command has a leading `[...]`-delimited argument, in `#2` the command control sequence, in `#3` the name of the command (note that `#1` ≠ `\csname#2\endcsname` at this point!), and in `#4` the hook position, either `before` or `after`.

Patching with expansion+redefinition is trickier than it looks like at first glance. Suppose the simple definition:

```
\def\foo#1{#1##2}
```

When defined, its ⟨*replacement text*⟩ will be a token list containing:

out_param `1`, *mac_param* `#`, *character* `2`

Then, after expanding `\foo{##1}` (here `##` denotes a single $\#_6$) we end up with a token list with *out_param* `1` replaced:

mac_param `#`, *character* `1`, *mac_param* `#`, *character* `2`

that is, the definition would be:

```
\def\foo#1{#1#2}
```

which obviously fails, because the original input in the definition was `##` but TEX reduced that to a single parameter token $\#_6$ when carrying out the definition. That leaves no room for a clever solution with (say) `\unexpanded`, because anything that would double the second $\#_6$, would also (incorrectly) double the first, so there's not much to do other than a manual solution.

There are three cases we can distinguish to make things hopefully faster on simpler cases:

1. a macro with no parameters;
2. a macro with no parameter tokens in its definition;
3. a macro with parameters *and* parameter tokens.

The first case is trivial: if the macro has no parameters, we can just use `\unexpanded` around it, and if there is a parameter token in it, it is handled correctly (the macro can be treated as a `tl` variable).

The second case requires looking at the ⟨*replacement text*⟩ of the macro to see if it has a parameter token in there. If it does not, then there is no worry, and the macro can be redefined normally (without `\unexpanded`).

The third case, as usual, is the devious one. Here we'll have to loop through the definition token by token, and double every parameter token, so that this case can be handled like the previous one.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_patch_expand_redefine:NNnn}
⟨latexrelease⟩                  {cmd~hooks~with~args}
\cs_new_protected:Npn \__hook_patch_expand_redefine:NNnn #1 #2 #3 #4
  {
    \__hook_patch_debug:x { ++~command~can~be~patched~without~rescanning }
```

We'll start by counting the number of arguments in the command by counting the number of characters in the `\cs_parameter_spec:N` of the macro, divided by two, and subtracting one if the command has an optional argument (that is, an extra `[]` in its ⟨*parameter text*⟩).

```
    \int_set:Nn \l__hook_patch_num_args_int
      {
        \exp_args:Nf \str_count:n { \__kernel_cs_parameter_spec:N #2 } / 2
        \bool_if:NT #1 { -1 }
      }
```

Now build two token lists:

`\l__hook_param_text_tl` will contain the ⟨*parameter text*⟩ to be used when redefining the macro. It should be identical to the ⟨*parameter text*⟩ used when originally defining that macro.

`\l__hook_replace_text_tl` will contain braced pairs of `\c__hook_hashes_tl`⟨*num*⟩ to feed to the macro when expanded. This token list as well as the previous will have the first item surrounded by `[...]` in the case of an optional argument.

The use of `\c__hook_hashes_tl` here is to differentiate actual parameters in the macro from parameter tokens in the original definition of the macro. Later on, `\c__hook_hashes_tl` is either replaced by actual parameter tokens, or expanded into them.

```
    \int_compare:nNnTF { \l__hook_patch_num_args_int } > { \c_zero_int }
      {
```

We'll first check if the command has any parameter token in its definition (feeding it empty arguments), and set `\__hook_exp_not:n` accordingly. `\__hook_exp_not:n` will be used later to either leave `\c__hook_hashes_tl` or expand it, and also to remember the result of `\__hook_if_has_hash:nTF` to avoid testing twice (the test can be rather slow).

```
        \tl_set:Nx \l__hook_tmpa_tl { \bool_if:NTF #1 { [ ] } { { } } }
        \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
```

```
          { \tl_put_right:Nn \l__hook_tmpa_tl { { } } }
        \exp_args:NNo \exp_args:No \__hook_if_has_hash:nTF
            { \exp_after:wN #2 \l__hook_tmpa_tl }
          { \cs_set_eq:NN \__hook_exp_not:n \exp_not:n }
          { \cs_set_eq:NN \__hook_exp_not:n \use:n }
        \cs_set_protected:Npn \__hook_tmp:w ##1 ##2
          {
            ##1 \l__hook_param_text_tl   { \use:n ##2 }
            ##1 \l__hook_replace_text_tl { \__hook_exp_not:n {##2} }
          }
```

Here we'll conditionally add `[...]` around the first parameter:

```
        \bool_if:NTF #1
          { \__hook_tmp:w \tl_set:Nx { [ \c__hook_hashes_tl 1 ] } }
          { \__hook_tmp:w \tl_set:Nx { { \c__hook_hashes_tl 1 } } }
```

Then, for every parameter from the second, just add it normally:

```
        \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
          { \__hook_tmp:w \tl_put_right:Nx { { \c__hook_hashes_tl ##1 } } }
```

Now, if the command has any parameter token in its definition (then `\__hook_exp_not:n` is `\exp_not:n`), call `\__hook_double_hashes:n` to double them, and replace every `\c__hook_hashes_tl` by `#`:

```
        \tl_set:Nx \l__hook_replace_text_tl
          { \exp_not:N #2 \exp_not:V \l__hook_replace_text_tl }
        \tl_set:Nx \l__hook_replace_text_tl
          {
            \token_if_eq_meaning:NNTF \__hook_exp_not:n \exp_not:n
              { \exp_args:NNV \exp_args:No \__hook_double_hashes:n }
              { \exp_args:NV \exp_not:o }
                  \l__hook_replace_text_tl
          }
```

And now, set a few auxiliaries for the case that the macro has parameters, so it won't be passed through `\unexpanded` (twice):

```
        \cs_set_eq:NN \__hook_def_cmd:w \tex_gdef:D
        \cs_set_eq:NN \__hook_exp_not:NN \prg_do_nothing:
      }
      {
```

In the case the macro has no parameters, we'll treat it as a token list and things are much simpler (expansion control looks a bit complicated, but it's just a pair of `\exp_not:N` preventing another `\exp_not:n` from expanding):

```
        \tl_clear:N \l__hook_param_text_tl
        \tl_set_eq:NN \l__hook_replace_text_tl #2
        \cs_set_eq:NN \__hook_def_cmd:w \tex_xdef:D
        \cs_set:Npn \__hook_exp_not:NN ##1 { \exp_not:N ##1 \exp_not:N }
      }
```

Before redefining, we need to also get the prefixes used when defining the command. Here we ensure that the `\escapechar` is printable, otherwise a macro defined with prefixes `\protected \long` will have it `\meaning` printed as `protectedlong`, making life unnecessarily complicated. Here the `\escapechar` is changed to `/`, then we loop between pairs of `/.../` extracting the prefixes.

```
    \group_begin:
      \int_set:Nn \tex_escapechar:D { `\/ }
```

```
      \use:x
        {
    \group_end:
    \tl_set:Nx \exp_not:N \l__hook_patch_prefixes_tl
      { \exp_not:N \__hook_make_prefixes:w \cs_prefix_spec:N #2 / / }
        }
```

Here we redefine the hook to have the right number of arguments. Disabling the hook, undefining the `parameter` token list then calling `\__hook_make_usable:nn` are enough to redefine the hook to the extent we want. Code stored in the hook and other metadata about it are not lost in the process.

```
    \__hook_disable:n { cmd / #3 / #4 }
    \cs_undefine:c { c__hook_cmd / #3 / #4_parameter_tl }
    \__hook_make_usable:nn { cmd / #3 / #4 } { \l__hook_patch_num_args_int }
```

Now call `\__hook_redefine_with_hooks:Nnnn` with the macro being redefined in `#1`, then `\UseHook{cmd/<name>/before}` in `#2` or `\UseHook{cmd/<name>/after}` in `#3` (one is always empty), and in `#4` the ⟨*replacement text*⟩ of the macro.

```
    \use:e
      {
        \__hook_redefine_with_hooks:Nnnn \exp_not:N #2
        \str_if_eq:nnTF {#4} { after }
          { \use_ii_i:nn }
          { \use:nn }
            { {
                \__hook_exp_not:NN \exp_not:N \UseHookWithArguments
                  { cmd / #3 / #4 } { \int_use:N \l__hook_patch_num_args_int }
                  \__hook_braced_parameter:n { cmd / #3 / #4 }
            } }
            { { } }
            { \__hook_exp_not:NN \exp_not:V \l__hook_replace_text_tl }
      }
```

Finally, update the hook code.

```
    \__hook_update_hook_code:n { cmd / #3 / #4 }
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\__hook_patch_expand_redefine:NNnn}
⟨latexrelease⟩                  {cmd~hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_patch_expand_redefine:NNnn #1 #2 #3 #4
⟨latexrelease⟩  {
⟨latexrelease⟩    \__hook_patch_debug:x { ++~command~can~be~patched~without~rescanning }
⟨latexrelease⟩    \int_set:Nn \l__hook_patch_num_args_int
⟨latexrelease⟩      {
⟨latexrelease⟩        \exp_args:Nf \str_count:n { \__kernel_cs_parameter_spec:N #2 } / 2
⟨latexrelease⟩        \bool_if:NT #1 { -1 }
⟨latexrelease⟩      }
⟨latexrelease⟩    \int_compare:nNnTF { \l__hook_patch_num_args_int } > { \c_zero_int }
⟨latexrelease⟩      {
⟨latexrelease⟩        \tl_set:Nx \l__hook_tmpa_tl { \bool_if:NTF #1 { [ ] } { { } } }
⟨latexrelease⟩        \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
⟨latexrelease⟩          { \tl_put_right:Nn \l__hook_tmpa_tl { { } } }
⟨latexrelease⟩        \exp_args:NNo \exp_args:No \__hook_if_has_hash:nTF
⟨latexrelease⟩            { \exp_after:wN #2 \l__hook_tmpa_tl }
⟨latexrelease⟩          { \cs_set_eq:NN \__hook_exp_not:n \exp_not:n }
```

```
⟨latexrelease⟩          { \cs_set_eq:NN \__hook_exp_not:n \use:n }
⟨latexrelease⟩        \cs_set_protected:Npn \__hook_tmp:w ##1 ##2
⟨latexrelease⟩          {
⟨latexrelease⟩            ##1 \l__hook_param_text_tl   { \use:n ##2 }
⟨latexrelease⟩            ##1 \l__hook_replace_text_tl { \__hook_exp_not:n {##2} }
⟨latexrelease⟩          }
⟨latexrelease⟩        \bool_if:NTF #1
⟨latexrelease⟩          { \__hook_tmp:w \tl_set:Nx { [ \c__hook_hash_tl 1 ] } }
⟨latexrelease⟩          { \__hook_tmp:w \tl_set:Nx { { \c__hook_hash_tl 1 } } }
⟨latexrelease⟩        \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
⟨latexrelease⟩          { \__hook_tmp:w \tl_put_right:Nx { { \c__hook_hash_tl ##1 } } }
⟨latexrelease⟩        \tl_set:Nx \l__hook_replace_text_tl
⟨latexrelease⟩          { \exp_not:N #2 \exp_not:V \l__hook_replace_text_tl }
⟨latexrelease⟩        \tl_set:Nx \l__hook_replace_text_tl
⟨latexrelease⟩          {
⟨latexrelease⟩            \token_if_eq_meaning:NNTF \__hook_exp_not:n \exp_not:n
⟨latexrelease⟩              { \exp_args:NNV \exp_args:No \__hook_double_hashes:n }
⟨latexrelease⟩              { \exp_args:NV \exp_not:o }
⟨latexrelease⟩                  \l__hook_replace_text_tl
⟨latexrelease⟩          }
⟨latexrelease⟩        \cs_set_eq:NN \__hook_def_cmd:w \tex_gdef:D
⟨latexrelease⟩        \cs_set_eq:NN \__hook_exp_not:NN \prg_do_nothing:
⟨latexrelease⟩      }
⟨latexrelease⟩      {
⟨latexrelease⟩        \tl_clear:N \l__hook_param_text_tl
⟨latexrelease⟩        \tl_set_eq:NN \l__hook_replace_text_tl #2
⟨latexrelease⟩        \cs_set_eq:NN \__hook_def_cmd:w \tex_xdef:D
⟨latexrelease⟩        \cs_set:Npn \__hook_exp_not:NN ##1 { \exp_not:N ##1 \exp_not:N }
⟨latexrelease⟩      }
⟨latexrelease⟩    \group_begin:
⟨latexrelease⟩      \int_set:Nn \tex_escapechar:D { `\/ }
⟨latexrelease⟩      \use:x
⟨latexrelease⟩        {
⟨latexrelease⟩    \group_end:
⟨latexrelease⟩    \tl_set:Nx \exp_not:N \l__hook_patch_prefixes_tl
⟨latexrelease⟩      { \exp_not:N \__hook_make_prefixes:w \cs_prefix_spec:N #2 / / }
⟨latexrelease⟩        }
⟨latexrelease⟩    \use:x
⟨latexrelease⟩      {
⟨latexrelease⟩        \__hook_redefine_with_hooks:Nnnn \exp_not:N #2
⟨latexrelease⟩        \str_if_eq:nnTF {#4} { after }
⟨latexrelease⟩          { \use_ii_i:nn }
⟨latexrelease⟩          { \use:nn }
⟨latexrelease⟩            { { \__hook_exp_not:NN \exp_not:N \UseHook { cmd / #3 / #4 } } }
⟨latexrelease⟩            { { } }
⟨latexrelease⟩            { \__hook_exp_not:NN \exp_not:V \l__hook_replace_text_tl }
⟨latexrelease⟩      }
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

Now that all the needed tools are ready, without further ado we'll redefine the command. The definition uses the prefixes gathered in `\l__hook_patch_prefixes_tl`, a primitive `\__hook_def_cmd:w` (which is `\tex_gdef:D` or `\tex_xdef:D`) to avoid adding extra prefixes, and the ⟨*parameter text*⟩ from `\l__hook_param_text_tl`.

Then finally, in the body of the definition, we insert #2, which is cmd/#1/before or empty, #4 which is the ⟨*replacement text*⟩, and #3 which is cmd/#1/after or empty.

```
\cs_new_protected:Npn \__hook_redefine_with_hooks:Nnnn #1 #2 #3 #4
  {
    \l__hook_patch_prefixes_tl
      \exp_after:wN \__hook_def_cmd:w
        \exp_after:wN #1 \l__hook_param_text_tl
      { #2  #4  #3 }
  }
```

Here's the auxiliary that makes the prefix control sequences for the redefinition. Each item has to be \tl_trim_spaces:n'd because the last item (and not any other) has a trailing space.

```
\cs_new:Npn \__hook_make_prefixes:w / #1 /
  {
    \tl_if_empty:nF {#1}
      {
        \exp_not:c { tex_ \tl_trim_spaces:n {#1} :D }
        \__hook_make_prefixes:w /
      }
  }
```

(*End of definition for* __hook_patch_expand_redefine:NNnn*,* __hook_redefine_with_hooks:Nnnn*, and* __hook_make_prefixes:w.)

Here are some auxiliaries for the contraption above.

__hook_if_has_hash_p:n
__hook_if_has_hash:nTF
__hook_if_has_hash:w
__hook_if_has_hash_check:w

__hook_if_has_hash:nTF searches the token list #1 for a catcode 6 token, and if any is found, it returns true, and false otherwise. The searching doesn't care about preserving groups or spaces: we can ignore those safely (braces are removed) so that searching is as fast as possible.

```
\prg_new_conditional:Npnn \__hook_if_has_hash:n #1 { TF }
  { \__hook_if_has_hash:w #1 ## \s__hook_mark }
\cs_new:Npn \__hook_if_has_hash:w #1
  {
    \tl_if_single_token:nTF {#1}
      {
        \token_if_eq_catcode:NNTF ## #1
          { \__hook_if_has_hash_check:w }
          { \__hook_if_has_hash:w }
      }
      { \__hook_if_has_hash:w #1 }
  }
\cs_new:Npn \__hook_if_has_hash_check:w #1 \s__hook_mark
  { \tl_if_empty:nTF {#1} { \prg_return_false: } { \prg_return_true: } }
```

(*End of definition for* __hook_if_has_hash:nTF*,* __hook_if_has_hash:w*, and* __hook_if_has_hash_check:w.)

__hook_double_hashes:n
__hook_double_hashes:w
__hook_double_hashes_output:N
__hook_double_hashes_stop:w
__hook_double_hashes_group:n
__hook_double_hashes_space:w

__hook_double_hashes:n loops through the token list #1 and duplicates any catcode 6 token, and expands tokens \ifx-equal to \c__hook_hashes_tl, and leaves all other tokens \notexpanded with \exp_not:N. Unfortunately pairs of explicit catcode 1 and catcode 2 character tokens are normalised to $\{_1$ and $\}_1$ because it's not feasible to expandably detect the character code (*maybe* it could be done using something along the

lines of https://tex.stackexchange.com/a/527538, but it's far too much work for close to zero benefit).

`\__hook_double_hashes:w` is the tail-recursive loop macro, that tests which of the three types of item is in the head of the token list.

```
\cs_new:Npn \__hook_double_hashes:n #1
  { \__hook_double_hashes:w #1 \q__hook_recursion_tail \q__hook_recursion_stop }
\cs_new:Npn \__hook_double_hashes:w #1 \q__hook_recursion_stop
  {
    \tl_if_head_is_N_type:nTF {#1}
      { \__hook_double_hashes_output:N }
      {
        \tl_if_head_is_group:nTF {#1}
          { \__hook_double_hashes_group:n }
          { \__hook_double_hashes_space:w }
      }
    #1 \q__hook_recursion_stop
  }
```

`\__hook_double_hashes_output:N` checks for the end of the token list, then checks if the token is `\c__hook_hashes_tl`, and if so just leaves it.

```
\cs_new:Npn \__hook_double_hashes_output:N #1
  {
    \if_meaning:w \q__hook_recursion_tail #1
      \__hook_double_hashes_stop:w
    \fi:
    \if:w ?
      \if_meaning:w \c__hook_hash_tl #1 ! \fi:
      \if_meaning:w \c__hook_hashes_tl #1 ! \fi:
          ?
    \else:
```

(this `\use_i:nnnn` uses `\fi:` and consumes `\use:n`, the whole `\if_catcode:w` block, and the `\exp_not:N`, leaving just `#1` which is `\c__hook_hashes_tl`.)

```
      \use_i:nnnn
    \fi:
    \use:n
      {
```

If `#1` is not `\c__hook_hashes_tl`, then check if its catcode is 6, and if so, leave it doubled in `\exp_not:n` and consume the following `\exp_not:N #1`.

```
        \if_catcode:w ## \exp_not:N #1
          \exp_after:wN \use_ii:nnnn
        \fi:
        \use_none:n
          { \exp_not:n { #1 #1 } }
      }
```

If both previous tests returned `false`, then leave the token unexpanded and resume the loop.

```
    \exp_not:N #1
    \__hook_double_hashes:w
  }
\cs_new:Npn \__hook_double_hashes_stop:w #1 \q__hook_recursion_stop { \fi: }
```

Dealing with spaces and grouped tokens is trivial:

```
\cs_new:Npn \__hook_double_hashes_group:n #1
  { { \__hook_double_hashes:n {#1} } \__hook_double_hashes:w }
\exp_last_unbraced:NNo
\cs_new:Npn \__hook_double_hashes_space:w \c_space_tl
  { ~ \__hook_double_hashes:w }
```

(*End of definition for* `\__hook_double_hashes:n` *and others.*)

4.5.2 Patching by retokenization

At this point we've drained the possibilities of patching a command by expansion-and-redefinition, so we have to resort to patching by retokenizing the command. Patching by retokenization is done by getting the `\meaning` of the command, doing the necessary manipulations on the generated string, and the retokenizing that again by using `\scantokens`.

Patching by retokenization is definitely a riskier business, because it relies that the tokens printed by `\meaning` produce the exact same tokens as the ones in the original definition. That is, the catcode régime must be exactly(ish) the same, and there is no way of telling except by trial and error.

`\__hook_retokenize_patch:Nnn`

This is the macro that will control the whole process. First we'll try out one final, rather trivial case, of a command with no arguments; that is, a token list. This case can be patched with the expand-and-redefine routine but it has to be the very last case tested for, because most (all?) robust commands start with a top-level macro with no arguments, so testing this first would short-circuit `\robust@command@act` and the top-level macros would be incorrectly patched. In that case, we just check if the `\cs_parameter_spec:N` is empty, and call `\__hook_patch_expand_redefine:NNnn`.

```
\cs_new_protected:Npn \__hook_retokenize_patch:Nnn #1 #2 #3
  {
    \str_if_eq:eeTF { \__kernel_cs_parameter_spec:N #1 } { }
      { \__hook_patch_expand_redefine:NNnn \c_false_bool #1 {#2} {#3} }
      {
        \__hook_patch_debug:x { ..~command~can~only~be~patched~by~rescanning }
```

Otherwise, we start the actual patching by retokenization job. The code calls `\__hook_try_patch_with_catcodes:Nnnnw` with a different catcode setting:

- The current catcode setting;
- Switching the catcode of `@`;
- Switching the expl3 syntax on or off;
- Both of the above.

If patching succeeds, `\__hook_try_patch_with_catcodes:Nnnnw` has the side-effect of patching the macro `#1` (which may be an internal from the command whose name is `#2`).

```
        \tl_set:Nx \l__hook_tmpa_tl
          {
            \int_compare:nNnTF { \char_value_catcode:n {`\@ } } = { 12 }
              { \exp_not:N \makeatletter } { \exp_not:N \makeatother }
          }
        \tl_set:Nx \l__hook_tmpb_tl
```

```
          {
            \bool_if:NTF \l__kernel_expl_bool
              { \ExplSyntaxOff } { \ExplSyntaxOn }
          }
        \use:x
          {
            \exp_not:N \__hook_try_patch_with_catcodes:Nnnnw
                \exp_not:n { #1 {#2} {#3} }
              { \prg_do_nothing: }
              { \exp_not:V \l__hook_tmpa_tl } % @
              { \exp_not:V \l__hook_tmpb_tl } % _:
              {
                \exp_not:V \l__hook_tmpa_tl   % @
                \exp_not:V \l__hook_tmpb_tl   % _:
              }
          }
              \q_recursion_tail \q_recursion_stop
```

If no catcode setting succeeds, give up and raise an error. The command isn't changed in any way in that case.

```
          {
            \msg_error:nnxx { hooks } { cant-patch }
              { \c_backslash_str #2 } { retok }
          }
      }
  }
```

(*End of definition for* `\__hook_retokenize_patch:Nnn`.)

`\__hook_try_patch_with_catcodes:Nnnnw` This function is a simple wrapper around `\__hook_cmd_if_scanable:NnTF` and `\__hook_patch_retokenize:Nnnn` if the former returns ⟨*true*⟩, plus some debug messages.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_try_patch_with_catcodes:Nnnnw}
⟨latexrelease⟩                  {cmd~hooks~with~args}
\cs_new_protected:Npn \__hook_try_patch_with_catcodes:Nnnnw #1 #2 #3 #4
  {
    \quark_if_recursion_tail_stop_do:nn {#4} { \use:n }
    \__hook_patch_debug:x { ++~trying~to~patch~by~retokenization }
    \__hook_cmd_if_scanable:NnTF {#1} {#4}
      {
        \__hook_patch_debug:x { ++~macro~can~be~retokenized~cleanly }
        \__hook_patch_debug:x { ==~retokenizing~macro~now }
        \__hook_patch_retokenize:Nnnn #1 { cmd / #2 / #3 } {#3} {#4}
        \use_i_delimit_by_q_recursion_stop:nw \use_none:n
      }
      {
        \__hook_patch_debug:x { --~macro~cannot~be~retokenized~cleanly }
        \__hook_try_patch_with_catcodes:Nnnnw #1 {#2} {#3}
      }
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\__hook_try_patch_with_catcodes:Nnnnw}
⟨latexrelease⟩                  {cmd~hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_try_patch_with_catcodes:Nnnnw #1 #2 #3 #4
⟨latexrelease⟩   {
⟨latexrelease⟩     \quark_if_recursion_tail_stop_do:nn {#4} { \use:n }
```

```
⟨latexrelease⟩     \__hook_patch_debug:x { ++~trying~to~patch~by~retokenization }
⟨latexrelease⟩     \__hook_cmd_if_scanable:NnTF {#1} {#4}
⟨latexrelease⟩       {
⟨latexrelease⟩         \__hook_patch_debug:x { ++~macro~can~be~retokenized~cleanly }
⟨latexrelease⟩         \__hook_patch_debug:x { ==~retokenizing~macro~now }
⟨latexrelease⟩         \__hook_patch_retokenize:Nnnn #1 {#2} {#3} {#4}
⟨latexrelease⟩         \use_i_delimit_by_q_recursion_stop:nw \use_none:n
⟨latexrelease⟩       }
⟨latexrelease⟩       {
⟨latexrelease⟩         \__hook_patch_debug:x { ---macro~cannot~be~retokenized~cleanly }
⟨latexrelease⟩         \__hook_try_patch_with_catcodes:Nnnnw #1 {#2} {#3}
⟨latexrelease⟩       }
⟨latexrelease⟩   }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_try_patch_with_catcodes:Nnnnw`.)

`\kerneltmpDoNotUse` This is an oddity required to be safe (as safe as reasonably possible) when patching the command. The entirety of

⟨*prefixes*⟩ `\def` ⟨*cs*⟩ ⟨*parameter text*⟩ `{`⟨*replacement text*⟩`}`

will go through `\scantokens`. The ⟨*parameter text*⟩ and ⟨*replacement text*⟩ are what we are trying to retokenize, so not much worry there. The other items, however, should "just work", so some care is needed to not use too fancy catcode settings. Therefore we can't use an expl3-named macro for ⟨*cs*⟩, nor the expl3 versions of `\def` or the ⟨*prefixes*⟩. That is why the definitions that will eventually go into `\scantokens` will use the oddly (but hopefully clearly)-named `\kerneltmpDoNotUse`:

```
\cs_new_eq:NN \kerneltmpDoNotUse !
```

PhO: Maybe this can be avoided by running the ⟨`parameter text`⟩ *and the* ⟨`replacement text`⟩ *separately through* `\scantokens` *and then putting everything together at the end.*

(*End of definition for* `\kerneltmpDoNotUse`.)

`\__hook_patch_required_catcodes:` Here are the catcode settings that are *mandatory* when retokenizing commands. These are the minimum necessary settings to perform the definitions: they identify control sequences, which must be escaped with $\backslash_0$, delimit the definition with $\{_1$ and $\}_2$, and mark parameters with $\#_6$. Everything else may be changed, but not these.

```
\cs_new_protected:Npn \__hook_patch_required_catcodes:
  {
    \char_set_catcode_escape:N \\
    \char_set_catcode_group_begin:N \{
    \char_set_catcode_group_end:N \}
    \char_set_catcode_parameter:N \#
    % \int_set:Nn \tex_endlinechar:D { -1 }
    % \int_set:Nn \tex_newlinechar:D { -1 }
  }
```

PhO: etoolbox sets the `\endlinechar` *and* `\newlinechar` *when patching, but as far as I tested these didn't make much of a difference, so I left them out for now. Maybe* `\newlinechar=-1` *avoids a space token being added after the definition.*

PhO: If the patching is split by ⟨`parameter text`⟩ *and* ⟨`replacement text`⟩*, then only # will have to stay in that list.*

PhO: Actually now that we patch `\UseHook{cmd/foo/before}`*, all the tokens there need to have the right catcodes, so this list now includes all lowercase letters, U and H, the slash, and whatever characters in the command name... sigh...*

(*End of definition for* `\__hook_patch_required_catcodes:`.)

`\__hook_cmd_if_scanable:NnTF` Here we'll do a quick test if the command being patched can in fact be retokenized with the specific catcode setting without changing in meaning. The test is straightforward:

1. apply `\meaning` to the command;
2. split the ⟨*prefixes*⟩, ⟨*parameter text*⟩ and ⟨*replacement text*⟩ and arrange them as

   ⟨*prefixes*⟩`\def\kerneltmpDoNotUse`⟨*parameter text*⟩`{`⟨*replacement text*⟩`}`

3. rescan that with the given catcode settings, and do the definition; then finally
4. compare `\kerneltmpDoNotUse` with the original command.

If both are `\ifx`-equal, the command can be safely patched.

```
\prg_new_protected_conditional:Npnn \__hook_cmd_if_scanable:Nn #1 #2 { TF }
  {
    \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
    \cs_set_eq:NN \__hook_tmp:w \scan_stop:
    \use:x
      {
        \cs_set:Npn \__hook_tmp:w
            ####1 \tl_to_str:n { macro: } ####2 -> ####3 \s__hook_mark
          { ####1 \def \kerneltmpDoNotUse ####2   {####3} }
        \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
          { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
      }
    \tl_rescan:nV { #2 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
    \token_if_eq_meaning:NNTF #1 \kerneltmpDoNotUse
      { \prg_return_true: }
      { \prg_return_false: }
  }
```

(*End of definition for* `\__hook_cmd_if_scanable:NnTF`.)

`\__hook_guess_arg_count:NN`
`\__hook_guess_arg_count:wN`
`\__hook_guess_arg_count:nw`

Looks at the parameter text of a macro, and counts the parameters by looking at the number after a `#`, and checking if they are sequential. This macro assumes that all parameters are marked with hashes, and not other characters, and that there is no "trick parameter".

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_guess_arg_count:NN}
⟨latexrelease⟩                    {cmd~hooks~with~args}
\cs_new_protected:Npn \__hook_guess_arg_count:NN #1
  {
    \exp_after:wN \__hook_guess_arg_count:wN
      \token_to_meaning:N #1 \s__hook_mark
  }
\exp_last_unbraced:NNNNo
\cs_new_protected:Npx \__hook_guess_arg_count:wN
    #1 { \tl_to_str:n { macro: } } #2 \s__hook_mark #3
  {
```

```
    \int_set:Nn #3
      {
        \exp_not:N \__hook_guess_arg_count:nw { 0 } #2
          \c_hash_str 0 \s__hook_mark
      }
  }
\use:e
  { \cs_new:Npn \exp_not:N \__hook_guess_arg_count:nw #1 #2 \c_hash_str #3 }
  {
    \int_compare:nNnTF { #1 + 1 } = {#3}
      { \__hook_guess_arg_count:nw {#3} }
      { #1 \__hook_use_none_delimit_by_s_mark:w }
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\__hook_guess_arg_count:NN}
⟨latexrelease⟩                  {cmd~hooks~with~args}
⟨latexrelease⟩\cs_undefine:N \__hook_guess_arg_count:NN
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_guess_arg_count:NN`*,* `\__hook_guess_arg_count:wN`*, and* `\__hook_guess_arg_count:nw`*.*)

`\__hook_patch_retokenize:Nnnn` Then, if `\__hook_cmd_if_scanable:NnTF` returned true, we can go on and patch the command.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_patch_retokenize:Nnnn}
⟨latexrelease⟩                  {cmd~hooks~with~args}
\cs_new_protected:Npn \__hook_patch_retokenize:Nnnn #1 #2 #3 #4
  {
```

Here, when patching by retokenization, we can only guess the number of arguments of the macro.

```
    \__hook_guess_arg_count:NN #1 \l__hook_patch_num_args_int
```

Then we redefine the hook to have the right number of arguments. Disabling the hook, undefining the `parameter` token list then calling `\__hook_make_usable:nn` are enough to redefine the hook to the extent we want. Code stored in the hook and other metadata about it are not lost in the process.

```
    \__hook_disable:n {#2}
    \cs_undefine:c { c__hook_#2_parameter_tl }
    \__hook_make_usable:nn {#2} { \l__hook_patch_num_args_int }
    \tl_set:Ne \l__hook_tmpa_tl
      { \exp_args:Ne \tl_to_str:n { \__hook_braced_parameter:n {#2} } }
    \use:x
      {
        \str_replace_all:Nnn \exp_not:N \l__hook_tmpa_tl
          { #### } { \c_hash_str }
      }
```

Then, make make some things `\relax` to avoid lots of `\noexpand` below.

```
    \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
    \cs_set_eq:NN \__hook_tmp:w \scan_stop:
    \use:x
      {
```

Now we'll define `\__hook_tmp:w` such that it splits the `\meaning` of the macro (`#1`) into its three parts:

```
####1. ⟨prefixes⟩
```

```
####2. ⟨parameter text⟩
```

```
####3. ⟨replacement text⟩
```

and arrange that a complete definition, then place the `before` or `after` hooks around the ⟨*replacement text*⟩: accordingly.

```
        \cs_set:Npn \__hook_tmp:w
            ####1 \tl_to_str:n { macro: } ####2 -> ####3 \s__hook_mark
          {
            ####1 \def \kerneltmpDoNotUse ####2
              {
                \str_if_eq:nnT {#3} { before }
                  {
                    \token_to_str:N \UseHookWithArguments {#2}
                      { \int_use:N \l__hook_patch_num_args_int }
                      \l__hook_tmpa_tl
                  }
                ####3
                \str_if_eq:nnT {#3} { after }
                  {
                    \token_to_str:N \UseHookWithArguments {#2}
                      { \int_use:N \l__hook_patch_num_args_int }
                      \l__hook_tmpa_tl
                  }
              }
          }
```

Now we just have to get the `\meaning` of the command being patched and pass it through the meat grinder above.

```
        \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
          { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
      }
```

Now rescan with the given catcode settings (overridden by the `\__hook_patch_required_catcodes:`), and implicitly (by using the rescanned token list) carry out the definition from above.

```
    \tl_rescan:nV { #4 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
```

And to close, copy the newly-defined command into the old name and the patching is finally completed:

```
    \cs_gset_eq:NN #1 \kerneltmpDoNotUse
```

Finally, update the hook code.

```
    \__hook_update_hook_code:n {#2}
  }
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\__hook_patch_retokenize:Nnnn}
⟨latexrelease⟩                  {cmd~hooks~with~args}
⟨latexrelease⟩\cs_gset_protected:Npn \__hook_patch_retokenize:Nnnn #1 #2 #3 #4
⟨latexrelease⟩  {
⟨latexrelease⟩    \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
⟨latexrelease⟩    \cs_set_eq:NN \__hook_tmp:w \scan_stop:
⟨latexrelease⟩    \use:x
```

```
⟨latexrelease⟩      {
⟨latexrelease⟩        \cs_set:Npn \__hook_tmp:w
⟨latexrelease⟩            ####1 \tl_to_str:n { macro: } ####2 -> ####3 \s__hook_mark
⟨latexrelease⟩          {
⟨latexrelease⟩            ####1 \def \kerneltmpDoNotUse ####2
⟨latexrelease⟩              {
⟨latexrelease⟩                \str_if_eq:nnT {#3} { before }
⟨latexrelease⟩                  { \token_to_str:N \UseHook { cmd / #2 / #3 } }
⟨latexrelease⟩                ####3
⟨latexrelease⟩                \str_if_eq:nnT {#3} { after }
⟨latexrelease⟩                  { \token_to_str:N \UseHook { cmd / #2 / #3 } }
⟨latexrelease⟩              }
⟨latexrelease⟩          }
⟨latexrelease⟩        \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
⟨latexrelease⟩          { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
⟨latexrelease⟩      }
⟨latexrelease⟩    \tl_rescan:nV { #4 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
⟨latexrelease⟩    \cs_gset_eq:NN #1 \kerneltmpDoNotUse
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\__hook_patch_retokenize:Nnnn`.)

4.6 Messages

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}{wrong-cmd-hook}%
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{wrong-cmd-hook}%
⟨latexrelease⟩                  {Standardise~generic~hook~names}
⟨latexrelease⟩\msg_new:nnnn { hooks } { wrong-cmd-hook }
⟨latexrelease⟩  {
⟨latexrelease⟩    Generic~hook~`cmd/#1/#2'~is~invalid.
⟨latexrelease⟩%    The~hook~should~be~`cmd/#1/before'~or~`cmd/#1/after'.
⟨latexrelease⟩  }
⟨latexrelease⟩  {
⟨latexrelease⟩    You~tried~to~add~a~generic~hook~to~command~\iow_char:N \\#1,~but~`#2'~
⟨latexrelease⟩    is~an~invalid~component.~Only~`before'~or~`after'~are~allowed.
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
\msg_new:nnnn { hooks } { cant-patch }
  {
    Generic~hooks~cannot~be~added~to~'#1'.
  }
  {
    You~tried~to~add~a~hook~to~'#1',~but~LaTeX~was~unable~to~
    patch~the~command~because~it~\__hook_unpatchable_cases:n {#2}.
  }
\cs_new:Npn \__hook_unpatchable_cases:n #1
  {
    \str_case:nn {#1}
      {
        { undef } { doesn't~exist }
        { macro } { is~not~a~macro }
```

```
        { expl3 } { is~a~private~expl3~macro }
        { retok } { can't~be~retokenized~cleanly }
      }
  }
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{ltcmdhooks}%
⟨latexrelease⟩                  {The~hook~management~system~for~commands}
⟨latexrelease⟩
```

The command `\__hook_cmd_begindocument_code:` is used in an internal hook, so we need to make sure it has a harmless definition after rollback as that will not remove it from the kernel hook.

```
⟨latexrelease⟩\cs_set_eq:NN \__hook_cmd_begindocument_code: \prg_do_nothing:
⟨latexrelease⟩
⟨latexrelease⟩\EndModuleRelease
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨@@=⟩
```

File 10
ltsockets.dtx

Abstract

This code implements sockets which are places in the code into which predeclared chunks of code (plugs) can be placed. Both the sockets and the plugs are "named" and each socket is assigned exactly one plug at any given time.

1 Introduction

A LaTeX source file is transformed into a typeset document by executing code for each command or environment in the document source. Through various steps this code transforms the input and eventually generates typeset output appearing in a "galley" from which individual pages are cut off in an asynchronous way. This page generating process is normally not directly associated with commands in the input[21] but is triggered whenever the galley has received enough material to form another page (giving current settings).

As part of this transformation input data may get stored in some form and later reused, for example, as part of the output routine processing.

2 Configuration of the transformation process

There are three different major methods offered by LaTeX to configure the transformation process:

- through the template mechanism,
- through the hook mechanism, or
- through sockets and plugs.

They offer different possibilities (with different features and limitations) and are intended for specific use cases, though it is possible to combine them.

2.1 The template mechanism

The template mechanism is intended for more complex document-level elements (e.g., headings such as `\section` or environments like `itemize`). The template code implements the overall processing logic for such an element and offers a set of parameters to influence the final result.

The document element is then implemented by (a) selecting a suitable template (there may be more than one available for the kind of document element) and (b) by setting its parameters to desired values. This then forms a so-called instance which is executed when the document element is found in the source.

By altering the parameter values (in a document class or in the document preamble) or, if more drastic layout changes are desired, by selecting a different template and

[21] Excepts for directives such as `\newpage`.

then adjusting its parameters, a wide variety of layouts can be realized through simple configuration setups without the need to develop new code.

The target audience of this method are therefore document class developers or users who wish to alter an existing layout (implemented by a document class) in certain (minor) ways.

The template mechanism is currently documented as part of the `xtemplate` package and one more elaborate implementation can be found as part of the `latex-lab` code for lists (to be documented further).

2.2 The hook mechanism

Hooks are places in the kernel code (or in packages) that offer packages the possibility to inject additional code at specific points in the processing in a controlled way without the need to replace the existing code block (and thereby overwriting modifications/extensions made by other packages). The target audience is therefore mainly package developers, even though some hooks can be useful for document authors.

Obviously, what can reasonably be added into a hook depends on the individual hook (hopefully documented as part of the hook documentation), but in general the idea behind hooks is that more than one package could add code into the hook at the same time. Perhaps the most famous hook (that LaTeX had for a very long time) is `begindocument` into which many packages add code to through `\AtBeginDocument{`⟨*code*⟩`}` (which is nowadays implemented as a shorthand for `\AddToHook{begindocument}{`⟨*code*⟩`}`). To resolve possible conflicts between injections by different packages there is a rule mechanism by which code chunks in a hook can be ordered in a certain way and by which incompatible packages can be detected if a resolution is impossible.

In contrast to template code, there is no standard configuration method through parameters for hooks, i.e., the code added to a hook "is" the configuration. If it wants to provide for configuration through parameters it has to also provide its own method to set such parameters in some way. However, in that case it is likely that using a hook is not the right approach and the developer better calls a template instance instead which then offers configuration through a key/value interface.

In most cases, hooks do not take any arguments as input. Instead, the data that they can (and are allowed to) access depends on the surrounding context.

For example, the various hooks available during the page shipout process in LaTeX's output routine can (and have to) access the accumulated page material stored in a box named `\ShipoutBox`. This way, code added to, say, the `shipout/before` hook could access the page content, alter it, and then write it back into `\ShipoutBox` and any other code added to this hook could then operate on the modified content. Of course, for such a scheme to work the code prior to executing the hook would need to setup up data in appropriate places and the hook documentation would need to document what kind of storage can be accessed (and possibly altered) by the hook.

There are also hooks that take arguments (typically portions of document data) and in that case the hook code can access these arguments through `#1`, `#2`, etc.

The hook mechanism is documented in `lthooks-doc.pdf`.

2.3 The socket mechanism

In some cases there is code that implements a certain programming logic (for example, combining footnotes, floats, and the text for the current page to be shipped out) and

if this logic should change (e.g., footnotes to be placed above bottom floats instead of below) then this whole code block needs to be replaced with different code.

In theory, this could be implemented with templates, i.e., the code simply calls some instance that implements the logic and that instance is altered by selecting a different templates and/or adjusting their parameters. However, in many cases customization through parameters is overkill in such a case (or otherwise awkward, because parameterization is better done on a higher level instead of individually for small blocks of code) and using the template mechanism just to replace one block of code with a different one results in a fairly high performance hit. It is therefore usually not a good choice.

In theory, it would also be possible to use a hook, but again that is basically a misuse of the concept, because in this use case there should never be more than one block of code inside the hook; thus, to alter the processing logic one would need to set up rules that replace code rather than (as intended) execute all code added to the hook.

For this reason LaTeX now offers a third mechanism: "sockets" into which one can place exactly one code block — a "plug".

In a nutshell: instead of having a fixed code block somewhere as part of the code, implementing a certain programming logic there is a reference to a named socket at this point. This is done by first declaring the named socket with:

`\NewSocket{`⟨*socket-name*⟩`}{`⟨*number-of-inputs*⟩`}`

This is then referenced at the point where the replaceable code block should be executed with:

`\UseSocket{`⟨*socket-name*⟩`}`

or, if the socket should take a number of inputs (additional arguments beside the name) with

`\UseSocket{`⟨*socket-name*⟩`}{`⟨arg_1⟩`}...{`⟨$arg_{\langle \texttt{number-of-inputs} \rangle}$⟩`}`

In addition, several code blocks (a.k.a. plugs) implementing different logic for this socket are set up, each with a declaration of the form:

`\NewSocketPlug{`⟨*socket-name*⟩`}{`⟨*socket-plug-name*⟩`}{`⟨*code*⟩`}`

Finally, one of them is assigned to the socket:

`\AssignSocketPlug{`⟨*socket-name*⟩`}{`⟨*socket-plug-name*⟩`}`

If the programming logic should change, then all that is necessary is to make a new assignment with `\AssignSocketPlug` to a different `{`⟨*socket-plug-name*⟩`}`. This assignment obeys scope so that an environment can alter a socket without the need to restore the previous setting manually.

If the socket takes inputs, then those need to be provided to `\UseSocket` and in that case they can be referenced in the ⟨*code*⟩ argument of `\NewSocketPlug` with `#1`, `#2`, etc.

In most cases a named socket is used only in a single place, but there is, of course, nothing wrong with using it in several places, as long as the code in all places is supposed to change in the same way.

2.3.1 Examples

We start by declaring a new socket named `foo` that expects two inputs:

```
\NewSocket{foo}{2}
```

Such a declaration has do be unique across the whole LaTeX run. Thus, if another package attempts to use the same name (regardless of the number of inputs) it will generate an error:

```
\NewSocket{foo}{2}
\NewSocket{foo}{1}
```

Both declarations would therefore produce:

```
! LaTeX socket Error: Socket 'foo' already declared!
```

You also get an error if you attempt to declare some socket plug and the socket name is not yet declared, e.g.,

```
\NewSocketPlug{baz}{undeclared}{some code}
```

generates

```
! LaTeX socket Error: Socket 'baz' undeclared!
```

Setting up plugs for the socket is done like this:

```
\NewSocketPlug{foo}{plug-A}
   {\begin{quote}\itshape foo-A: #1!#2\end{quote}}
\NewSocketPlug{foo}{plug-B}
   {\begin{quote}\sffamily foo-B: #2\textsuperscript{2}\end{quote}}
```

This will set up the plugs `plug-A` and `plug-B` for this socket.

We still have to assign one or the other to the socket, thus without doing that the line

```
\UseSocket{foo}{hello}{world}
```

produces nothing because the default plug for sockets with 2 inputs is `noop` (which grabs the additional arguments and throws them away).[22]

So let's do the assignment

```
\AssignSocketPlug{foo}{plug-A}
```

and then

```
\UseSocket{foo}{hello}{world}
```

will properly typeset

foo-A: hello!world

and after

```
\AssignSocketPlug{foo}{plug-B}
```

[22] If socket `foo` would have been a socket with one input, then the default plug would be `identity`, in which case the socket input would remain without braces and gets typeset!

and another call to

```
\UseSocket{foo}{hello}{world}
```

we get

foo-B: world[2]

If we attempt to assign a plug that was not defined, e.g.,

```
\AssignSocketPlug{foo}{plug-C}
```

then we get an error during the assignment

```
! LaTeX socket Error: Plug 'plug-C' for socket 'foo' undeclared!
```

and the previous assignment remains in place.

To see what is known about a socket and its plugs you can use `\ShowSocket` or `\LogSocket` which displays information similar to this on the terminal or in the transcript file:

```
Socket foo:
    number of inputs = 2
    available plugs = noop, plug-A, plug-B
    current plug = plug-B
    definition = \long macro:#1#2->\begin {quote}\sffamily
foo-B: #2\textsuperscript {2}\end {quote}
```

2.3.2 Details and semantics

In this section we collect some normative statements.

- From a functional point of view sockets are like simple TEX macros, i.e., they expect 0 to 9 mandatory arguments (the socket inputs) and get replaced by their "expansion"
- A socket is "named" and the name consists of ASCII letters `[a-z]`, `[A-Z]`, `[0-9]`, `[-/@]` only
- Socket names have to be unique, i.e., there can be only one socket named ⟨*name*⟩. This is ensured by declaring each socket with `\NewSocket`.

  However, there is no requirement that sockets and hook names have to be different. In fact, if a certain action that could otherwise be specified as hook code has to be executed always last (or first) one could ensure this by placing a socket (single action) after a hook (or vice versa) and using the same name to indicate the relationship, e.g.,

  ```
  \UseHook{foo}        % different package can add code here
  \UseSocket{foo}      % only one package can assign a plug
  ```

  This avoids the need to order the hook code to ensure that something is always last.
- Best practice naming conventions are ... *to be documented*

- A socket has documented inputs which are
  - the positional arguments (if any) with a description of what they contain when used
  - implicit data (registers and other 2e/expl3 data stores) that the socket is allowed to make use of, with a documented description of what they contain (if relevant for the task at hand—no need to describe the whole LaTeX universe)
  - information about the state of the TeX engine (again when relevant), e.g. is called in mmode or vmode or in the output routine or ...
  - ... anything missing?
- A socket has documented results/outputs which can be
  - what kind of data it should write to the current list (if that is part of its task)
  - what kind of registers and other 2e/expl3 data stores it should modify and in what way
  - what kind of state changes it should do (if any)
  - *... anything else?*
- At any time a socket has one block of code (a plug :-)) associated with it. Such code is itself named and the association is done by linking the socket name to the code name (putting a plug into the socket).
- The name of a plug consists of ASCII letters `[a-z]`, `[A-Z]`, `[0-9]`, `[-/@]` only.
- Socket plug names have to be unique within on a per socket basis, but it is perfectly allowed (and sensible in some cases) to use the same plug name with different sockets (where based on the sockets' purposes, different actions may be associated with the plug name). For example `noop` is a plug name declared for every socket, yet it action "grab the socket inputs and throw them away" obviously differs depending on how many inputs the socket has.
- When declaring a plug it is stated for which socket it is meant (i.e., its code can only be used with that socket). This means that the same plug name can be used with different sockets referring to different code in each case.
- Configuration of a socket can only be done by linking different code to it. Nevertheless the code linked to it can provide its own means of configuration (but this is outside of the spec).
- Technically execution of a socket (`\UseSocket`) involves
  - doing any house keeping (like writing debugging info, ...);
  - looking up the current code association (what plug is in the socket);
  - executing this code which will pick up the mandatory arguments (happens at this point, not before), i.e., it is like calling a csname defined with

    ```
    \def\foo#1#2...{...#1...#2...}
    ```
  - do some further house keeping (if needed).
- A socket is typically only used in one place in code, but this is not a requirement, i.e., if the same operation with the same inputs need to be carried out in several places the same named socket can be used.

2.3.3 Command syntax

We give both the LaTeX 2_ε and the L3 programming layer command names.

`\NewSocket` `\NewSocket {⟨socket-name⟩} {⟨number-of-inputs⟩}`
`\socket_new:nn` `\socket_new:nn {⟨socket-name⟩} {⟨number-of-inputs⟩}`

Declares a new socket with name ⟨*socket-name*⟩ having ⟨*number-of-inputs*⟩ inputs. There is automatically a plug `noop` declared for it, which does nothing, i.e., it gobbles the socket inputs (if any). This is made the default plug except for sockets with one input which additionally define the plug `identity` and assign that as their default.

This `identity` plug simply returns the socket input without its outer braces. The use case for this plug are situations like this:

```
\UseSocket{tagsupport/footnote}{⟨code⟩}
```

If tagging is not active and the socket contains the plug `identity` then this returns ⟨*code*⟩ without the outer braces and to activate tagging all that is necessary is to change the plug to say `tagpdf` so that it surrounds ⟨*code*⟩ by some tagging magic. This is the most common use case for sockets with one input, which is why they have this special default.

The socket documentation should describe its purpose, its inputs and the expected results as discussed above.

The declaration is only allowed at top-level, i.e., not inside a group.

`\NewSocketPlug` `\NewSocketPlug {⟨socket-name⟩} {⟨socket-plug-name⟩} {⟨code⟩}`
`\socket_new_plug:nnn` `\socket_new_plug:nnn {⟨socket-name⟩} {⟨socket-plug-name⟩} {⟨code⟩}`
`\socket_set_plug:nnn` `\socket_set_plug:nnn {⟨socket-name⟩} {⟨socket-plug-name⟩} {⟨code⟩}`

Declares a new plug for socket ⟨*socket-name*⟩ that runs ⟨*code*⟩ when executing. It complains if the plug was already declared previously.

The form `\socket_set_plug:nnn` changes an existing plug. As this should normally not be necessary, we currently have only an L3 layer name for the few cases it might be useful.

The declarations can be made inside a group and obey scope, i.e., they vanish if the group ends.

`\AssignSocketPlug` `\AssignSocketPlug {⟨socket-name⟩} {⟨socket-plug-name⟩}`
`\socket_assign_plug:nn` `\socket_assign_plug:nn {⟨socket-name⟩} {⟨socket-plug-name⟩}`

Assigns the plug ⟨*socket-plug-name*⟩ to the socket ⟨*socket-name*⟩. It errors if either socket or plug is not defined.

The assignment is local, i.e., it obeys scope.

\UseSocket
\socket_use:nw
\socket_use:n
\socket_use:nn
\socket_use:nnn
\socket_use:nnnn

\UseSocket {⟨socket-name⟩}
\socket_use:nnn {⟨socket-name⟩} {⟨socket-arg$_1$⟩} {⟨socket-arg$_2$⟩}

Executes the socket ⟨socket-name⟩ by retrieving the ⟨code⟩ of the current plug assigned to the socket. This is the only command that would appear inside macro code in packages.

For performance reasons there is no explicit check that the socket was declared!

The different L3 programming layer commands are really doing the same thing: they grab as many arguments as defined as inputs for the socket and then pass them to the plug. The different names are only there to make the code more readable, i.e., to indicate how many arguments are grabbed in total (note that no runtime check is made to verify that this is actually true). We only provide them for sockets with up to 3 inputs (most likely those with zero or one input would have been sufficient). If you happen to have a socket with more inputs, use \socket_use:nw.

\socket_use_expandable:nw ⋆
\socket_use_expandable:n ⋆

\socket_use_expandable:n {⟨socket-name⟩}

Fully expandable variant of \socket_use:n. This can be used in macro code to retrieve code from sockets which need to appear in an expandable context.

This usually requires the plug to only contain expandable code and should therefore only be used for sockets which are clearly documented to be used in an expandable context. This command does not print any debugging info when \DebugSocketsOn is active and should therefore be avoided whenever possible.

For performance reasons there is no explicit check that the socket was declared!

\ShowSocket
\LogSocket
\socket_show:n
\socket_log:n

\ShowSocket {⟨socket-name⟩}
\socket_show:n {⟨socket-name⟩}

Displays information about the socket ⟨socket-name⟩ and its state then stops and waits for further instructions — at the moment some what rudimentary.

\LogSocket and \socket_log:n only differ in that they don't stop.

\DebugSocketsOn
\DebugSocketsOff
\socket_debug_on:
\socket_debug_off:

\DebugSocketsOn ... \DebugSocketsOff

Turns debugging of sockets on or off.

2.3.4 Rationale for error handling

The errors during the declarations are produced to help with typos—after all, such declarations might be part of a document preamble (not that likely, but possible). However, \UseSocket is not doing much checking, e.g.,

```
\UseSocket{mispelled-socket}{hello}{world}
```

will generate a rather low-level error and then typesets "helloworld" because there is no dedicated runtime check if mispelled-socket is a known socket.

The reason is that if the misspelling is in the code, then this is a programming error in the package and for speed reasons LaTeX does not repeately make runtime checks for coding errors unless they can or are likely to be user introduced.

3 The Implementation

The implementation of the socket mechanism should be (partially) redone and we should probably store the different code chunks in a property list so that we can have a decent `\ShowSocket` command that shows the available alternatives.

TODO: *implement?*

```
⟨*2ekernel | latexrelease⟩
\ExplSyntaxOn
⟨@@=socket⟩
⟨latexrelease⟩\NewModuleRelease{2023/11/01}{ltsockets}
⟨latexrelease⟩                    {The~socket~management~system}
```

3.1 Debugging the socket structures

Code and commands in this section are not final, it needs more experimentation to see what kind of tracing information is going to be useful in practice. For now the tracing is mainly meant to be used for code testing and not so much for application testing.

It is quite likely that the commands and the behavior of the tracing might change in the future once we gained some experience with it.

`\g__socket_debug_bool` Holds the current debugging state.

```
\bool_new:N \g__socket_debug_bool
```

(*End of definition for* `\g__socket_debug_bool`.)

`\socket_debug_on:` `\socket_debug_off:` `\__socket_debug:n` `\__socket_debug_term:n` `\__socket_debug_gset:` Turns debugging on and off by redefining `\__socket_debug:n` and `\__socket_debug_term:n`. By default they do nothing.

```
\cs_new_eq:NN \__socket_debug:n \use_none:n
\cs_new_eq:NN \__socket_debug_term:n \use_none:n
\cs_new_protected:Npn \socket_debug_on:
  {
    \bool_gset_true:N \g__socket_debug_bool
    \__socket_debug_gset:
  }
\cs_new_protected:Npn \socket_debug_off:
  {
    \bool_gset_false:N \g__socket_debug_bool
    \__socket_debug_gset:
  }
\cs_new_protected:Npn \__socket_debug_gset:
  {
    \cs_gset_protected:Npx \__socket_debug:n ##1
      { \bool_if:NT \g__socket_debug_bool {##1} }
    \cs_gset_protected:Npx \__socket_debug_term:n ##1
      { \bool_if:NT \g__socket_debug_bool
        { \iow_term:x { ^^J [Sockets]~ ==>~ ##1} } }
  }
```

(*End of definition for* `\socket_debug_on:` *and others. These functions are documented on page* 335.)

3.2 The L3 layer commands

\socket_new:nn

Declaring a socket creates a str to hold the name (a pointer) to the code that should be used when the socket is executed, and an integer to hold the number of inputs of that socket. Initially, an "empty" code chunk is created and assigned so the socket does nothing by default other than swallowing its inputs (if any).

```
\cs_new_protected:Npn \socket_new:nn #1 #2 {
  \str_if_exist:cTF { l__socket_#1_plug_str }
      {
        \msg_error:nnn { socket } { already-declared } {#1}
      }
      {
```

We only support declarations on top-level.

```
        \int_if_zero:nTF \tex_currentgrouplevel:D
          {
            \str_new:c { l__socket_#1_plug_str }
            \seq_new:c { l__socket_#1_plugs_seq }
            \int_const:cn { c__socket_#1_args_int } {#2}
            \socket_new_plug:nnn {#1} { noop } {}
            \int_compare:nNnTF {#2} = 1
              {
                \socket_new_plug:nnn {#1} { identity } {##1}
                \socket_assign_plug:nn {#1} { identity }
              }
              { \socket_assign_plug:nn {#1} { noop } }
            \__socket_debug_term:n
              { Socket~ '#1'~ declared~ with~ #2~ input(s) }
          }
          {
            \msg_error:nn { socket } { not-top-level }
          }
      }
}
```

(*End of definition for* `\socket_new:nn`. *This function is documented on page* 334.)

\socket_log:n
\socket_show:n

Show the current state of the socket — for now this is just a quick draft and should be redone and extended.

```
\cs_new_protected:Npn \socket_log:n #1 {
  \typeout{ Socket~ #1:}
  \str_if_exist:cTF { l__socket_#1_plug_str }
      {
        \typeout{ \@spaces number~ of~ inputs~ =~
                  \int_use:c { c__socket_#1_args_int } }
        \typeout{ \@spaces available~plugs~ =~
                  \seq_use:cnnn { l__socket_#1_plugs_seq }{,~}{,~}{,~} }
        \typeout{ \@spaces current~ plug~ =~
                  \str_use:c { l__socket_#1_plug_str } }
        \typeout{ \@spaces definition~ =~
                  \cs_meaning:c
                    { __socket_#1_plug_ \str_use:c { l__socket_#1_plug_str } :w } }
        \typeout{}
      }
      {
```

If we are showing a socket it is not an error if it doesn't exist.

```
        \typeout { Socket~ is~ not~ declared! }
      }
}
```

And here the version that stops:

```
\cs_new_protected:Npn \socket_show:n #1 {\socket_log:n {#1} \errmessage{}}
```

(*End of definition for* `\socket_log:n` *and* `\socket_show:n`*. These functions are documented on page 335.*)

`\socket_new_plug:nnn`
`\socket_set_plug:nnn`

Declaring a code for a socket is just making a definition, taking the number of arguments from the saved int.

```
\cs_new_protected:Npn \socket_new_plug:nnn #1#2#3 {
  \str_if_exist:cTF { l__socket_#1_plug_str }
      {
        \cs_if_exist:cTF { __socket_#1_plug_#2:w }
          {
            \msg_error:nnnn { socket } { plug-already-declared } {#1} {#2}
          }
          {
            \cs_generate_from_arg_count:cNnn
              { __socket_#1_plug_#2:w }
              \cs_new:Npn
              { \int_use:c { c__socket_#1_args_int } }
              {#3}
```

This is a new declaration so we add the name to a seq for the debugging info.

```
            \seq_put_right:cn { l__socket_#1_plugs_seq } {#2}
            \__socket_debug_term:n
              { Plug~ '#2'~ for~ socket~ '#1'~ declared. }
          }
      }
      {
        \msg_error:nnn { socket } { undeclared } {#1}
      }
}
```

Changing the plug of an existing socket is rather similar, except that we don't have to deal with adding it to the debugging sequence.

```
\cs_new_protected:Npn \socket_set_plug:nnn #1#2#3 {
  \str_if_exist:cTF { l__socket_#1_plug_str }
      {
        \cs_if_exist:cTF { __socket_#1_plug_#2:w }
          {
            \cs_generate_from_arg_count:cNnn
              { __socket_#1_plug_#2:w }
              \cs_set:Npn
              { \int_use:c { c__socket_#1_args_int } }
              {#3}
            \__socket_debug_term:n
              {  Plug~ '#2'~ for~ socket~ '#1'~ changed. }
          }
          {
            \msg_error:nnnn { socket } { plug-undeclared } {#1} {#2}
```

```
          }
      }
      {
        \msg_error:nnn { socket } { undeclared } {#1}
      }
}
```

(*End of definition for* `\socket_new_plug:nnn` *and* `\socket_set_plug:nnn`*. These functions are documented on page 334.*)

`\socket_assign_plug:nn`

Assigning a plug to a socket just changes the name in the socket string. The assignment is local to the current group.

```
\cs_new_protected:Npn \socket_assign_plug:nn #1 #2 {
  \str_if_exist:cTF { l__socket_#1_plug_str }
      {
        \cs_if_exist:cTF { __socket_#1_plug_#2:w }
          {
            \__socket_debug_term:n
              { Replacing~ plug~ '\str_use:c { l__socket_#1_plug_str }'~
                with~ '#2'~ in~ socket~ '#1'. }
            \str_set:cn { l__socket_#1_plug_str } {#2}
          }
          {
            \msg_error:nnnn { socket } { plug-undeclared } {#1} {#2}
          }
      }
      { \msg_error:nnn { socket } { undeclared } {#1} }
}
```

(*End of definition for* `\socket_assign_plug:nn`*. This function is documented on page 334.*)

`\socket_use:nw` `\socket_use:n` `\socket_use:nn` `\socket_use:nnn` `\socket_use:nnnn`

And using it is more or less a `\use:c` so very lightweight. We do not add a runtime check for speed reasons!

This command is named `\socket_use:nw` because we don't know how many inputs the socket has until we have looked at the socket name (in argument `#1`). But, of course, the developer knows so we also offer a few aliases `\socket_use:nn`, etc. so that one can indicate the correct number of arguments (socket inputs plus one) in the L3 layer code.

```
\cs_new_protected:Npn \socket_use:nw #1 {
  \__socket_debug_term:n
    { Socket~ '#1'~ containing~ plug~
      '\str_use:c { l__socket_#1_plug_str }'~ used. }
  \use:c { __socket_#1_plug_ \str_use:c { l__socket_#1_plug_str } :w }
}
```

To make code a bit more readable we also define functions that indicate how many arguments are picked up. However, this is just for code documentation: internally they all do the same and the number of arguments isn't checked by default.

```
\cs_new_eq:NN  \socket_use:n \socket_use:nw     % socket with no inputs
\cs_new_eq:NN  \socket_use:nn \socket_use:nw    % socket with one input
\cs_new_eq:NN  \socket_use:nnn \socket_use:nw   % socket with two inputs
\cs_new_eq:NN  \socket_use:nnnn \socket_use:nw  % socket with three inputs
```

The above commands could be changed to check how many inputs the socket is declared with (for example, when checking is in force).

TODO: *Implement?*

(*End of definition for* `\socket_use:nw` *and others. These functions are documented on page* 335.)

\socket_use_expandable:nw
\socket_use_expandable:n

The same as the non-expandable code, except for the missing debug output.

```
\cs_new:Npn \socket_use_expandable:nw #1 {
  \use:c { __socket_#1_plug_ \str_use:c { l__socket_#1_plug_str } :w }
}
\cs_new_eq:NN  \socket_use_expandable:n \socket_use_expandable:nw     % socket with no inputs
```

(*End of definition for* `\socket_use_expandable:nw` *and* `\socket_use_expandable:n`*. These functions are documented on page* 335.)

3.3 Error messages

```
\msg_new:nnnn { socket } { already-declared }
    { Socket~ '#1'~ already~ declared! }
    { A~ socket~ can~ only~ be~ declared~ once.~ The~ name~ '#1'~ is~
      already~ taken.~ Use~ \ShowSocket{#1}~ to~ see~ its~ definition. }

\msg_new:nnnn { socket } { undeclared }
    { Socket~ '#1'~ undeclared! }
    { You~ tried~ to~ use~ a~ socket~ that~ was~ not~ declared~ before. }

\msg_new:nnnn { socket } { not-top-level }
    { Sockets~ can~ only~ be~ declared~ at~ top-level! }
    { It~ is~ not~ allowed~ to~ declare~ sockets~ inside~ a~
      group.~ Move~ the~ declaration~ to~ the~ top-level. }

\msg_new:nnnn { socket } { plug-already-declared }
    { Plug~ '#2'~ for~ socket~ '#1'~ already~ declared! }
    { You~ can't~ change~ an~ existing~ plug~ with~ \NewSocketPlug~ and~ it~
      is~ normally~ not~ sensible~ to~ do~ so.~ Use~ the~ L3~ programming~
      layer~ function~ \socket_set_plug:nnn~ if~ you~ really~ have~ to. }

\msg_new:nnnn { socket } { plug-undeclared }
    { Plug~ '#2'~ for~ socket~ '#1'~ undeclared! }
    { The~ plug~ name~ is~ unknown.~ Is~ the~ name~ misspelled~ or~ did~ you~
      intend~ to~ assign~ it~ to~ a~ different~ socket? }

\prop_gput:Nnn \g_msg_module_type_prop { socket } { LaTeX }
```

3.4 The LaTeX 2_ε interface commands

\NewSocket
\NewSocketPlug
\ShowSocket
\LogSocket
\AssignSocketPlug
\UseSocket
\DebugSocketsOn
\DebugSocketsOff

As we expect that there are existing LaTeX 2_ε packages that may want to make use of the socket mechanism, we provide 2e names for most of the commands.

```
\cs_new_eq:NN \NewSocket          \socket_new:nn
\cs_new_eq:NN \ShowSocket         \socket_show:n
\cs_new_eq:NN \LogSocket          \socket_log:n

\cs_new_eq:NN \NewSocketPlug      \socket_new_plug:nnn
\cs_new_eq:NN \AssignSocketPlug   \socket_assign_plug:nn
\cs_new_eq:NN \UseSocket          \socket_use:nw

\cs_new_eq:NN \DebugSocketsOn     \socket_debug_on:
\cs_new_eq:NN \DebugSocketsOff    \socket_debug_off:
```

(*End of definition for* `\NewSocket` *and others. These functions are documented on page* 334.)

```
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{ltsockets}
⟨latexrelease⟩                  {The~socket~management~(undo)}%
⟨latexrelease⟩
⟨latexrelease⟩\let \NewSocket  \@undefined
⟨latexrelease⟩\let \ShowSocket \@undefined
⟨latexrelease⟩\let \LogSocket  \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let \NewSocketPlug    \@undefined
⟨latexrelease⟩\let \AssignSocketPlug \@undefined
⟨latexrelease⟩\let \UseSocket        \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let \DebugSocketsOn  \@undefined
⟨latexrelease⟩\let \DebugSocketsOff \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndModuleRelease
```

```
\ExplSyntaxOff
```

```
⟨/2ekernel | latexrelease⟩
```

Reset module prefix:

```
⟨@@=⟩
```

File 11
lttemplates.dtx

1 Introduction

There are three broad "layers" between putting down ideas into a source file and ending up with a typeset document. These layers of document writing are

1. authoring of the text with mark-up;
2. document layout design;
3. implementation (with TEX programming) of the design.

We write the text as an author, and we see the visual output of the design after the document is generated; the TEX implementation in the middle is the glue between the two.

LATEX's greatest success has been to standardise a system of mark-up that balances the trade-off between ease of reading and ease of writing to suit almost all forms of technical writing. It's other original strength was a good background in typographical design; while the standard LATEX 2_ε classes look somewhat dated now in terms of their visual design, their typography is generally sound (barring the occasional minor faults).

However, LATEX 2_ε has always lacked a standard approach to customising the visual design of a document. Changing the looks of the standard classes involved either:

- Creating a new version of the implementation code of the class and editing it.
- Loading one of the many packages to customise certain elements of the standard classes.
- Loading a completely different document class, such as KOMA-Script or memoir, that allows easy customisation.

All three of these approaches have their drawbacks and learning curves.

The idea behind lttemplates is to cleanly separate the three layers introduced at the beginning of this section, so that document authors who are not programmers can easily change the design of their documents. lttemplates also makes it easier for LATEX programmers to provide their own customisations on top of a pre-existing class.

2 What is a document?

Besides the textual content of the words themselves, the source file of a document contains mark-up elements that add structure to the document. These elements include sectional divisions, figure/table captions, lists of various sorts, theorems/proofs, and so on. The list will be different for every document that can be written.

Each element can be represented logically without worrying about the formatting, with mark-up such as `\section`, `\caption`, `\begin{enumerate}` and so on. The output of each one of these document elements will be a typeset representation of the information marked up, and the visual arrangement and design of these elements can vary widely in producing a variety of desired outcomes.

For each type of document element, there may be design variations that contain the same sort of information but present it in slightly different ways. For example, the difference between a numbered and an unnumbered section, `\section` and `\section*`, or the difference between an itemised list or an enumerated list.

There are three distinct layers in the definition of "a document" at this level

1. semantic elements such as the ideas of sections and lists;
2. a set of design solutions for representing these elements visually;
3. specific variations for these designs that represent the elements in the document.

In the parlance of the template system, these are called types, templates, and instances, and they are discussed below in sections 4, 5, and 7, respectively.

3 Types, templates, and instances

By formally declaring documents to be composed of mark-up elements grouped into types, which are interpreted and typeset with a set of templates, each of which has one or more instances with which to compose each and every semantic unit of the text, we can cleanly separate the components of document construction.

All of the structures provided by the template system are global, and do not respect TEX grouping.

4 Template types

An *template type* (sometimes just "type") is an abstract idea of a document element that takes a fixed number of arguments corresponding to the information from the document author that it is representing. A sectioning type, for example, might take three inputs: "title", "short title", and "label".

Any given document class will define which types are to be used in the document, and any template of a given type can be used to generate an instance for the type. (Of course, different templates will produce different typeset representations, but the underlying content will be the same.)

`\NewTemplateType`

`\NewTemplateType {⟨template type⟩} {⟨no. of args⟩}`

This function defines an ⟨*template type*⟩ taking ⟨*number of arguments*⟩, where the ⟨*type*⟩ is an abstraction as discussed above. For example,

```
\NewTemplateType{sectioning}{3}
```

creates a type "sectioning", where each use of that type will need three arguments.

5 Templates

A *template* is a generalised design solution for representing the information of a specified type. Templates that do the same thing, but in different ways, are grouped together by their type and given separate names. There are two important parts to a template:

- the parameters it takes to vary the design it is producing;

Key-type	Description of input
`boolean`	`true` or `false`
`choice{`⟨*choices*⟩`}`	A list of pre-defined ⟨*choices*⟩
`commalist`	A comma-separated list
`function{`⟨N⟩`}`	A function definition with N arguments (N from 0 to 9)
`instance{`⟨*name*⟩`}`	An instance of type ⟨*name*⟩
`integer`	An integer or integer expression
`length`	A fixed length
`muskip`	A math length with shrink and stretch components
`real`	A real (floating point) value
`skip`	A length with shrink and stretch components
`tokenlist`	A token list: any text or commands

Table 1: Key-types for defining template interfaces with `\DeclareTemplateInterface`.

- the implementation of the design.

As a document author or designer does not care about the implementation but rather only the interface to the template, these two aspects of the template definition are split into two independent declarations, `\DeclareTemplateInterface` and `\DeclareTemplateCode`.

`\DeclareTemplateInterface`

```
\DeclareTemplateInterface
  {⟨type⟩} {⟨template⟩} {⟨no. of args⟩}
  {⟨key list⟩}
```

A ⟨*template*⟩ interface is declared for a particular ⟨*type*⟩, where the ⟨*number of arguments*⟩ must agree with the type declaration. The interface itself is defined by the ⟨*key list*⟩, which is itself a key–value list taking a specialized format:

```
⟨key1⟩ ":" ⟨key type1⟩ ","
⟨key2⟩ ":" ⟨key type2⟩ ","
⟨key3⟩ ":" ⟨key type3⟩ "=" ⟨default3⟩ ","
⟨key4⟩ ":" ⟨key type4⟩ "=" ⟨default4⟩ ","
...
```

Each ⟨*key*⟩ name should consist of ASCII characters, with the exception of `,`, `=` and ␣. The recommended form for key names is to use lower case letters, with dashes to separate out different parts. Spaces are ignored in key names, so they can be included or missed out at will. Each ⟨*key*⟩ must have a ⟨*key type*⟩, which defined the type of input that the ⟨*key*⟩ requires. A full list of key types is given in Table 1. Each key may have a ⟨*default*⟩ value, which will be used in by the template if the ⟨*key*⟩ is not set explicitly. The ⟨*default*⟩ should be of the correct form to be accepted by the ⟨*key type*⟩ of the ⟨*key*⟩: this is not checked by the code. Expressions for numerical values are evaluated when the template is used, thus for example values given in terms of `em` or `ex` will be set respecting the prevailing font.

\KeyValue `\KeyValue {⟨key name⟩}`

There are occasions where the default (or value) for one key should be taken from another. The `\KeyValue` function can be used to transfer this information without needing to know the internal implementation of the key:

```
\DeclareTemplateInterface { type } { template } { no. of args }
  {
    key-name-1 : key-type = value ,
    key-name-2 : key-type = \KeyValue { key-name-1 },
    ...
  }
```

\DeclareTemplateCode `\DeclareTemplateCode`
`{⟨type⟩} {⟨template⟩} {⟨no. of args⟩}`
`{⟨key bindings⟩} {⟨code⟩}`

The relationship between a templates keys and the internal implementation is created using the `\DeclareTemplateCode` function. As with `\DeclareTemplateInterface`, the ⟨*template*⟩ name is given along with the ⟨*type*⟩ and ⟨*number of arguments*⟩ required. The ⟨*key bindings*⟩ argument is a key–value list which specifies the relationship between each ⟨*key*⟩ of the template interface with an underlying⟨*variable*⟩.

⟨*key1*⟩ "=" ⟨*variable1*⟩,
⟨*key2*⟩ "=" ⟨*variable2*⟩,
⟨*key3*⟩ "=" global ⟨*variable3*⟩,
⟨*key4*⟩ "=" global ⟨*variable4*⟩,
...

With the exception of the choice, code and function key types, the ⟨*variable*⟩ here should be the name of an existing LaTeX3 register. As illustrated, the key word "global" may be included in the listing to indicate that the ⟨*variable*⟩ should be assigned globally. A full list of variable bindings is given in Table 2.

The ⟨*code*⟩ argument of `\DeclareTemplateCode` is used as the replacement text for the template when it is used, either directly or as an instance. This may therefore accept arguments `#1`, `#2`, *etc.* as detailed by the ⟨*number of arguments*⟩ taken by the type.

\AssignTemplateKeys `\AssignTemplateKeys`

In the final argument of `\DeclareTemplateCode` the assignment of keys defined by the template may be delayed by including the command `\AssignTemplateKeys`. If this is *not* present, keys are assigned immediately before the template code. If `\AssignTemplateKeys` is present, assignment is delayed until this point. Note that the command must be *directly* present in the code, not placed within a nested command/macro.

\DeclareTemplateCopy `\DeclareTemplateCopy`
`{⟨type⟩} {⟨template2⟩} {⟨template1⟩}`

Copies ⟨*template1*⟩ of ⟨*type*⟩ to a new name ⟨*template2*⟩: the copy can then be edited independent of teh original.

Key-type	Description of binding
	Boolean variable, *e.g.* `\l_tmpa_bool`
`choice`	List of choice implementations (see Section 6)
`commalist`	Comma list, *e.g.* `\l_tmpa_clist`
`function`	Function taking N arguments, *e.g.* `\use_i:nn`
`instance`	
`integer`	Integer variable, *e.g.* `\l_tmpa_int`
`length`	Dimension variable, *e.g.* `\l_tmpa_dim`
`muskip`	Muskip variable, *e.g.* `\l_tmpa_muskip`
`real`	Floating-point variable, *e.g.* `\l_tmpa_fp`
`skip`	Skip variable, *e.g.* `\l_tmpa_skip`
`tokenlist`	Token list variable, *e.g.* `\l_tmpa_tl`

Table 2: Bindings required for different key types when defining template implementations with `\DeclareTemplateCode`. Apart from `code`, `choice` and `function` all of these accept the key word `global` to carry out a global assignment.

6 Multiple choices

The `choice` key type implements multiple choice input. At the interface level, only the list of valid choices is needed:

```
\DeclareTemplateInterface { foo } { bar } { 0 }
  { key-name : choice { A, B, C } }
```

where the choices are given as a comma-list (which must therefore be wrapped in braces). A default value can also be given:

```
\DeclareTemplateInterface { foo } { bar } { 0 }
  { key-name : choice { A, B, C } = A }
```

At the implementation level, each choice is associated with code, using a nested key–value list.

```
\DeclareTemplateCode { foo } { bar } { 0 }
  {
    key-name =
      {
        A = Code-A ,
        B = Code-B ,
        C = Code-C
      }
   }
   { ... }
```

The two choice lists should match, but in the implementation a special `unknown` choice is also available. This can be used to ignore values and implement an "else" branch:

```
\DeclareTemplateCode { foo } { bar } { 0 }
  {
    key-name =
```

```
    {
      A       = Code-A ,
      B       = Code-B ,
      C       = Code-C ,
      unknown = Else-code
    }
  }
  { ... }
```

The `unknown` entry must be the last one given, and should *not* be listed in the interface part of the template.

For keys which accept the values `true` and `false` both the boolean and choice key types can be used. As template interfaces are intended to prompt clarity at the design level, the boolean key type should be favoured, with the choice type reserved for keys which take arbitrary values.

7 Instances

After a template is defined it still needs to be put to use. The parameters that it expects need to be defined before it can be used in a document. Every time a template has parameters given to it, an *instance* is created, and this is the code that ends up in the document to perform the typesetting of whatever pieces of information are input into it.

For example, a template might say "here is a section with or without a number that might be centred or left aligned and print its contents in a certain font of a certain size, with a bit of a gap before and after it" whereas an instance declares "this is a section with a number, which is centred and set in 12 pt italic with a 10 pt skip before and a 12 pt skip after it". Therefore, an instance is just a frozen version of a template with specific settings as chosen by the designer.

`\DeclareInstance`

`\DeclareInstance`
`{`⟨*type*⟩`}` `{`⟨*instance*⟩`}` `{`⟨*template*⟩`}` `{`⟨*parameters*⟩`}`

This function uses a ⟨*template*⟩ for an ⟨*type*⟩ to create an ⟨*instance*⟩. The ⟨*instance*⟩ will be set up using the ⟨*parameters*⟩, which will set some of the ⟨*keys*⟩ in the ⟨*template*⟩.

As a practical example, consider a type for document sections (which might include chapters, parts, sections, *etc.*), which is called `sectioning`. One possible template for this type might be called `basic`, and one instance of this template would be a numbered section. The instance declaration might read:

```
\DeclareInstance { sectioning } { section-num } { basic }
  {
    numbered      = true ,
    justification = center ,
    font          =\normalsize\itshape ,
    before-skip   = 10pt ,
    after-skip    = 12pt ,
  }
```

Of course, the key names here are entirely imaginary, but illustrate the general idea of fixing some settings.

\IfInstanceExistsT
\IfInstanceExistsF
\IfInstanceExistsTF

\IfInstanceExistsTF {⟨*type*⟩} {⟨*instance*⟩} {⟨*true code*⟩} {⟨*false code*⟩}

Tests if the named ⟨*instance*⟩ of a ⟨*type*⟩ exists, and then inserts the appropriate code into the input stream.

\DeclareInstanceCopy

\DeclareInstanceCopy
{⟨*type*⟩} {⟨*instance2*⟩} {⟨*instance1*⟩}

Copies the ⟨*values*⟩ for ⟨*instance1*⟩ for an ⟨*type*⟩ to ⟨*instance2*⟩.

8 Document interface

After the instances have been chosen, document commands must be declared to use those instances in the document. \UseInstance calls instances directly, and this command should be used internally in document-level mark-up.

\UseInstance

\UseInstance
{⟨*type*⟩} {⟨*instance*⟩} ⟨*arguments*⟩

Uses an ⟨*instance*⟩ of the ⟨*type*⟩, which will require ⟨*arguments*⟩ as determined by the number specified for the ⟨*type*⟩. The ⟨*instance*⟩ must have been declared before it can be used, otherwise an error is raised.

\UseTemplate

\UseTemplate {⟨*type*⟩} {⟨*template*⟩}
{⟨*settings*⟩} ⟨*arguments*⟩

Uses the ⟨*template*⟩ of the specified ⟨*type*⟩, applying the ⟨*settings*⟩ and absorbing ⟨*arguments*⟩ as detailed by the ⟨*type*⟩ declaration. This in effect is the same as creating an instance using \DeclareInstance and immediately using it with \UseInstance, but without the instance having any further existence. It is therefore useful where a template needs to be used once.

This function can also be used as the argument to instance key types:

```
\DeclareInstance { type } { template } { instance }
  {
    instance-key =
      \UseTemplate { type2 } { template2 } { <settings> }
  }
```

9 Changing existing definitions

Template parameters may be assigned specific defaults for instances to use if the instance declaration doesn't explicit set those parameters. In some cases, the document designer will wish to edit these defaults to allow them to "cascade" to the instances. The alternative would be to set each parameter identically for each instance declaration, a tedious and error-prone process.

\EditTemplateDefaults

\EditTemplateDefaults
{⟨*type*⟩} {⟨*template*⟩} {⟨*new defaults*⟩}

Edits the ⟨*defaults*⟩ for a ⟨*template*⟩ for an ⟨*type*⟩. The ⟨*new defaults*⟩, given as a key–value list, replace the existing defaults for the ⟨*template*⟩. This means that the change will apply to instances declared after the editing, but that instances which have already been created are unaffected.

\EditInstance

\EditInstance
{⟨*type*⟩} {⟨*instance*⟩} {⟨*new values*⟩}

Edits the ⟨*values*⟩ for an ⟨*instance*⟩ for an ⟨*type*⟩. The ⟨*new values*⟩, given as a key–value list, replace the existing values for the ⟨*instance*⟩. This function is complementary to \EditTemplateDefaults: \EditInstance changes a single instance while leaving the template untouched.

10 *Ad hoc* adjustment of templates

\SetTemplateKeys

\SetTemplateKeys {⟨*type*⟩} {⟨*template*⟩} {⟨*keyvals*⟩}

At point of use it may be useful to apply changed to individual instances. This is supported as each template key is made available for adjustment using \SetTemplateKeys.

For example, after

```
\NewTypeType{MyObj}{0}
\DeclareTemplateInterface{MyObj}{TemplateA}{0}
  {
    akey: tokenlist  ,
    bkey: function{2}
  }
\DeclareTemplateCode{MyObj}{TemplateA}{0}
  {
    akey = SomeTokens ,
    bkey = \func:nn ,
  }
```

the template keys could be adjusted in an *ad hoc* fashion using

```
\SetTemplateKeys{MyObj}{TemplateA}
  {
    akey = OtherTokens ,
    bkey = \AltFunc:nn
  }
```

11 Getting information about templates and instances

\ShowInstanceValues

\ShowInstanceValues {⟨*type*⟩} {⟨*instance*⟩}

Shows the ⟨*values*⟩ for an ⟨*instance*⟩ of the given ⟨*type*⟩ at the terminal.

\ShowTemplateCode `\ShowTemplateCode {⟨type⟩} {⟨template⟩}`

Shows the ⟨*code*⟩ of a ⟨*template*⟩ for an ⟨*type*⟩ in the terminal.

\ShowTemplateDefaults `\ShowTemplateDefaults {⟨type⟩} {⟨template⟩}`

Shows the ⟨*default*⟩ values of a ⟨*template*⟩ for an ⟨*type*⟩ in the terminal.

\ShowTemplateInterface `\ShowTemplateInterface {⟨type⟩} {⟨template⟩}`

Shows the ⟨*keys*⟩ and associated ⟨*key types*⟩ of a ⟨*template*⟩ for an ⟨*type*⟩ in the terminal.

\ShowTemplateVariables `\ShowTemplateVariables {⟨type⟩} {⟨template⟩}`

Shows the ⟨*variables*⟩ and associated ⟨*keys*⟩ of a ⟨*template*⟩ for an ⟨*type*⟩ in the terminal. Note that `code` and `choice` keys do not map directly to variables but to arbitrary code. For `choice` keys, each valid choice is shown as a separate entry in the list, with the key name and choice separated by a space, for example

```
Template 'example' of type 'example' has variable mapping:
>  demo unknown  =>  \def \demo {?}
>  demo c  =>  \def \demo {c}
>  demo b  =>  \def \demo {b}
>  demo a  =>  \def \demo {a}.
```

would be shown for a choice key `demo` with valid choices `a`, `b` and `c`, plus code for an `unknown` branch.

12 The implementation

```
⟨@@=template⟩
⟨*2ekernel⟩
\message{templates,}
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
\ExplSyntaxOn
⟨latexrelease⟩\NewModuleRelease{2024/06/01}{lttemplates}
⟨latexrelease⟩                    {Prototype~document~commands}%
```

12.1 Variables and constants

\c__template_code_root_tl
\c__template_defaults_root_tl
\c__template_instances_root_tl
\c__template_keytypes_root_tl
\c__template_key_order_root_tl
\c__template_restrict_root_tl
\c__template_values_root_tl
\c__template_vars_root_tl

So that literal values are kept to a minimum.

```
\tl_const:Nn \c__template_code_root_tl      { template~code~>~ }
\tl_const:Nn \c__template_defaults_root_tl  { template~defaults~>~ }
\tl_const:Nn \c__template_instances_root_tl { template~instance~>~  }
\tl_const:Nn \c__template_keytypes_root_tl  { template~key~types~>~ }
\tl_const:Nn \c__template_key_order_root_tl { template~key~order~>~ }
\tl_const:Nn \c__template_values_root_tl    { template~values~>~ }
\tl_const:Nn \c__template_vars_root_tl      { template~vars~>~ }
```

\c__template_keytypes_arg_seq

A list of keytypes which also need additional data (an argument), used to parse the keytype correctly.

```
\seq_const_from_clist:Nn \c__template_keytypes_arg_seq
  { choice , function , instance }
```

\g__template_type_prop

For storing types and the associated number of arguments.

```
\prop_new:N \g__template_type_prop
```

\l__template_assignments_tl

When creating an instance, the assigned values are collected here.

```
\tl_new:N \l__template_assignments_tl
```

\l__template_default_tl

The default value for a key is recovered here from the property list in which it is stored.

```
\tl_new:N \l__template_default_tl
```

\l__template_error_bool

A flag for errors to be carried forward.

```
\bool_new:N \l__template_error_bool
```

\l__template_global_bool

Used to indicate that assignments should be global.

```
\bool_new:N \l__template_global_bool
```

\l__template_key_name_tl
\l__template_keytype_tl
\l__template_keytype_arg_tl
\l__template_value_tl
\l__template_var_tl

When defining each key in a template, the name and type of the key need to be separated and stored. Any argument needed by the keytype is also stored separately.

```
\tl_new:N \l__template_key_name_tl
\tl_new:N \l__template_keytype_tl
\tl_new:N \l__template_keytype_arg_tl
\tl_new:N \l__template_value_tl
\tl_new:N \l__template_var_tl
```

\l__template_keytypes_prop
\l__template_key_order_seq
\l__template_values_prop
\l__template_vars_prop

To avoid needing too many difficult-to-follow csname assignments, various scratch token registers are used to build up data, which is then transferred

```
\prop_new:N \l__template_keytypes_prop
\seq_new:N \l__template_key_order_seq
\prop_new:N \l__template_values_prop
\prop_new:N \l__template_vars_prop
```

\l__template_tmp_clist
\l__template_tmp_dim
\l__template_tmp_int
\l__template_tmp_muskip
\l__template_tmp_skip
\l__template_tmp_tl

Scratch space.

```
\clist_new:N \l__template_tmp_clist
\dim_new:N \l__template_tmp_dim
\int_new:N \l__template_tmp_int
\muskip_new:N \l__template_tmp_muskip
\skip_new:N \l__template_tmp_skip
\tl_new:N \l__template_tmp_tl
```

\s__template_mark
\s__template_stop

Internal scan marks.

```
\scan_new:N \s__template_mark
\scan_new:N \s__template_stop
```

\q__template_nil

Internal quarks.

```
\quark_new:N \q__template_nil
```

__template_quark_if_nil_p:n
__template_quark_if_nil:n*TF*

Branching quark conditional.

```
\__kernel_quark_new_conditional:Nn \__template_quark_if_nil:N { F }
```

(*End of definition for* __template_quark_if_nil:nTF.)

12.2 Testing existence and validity

There are a number of checks needed for either the existence of a type, template or instance. There are also some for the validity of a particular call. All of these are collected up here.

__template_execute_if_arg_agree:nnT

A test agreement between the number of arguments for the template type and that specified when creating a template. This is not done as a separate conditional for efficiency and better error message

```
\cs_new_protected:Npn \__template_execute_if_arg_agree:nnT #1#2#3
  {
    \prop_get:NnN \g__template_type_prop {#1} \l__template_tmp_tl
    \int_compare:nNnTF {#2} = \l__template_tmp_tl
       {#3}
       {
         \msg_error:nneee { template } { argument-number-mismatch }
           {#1} { \l__template_tmp_tl } {#2}
       }
  }
```

(*End of definition for* `\__template_execute_if_arg_agree:nnT`.)

__template_execute_if_code_exist:nnT

A template is only fully declared if the code has been set up, which can be checked by looking for the template function itself.

```
\cs_new_protected:Npn \__template_execute_if_code_exist:nnT #1#2#3
  {
    \cs_if_exist:cTF { \c__template_code_root_tl #1 / #2 }
      {#3}
      { \msg_error:nnnn { template } { no-template-code } {#1} {#2} }
  }
```

(*End of definition for* `\__template_execute_if_code_exist:nnT`.)

__template_execute_if_keytype_exist:nT
__template_execute_if_keytype_exist:VT

The test for valid keytypes looks for a function to set up the key, which is part of the "code" side of the template definition. This avoids having different lists for the two parts of the process.

```
\cs_new_protected:Npn \__template_execute_if_keytype_exist:nT #1#2
  {
    \cs_if_exist:cTF { __template_store_value_ #1 :n }
      {#2}
      { \msg_error:nnn { template } { unknown-keytype } {#1} }
  }
\cs_generate_variant:Nn \__template_execute_if_keytype_exist:nT { V }
```

(*End of definition for* `\__template_execute_if_keytype_exist:nT`.)

__template_execute_if_type_exist:nT

To check that a particular type is valid.

```
\cs_new_protected:Npn \__template_execute_if_type_exist:nT #1#2
  {
    \prop_if_in:NnTF \g__template_type_prop {#1}
      {#2}
      { \msg_error:nnn { template } { unknown-type } {#1} }
  }
```

(*End of definition for* `\__template_execute_if_type_exist:nT`.)

__template_execute_if_keys_exist:nnT

To check that the keys for a template have been set up before trying to create any code, a simple check for the correctly-named keytype property list.

```
\cs_new_protected:Npn \__template_if_keys_exist:nnT #1#2#3
  {
    \cs_if_exist:cTF { \c__template_keytypes_root_tl #1 / #2 }
      {#3}
      { \msg_error:nnnn { template } { unknown-template } {#1} {#2} }
   }
```

(*End of definition for* __template_execute_if_keys_exist:nnT.)

__template_if_key_value:n*TF*
__template_if_key_value:V*TF*

Tests for the first token in a string being \KeyValue.

```
\prg_new_conditional:Npnn \__template_if_key_value:n #1 { T , F , TF }
  {
    \str_if_eq:noTF { \KeyValue } { \tl_head:w #1 \q_nil \q_stop }
      \prg_return_true:
      \prg_return_false:
  }
\prg_generate_conditional_variant:Nnn \__template_if_key_value:n { V } { T , F , TF }
```

(*End of definition for* __template_if_key_value:nTF.)

__template_if_instance_exist:nn*TF*

Testing for an instance

```
\prg_new_conditional:Npnn \__template_if_instance_exist:nn #1#2 { T, F, TF }
  {
    \cs_if_exist:cTF { \c__template_instances_root_tl #1 / #2 }
      \prg_return_true:
      \prg_return_false:
 }
```

(*End of definition for* __template_if_instance_exist:nnTF.)

__template_if_use_template:nTF

Tests for the first token in a string being \UseTemplate.

```
\prg_new_conditional:Npnn \__template_if_use_template:n #1 { TF }
  {
    \str_if_eq:noTF { \UseTemplate } { \tl_head:w #1 \q_nil \q_stop }
      \prg_return_true:
      \prg_return_false:
}
```

(*End of definition for* __template_if_use_template:nTF.)

12.3 Saving and recovering property lists

The various property lists for templates have to be shuffled in and out of storage.

__template_store_defaults:nn
__template_store_keytypes:nn
__template_store_values:nn
__template_store_vars:nn

The defaults and keytypes are transferred from the scratch property lists to the "proper" lists for the template being created.

```
\cs_new_protected:Npn \__template_store_defaults:nn #1#2
  {
    \debug_suspend:
    \prop_gclear_new:c { \c__template_defaults_root_tl #1 / #2  }
    \prop_gset_eq:cN { \c__template_defaults_root_tl #1 / #2 }
      \l__template_values_prop
```

```
    \debug_resume:
  }
\cs_new_protected:Npn \__template_store_keytypes:nn #1#2
  {
    \debug_suspend:
    \prop_if_exist:cTF { \c__template_keytypes_root_tl #1 / #2 }
      {
        \msg_info:nnnn { template } { declare-template-interface } {#1} {#2}
        \prop_gclear:c { \c__template_keytypes_root_tl #1 / #2 }
      }
      { \prop_new:c { \c__template_keytypes_root_tl #1 / #2 } }
    \prop_gset_eq:cN { \c__template_keytypes_root_tl #1 / #2 }
      \l__template_keytypes_prop
    \seq_gclear_new:c { \c__template_key_order_root_tl #1 / #2 }
    \seq_gset_eq:cN { \c__template_key_order_root_tl #1 / #2 }
      \l__template_key_order_seq
    \debug_resume:
  }
\cs_new_protected:Npn \__template_store_values:nn #1#2
  {
    \debug_suspend:
    \prop_clear_new:c { \c__template_values_root_tl #1 / #2 }
    \prop_set_eq:cN { \c__template_values_root_tl #1 / #2 }
      \l__template_values_prop
    \debug_resume:
  }
\cs_new_protected:Npn \__template_store_vars:nn #1#2
  {
   \debug_suspend:
    \prop_gclear_new:c { \c__template_vars_root_tl #1 / #2 }
    \prop_gset_eq:cN { \c__template_vars_root_tl #1 / #2 }
      \l__template_vars_prop
    \debug_resume:
  }
```

(*End of definition for* `\__template_store_defaults:nn` *and others.*)

`\__template_recover_defaults:nn`
`\__template_recover_keytypes:nn`
`\__template_recover_values:nn`
`\__template_recover_vars:nn`

Recovering the stored data for a template is rather less complex than storing it. All that happens is the data is transferred from the permanent to the scratch storage. However, we need to check the scratch storage does exist.

```
\cs_new_protected:Npn \__template_recover_defaults:nn #1#2
  {
    \prop_if_exist:cTF
      { \c__template_defaults_root_tl #1 / #2 }
      {
        \prop_set_eq:Nc \l__template_values_prop
          { \c__template_defaults_root_tl #1 / #2 }
      }
      { \prop_clear:N \l__template_values_prop }
  }
\cs_new_protected:Npn \__template_recover_keytypes:nn #1#2
  {
    \prop_if_exist:cTF
      { \c__template_keytypes_root_tl #1 / #2 }
```

```
      {
        \prop_set_eq:Nc \l__template_keytypes_prop
          { \c__template_keytypes_root_tl #1 / #2 }
      }
      { \prop_clear:N \l__template_keytypes_prop }
    \seq_if_exist:cTF { \c__template_key_order_root_tl #1 / #2 }
      {
        \seq_set_eq:Nc \l__template_key_order_seq
          { \c__template_key_order_root_tl #1 / #2 }
      }
      { \seq_clear:N \l__template_key_order_seq }
  }
\cs_new_protected:Npn \__template_recover_values:nn #1#2
  {
    \prop_if_exist:cTF
      { \c__template_values_root_tl #1 / #2 }
      {
        \prop_set_eq:Nc \l__template_values_prop
          { \c__template_values_root_tl #1 / #2 }
      }
      { \prop_clear:N \l__template_values_prop }
  }
\cs_new_protected:Npn \__template_recover_vars:nn #1#2
  {
    \prop_if_exist:cTF
      { \c__template_vars_root_tl #1 / #2 }
      {
        \prop_set_eq:Nc \l__template_vars_prop
          { \c__template_vars_root_tl #1 / #2 }
      }
      { \prop_clear:N \l__template_vars_prop }
  }
```

(*End of definition for* `\__template_recover_defaults:nn` *and others.*)

12.4 Creating new template types

`\__template_define_type:nn`
`\__template_declare_type:nn`

Although the type is the "top level" of the template system, it is actually very easy to implement. All that happens is that the number of arguments required is recorded, indexed by the name of the type.

```
\cs_new_protected:Npn \__template_define_type:nn #1#2
  {
    \prop_if_in:NnTF \g__template_type_prop {#1}
      { \msg_error:nnn { template } { type-already-defined } {#1} }
      { \__template_declare_type:nn {#1} {#2} }
  }
\cs_new_protected:Npn \__template_declare_type:nn #1#2
  {
    \int_set:Nn \l__template_tmp_int {#2}
    \int_compare:nTF { 0 <= \l__template_tmp_int <= 9 }
      {
        \msg_info:nnnV { template } { declare-type }
          {#1} \l__template_tmp_int
        \prop_gput:NnV \g__template_type_prop {#1}
```

```
          \l__template_tmp_int
      }
      {
        \msg_error:nnnV { template } { bad-number-of-arguments }
          {#1} \l__template_tmp_int
      }
  }
```

(*End of definition for* `\__template_define_type:nn` *and* `\__template_declare_type:nn`.)

12.5 Design part of template declaration

The "design" part of a template declaration defines the general behaviour of each key, and possibly a default value. However, it does not include the implementation. This means that what happens here is the two properties are saved to appropriate lists, which can then be used later to recover the information when implementing the keys.

`\__template_declare_template_keys:nnnn`

The main function for the "design" part of creating a template starts by checking that the type exists and that the number of arguments required agree. If that is all fine, then the two storage areas for defaults and keytypes are initialised. The mechanism is then set up for the l3keys module to actually parse the keys. Finally, the code hands of to the storage routine to save the parsed information properly.

```
\cs_new_protected:Npn \__template_declare_template_keys:nnnn #1#2#3#4
  {
    \__template_execute_if_type_exist:nT {#1}
      {
        \__template_execute_if_arg_agree:nnT {#1} {#3}
          {
            \prop_clear:N \l__template_values_prop
            \prop_clear:N \l__template_keytypes_prop
            \seq_clear:N \l__template_key_order_seq
            \keyval_parse:NNn
              \__template_parse_keys_elt:n \__template_parse_keys_elt:nn {#4}
            \__template_store_defaults:nn {#1} {#2}
            \__template_store_keytypes:nn {#1} {#2}
          }
      }
  }
```

(*End of definition for* `\__template_declare_template_keys:nnnn`.)

`\__template_parse_keys_elt:n`
`\__template_parse_keys_elt_aux:n`
`\__template_parse_keys_elt_aux:`

Processing the key part of the key–value pair is always carried out using this function, even if a value was found. First, the key name is separated from the keytype, and if necessary the keytype is separated into two parts. This information is then used to check that the keytype is valid, before storing the keytype (plus argument if necessary) as a property of the key name. The key name is also stored (in braces) in the token list to record the order the keys are defined in.

```
\cs_new_protected:Npn \__template_parse_keys_elt:n #1
  {
    \__template_split_keytype:n {#1}
    \bool_if:NF \l__template_error_bool
      {
        \__template_execute_if_keytype_exist:VT \l__template_keytype_tl
```

```
          {
            \seq_map_function:NN \c__template_keytypes_arg_seq
              \__template_parse_keys_elt_aux:n
            \bool_if:NF \l__template_error_bool
              {
                \seq_if_in:NoTF \l__template_key_order_seq
                  \l__template_key_name_tl
                  {
                    \msg_error:nnV { template } { duplicate-key-interface }
                      \l__template_key_name_tl
                  }
                  { \__template_parse_keys_elt_aux: }
              }
          }
      }
  }
\cs_new_protected:Npn \__template_parse_keys_elt_aux:n #1
  {
    \str_if_eq:VnT \l__template_keytype_tl {#1}
      {
        \tl_if_empty:NT \l__template_keytype_arg_tl
          {
            \msg_error:nnn { template } { keytype-requires-argument } {#1}
            \bool_set_true:N \l__template_error_bool
            \seq_map_break:
          }
      }
  }
\cs_new_protected:Npn \__template_parse_keys_elt_aux:
  {
    \tl_set:Ne \l__template_tmp_tl
      {
        \l__template_keytype_tl
        \tl_if_empty:NF \l__template_keytype_arg_tl
          { { \l__template_keytype_arg_tl } }
      }
    \prop_put:NVV \l__template_keytypes_prop \l__template_key_name_tl
      \l__template_tmp_tl
    \seq_put_right:NV \l__template_key_order_seq \l__template_key_name_tl
    \str_if_eq:VnT \l__template_keytype_tl { choice }
      {
        \clist_if_in:NnT \l__template_keytype_arg_tl { unknown }
          { \msg_error:nn { template } { choice-unknown-reserved } }
      }
  }
```

(*End of definition for* `\__template_parse_keys_elt:n`, `\__template_parse_keys_elt_aux:n`, *and* `\__template_parse_keys_elt_aux:`.)

`\__template_parse_keys_elt:nn` For keys which have a default, the keytype and key name are first separated out by the `\__template_parse_keys_elt:n` routine, before storing the default value in the scratch property list.

```
\cs_new_protected:Npn \__template_parse_keys_elt:nn #1#2
  {
```

```
    \__template_parse_keys_elt:n {#1}
    \use:c { __template_store_value_ \l__template_keytype_tl :n } {#2}
  }
```

(*End of definition for* `\__template_parse_keys_elt:nn`.)

`\__template_split_keytype:n`
`\__template_split_keytype_aux:w`

The keytype and key name should be separated by `:`. As the definition might be given inside or outside of a code block, the category code of colons is standardised. After that, the standard delimited argument method is used to separate the two parts.

```
\cs_new_protected:Npe \__template_split_keytype:n #1
  {
    \exp_not:N \bool_set_false:N \exp_not:N \l__template_error_bool
    \tl_set:Nn \exp_not:N \l__template_tmp_tl {#1}
    \tl_replace_all:Nnn \exp_not:N \l__template_tmp_tl { : } { \token_to_str:N : }
    \tl_if_in:VnTF \exp_not:N \l__template_tmp_tl { \token_to_str:N : }
      {
        \exp_not:n
          {
            \tl_clear:N \l__template_key_name_tl
            \exp_after:wN \__template_split_keytype_aux:w
              \l__template_tmp_tl \s__template_stop
          }
      }
      {
        \exp_not:N \bool_set_true:N \exp_not:N \l__template_error_bool
        \msg_error:nnn { template } { missing-keytype } {#1}
      }
  }
\use:e
  {
    \cs_new_protected:Npn \exp_not:N \__template_split_keytype_aux:w
      #1 \token_to_str:N : #2 \s__template_stop
      {
        \tl_put_right:Ne \exp_not:N \l__template_key_name_tl
          {
            \exp_not:N \tl_trim_spaces:e
              { \exp_not:N \tl_to_str:n {#1} }
          }
        \tl_if_in:nnTF {#2} { \token_to_str:N : }
          {
            \tl_put_right:Nn \exp_not:N \l__template_key_name_tl
              { \token_to_str:N : }
            \exp_not:N \__template_split_keytype_aux:w #2 \s__template_stop
          }
          {
            \exp_not:N \tl_if_empty:NTF \exp_not:N \l__template_key_name_tl
              {
                \msg_error:nnn { template } { empty-key-name }
                  { \token_to_str:N : #2 }
              }
              { \exp_not:N \__template_split_keytype_arg:n {#2} }
          }
      }
  }
```

(*End of definition for* `\__template_split_keytype:n` *and* `\__template_split_keytype_aux:w`.)

`\__template_split_keytype_arg:n`
`\__template_split_keytype_arg:V`
`\__template_split_keytype_arg_aux:n`
`\__template_split_keytype_arg_aux:w`

The second stage of sorting out the keytype is to check for an argument. As there is no convenient delimiting token to look for, a check is made instead for each possible text value for the keytype. To keep things faster, this only involves the keytypes that need an argument. If a match is made, then a check is also needed to see that it is at the start of the keytype information. All being well, the split can then be applied. Any non-matching keytypes are assumed to be "correct" as given, and are left alone (this is checked by other code).

```
\cs_new_protected:Npn \__template_split_keytype_arg:n #1
  {
    \tl_set:Ne \l__template_keytype_tl { \tl_trim_spaces:n {#1} }
    \tl_clear:N \l__template_keytype_arg_tl
    \cs_set_protected:Npn \__template_split_keytype_arg_aux:n ##1
      {
        \tl_if_in:nnT {#1} {##1}
          {
            \cs_set:Npn \__template_split_keytype_arg_aux:w
              ####1 ##1 ####2 \s__template_stop
              {
                \tl_if_blank:nT {####1}
                  {
                    \tl_set:Ne \l__template_keytype_tl
                      { \tl_trim_spaces:n {##1} }
                    \tl_if_blank:nF {####2}
                      {
                        \tl_set:Ne \l__template_keytype_arg_tl
                          { \use:n ####2 }
                      }
                    \seq_map_break:
                  }
              }
            \__template_split_keytype_arg_aux:w #1 \s__template_stop
          }
      }
    \seq_map_function:NN \c__template_keytypes_arg_seq
      \__template_split_keytype_arg_aux:n
  }
\cs_generate_variant:Nn \__template_split_keytype_arg:n { V }
\cs_new:Npn \__template_split_keytype_arg_aux:n #1 { }
\cs_new:Npn \__template_split_keytype_arg_aux:w #1 \s__template_stop { }
```

(*End of definition for* `\__template_split_keytype_arg:n`, `\__template_split_keytype_arg_aux:n`, *and* `\__template_split_keytype_arg_aux:w`.)

12.5.1 Storing values

As lttemplates pre-processes key values for efficiency reasons, there is a need to convert the values given as defaults into "ready to use" data. The same general idea is true when an instance is declared. However, assignments are not made until an instance is used, and so there has to be some intermediate storage. Furthermore, the ability to delay evaluation of results is needed. To achieve these aims, a series of "process and store" functions are defined here.

All of the information about the key (the key name and the keytype) is already stored as variables. The same property list is always used to store the data, meaning that the only argument required is the value to be processed and potentially stored.

__template_store_value_boolean:n

```
\cs_new_protected:Npn \__template_store_value_boolean:n #1
  { \prop_put:Non \l__template_values_prop \l__template_key_name_tl {#1} }
```

(*End of definition for* `\__template_store_value_boolean:n`.)

__template_store_value:n
__template_store_value_choice:n
__template_store_value_function:n
__template_store_value_instance:n

With no need to worry about delayed evaluation, these keytypes all just store the input directly.

```
\cs_new_protected:Npn \__template_store_value:n #1
  { \prop_put:Non \l__template_values_prop \l__template_key_name_tl {#1} }
\cs_new_eq:NN \__template_store_value_choice:n    \__template_store_value:n
\cs_new_eq:NN \__template_store_value_function:n  \__template_store_value:n
\cs_new_eq:NN \__template_store_value_instance:n  \__template_store_value:n
```

(*End of definition for* `\__template_store_value:n` *and others.*)

__template_store_value_aux:Nn
__template_store_value_integer:n
__template_store_value_length:n
__template_store_value_muskip:n
__template_store_value_real:n
__template_store_value_skip:n
__template_store_value_tokenlist:n
__template_store_value_commalist:n

Storing values in `\l__template_values_prop` is in most cases the same.

```
\cs_new_protected:Npn \__template_store_value_aux:Nn #1#2
  { \prop_put:Non \l__template_values_prop \l__template_key_name_tl {#2} }
\cs_new_protected:Npn \__template_store_value_integer:n
  { \__template_store_value_aux:Nn \int_eval:n }
\cs_new_protected:Npn \__template_store_value_length:n
  { \__template_store_value_aux:Nn \dim_eval:n }
\cs_new_protected:Npn \__template_store_value_muskip:n
  { \__template_store_value_aux:Nn \muskip_eval:n }
\cs_new_protected:Npn \__template_store_value_real:n
  { \__template_store_value_aux:Nn \fp_eval:n }
\cs_new_protected:Npn \__template_store_value_skip:n
  { \__template_store_value_aux:Nn \skip_eval:n }
\cs_new_protected:Npn \__template_store_value_tokenlist:n
  { \__template_store_value_aux:Nn \use:n }
\cs_new_eq:NN \__template_store_value_commalist:n \__template_store_value_tokenlist:n
```

(*End of definition for* `\__template_store_value_aux:Nn` *and others.*)

12.6 Implementation part of template declaration

__template_declare_template_code:nnnnn
__template_declare_template_code:nnnn

The main function for implementing a template starts with a couple of simple checks to make sure that there are no obvious mistakes: the number of arguments must agree and the template keys must have been declared.

```
\cs_new_protected:Npn \__template_declare_template_code:nnnnn #1#2#3#4#5
  {
    \__template_execute_if_type_exist:nT {#1}
      {
        \__template_execute_if_arg_agree:nnT {#1} {#3}
          {
            \__template_if_keys_exist:nnT {#1} {#2}
              {
                \__template_store_key_implementation:nnn {#1} {#2} {#4}
                \regex_match:nnTF { \c { AssignTemplateKeys } } {#5}
```

```
                  { \__template_declare_template_code:nnnn {#1} {#2} {#3} {#5} }
                  {
                    \__template_declare_template_code:nnnn
                      {#1} {#2} {#3} { \AssignTemplateKeys #5 }
                  }
              }
          }
      }
   }
\cs_new_protected:Npn \__template_declare_template_code:nnnn #1#2#3#4
  {
    \cs_if_exist:cT { \c__template_code_root_tl #1 / #2 }
      { \msg_info:nnnn { template } { declare-template-code } {#1} {#2} }
    \cs_generate_from_arg_count:cNnn
      { \c__template_code_root_tl #1 / #2 }
      \cs_gset_protected:Npn {#3} {#4}
  }
```

(*End of definition for* `\__template_declare_template_code:nnnnn` *and* `\__template_declare_template_code:nnnn`.)

`\__template_store_key_implementation:nnn`

Actually storing the implementation part of a template is quite easy as it only requires the list of keys given to be turned into a property list. There is also some error-checking to do, hence the need to have the list of defined keytypes available. In certain cases (when choices are involved) parsing the key results in changes to the default values. That is why they are loaded and then saved again.

```
\cs_new_protected:Npn \__template_store_key_implementation:nnn #1#2#3
  {
    \__template_recover_defaults:nn {#1} {#2}
    \__template_recover_keytypes:nn {#1} {#2}
    \prop_clear:N \l__template_vars_prop
    \keyval_parse:nnn
      { \__template_parse_vars_elt:n } { \__template_parse_vars_elt:nnn { #1 / #2 } } {#3}
    \__template_store_vars:nn {#1} {#2}
    \prop_map_inline:Nn \l__template_keytypes_prop
      { \msg_error:nnnnn { template } { key-not-implemented } {##1} {#2} {#1} }
  }
```

(*End of definition for* `\__template_store_key_implementation:nnn`.)

`\__template_parse_vars_elt:n`

At the implementation stage, every key must have a value given. So this is an error function.

```
\cs_new_protected:Npn \__template_parse_vars_elt:n #1
  { \msg_error:nnn { template } { key-no-variable } {#1} }
```

(*End of definition for* `\__template_parse_vars_elt:n`.)

`\__template_parse_vars_elt:nnn`
`\__template_parse_vars_elt_aux:nn`
`\__template_parse_vars_elt_aux:nw`
`\__template_parse_vars_elt_aux:nnn`
`\__template_parse_vars_elt_aux:nne`
`\__template_parse_vars_elt_key:nn`

The actual storage part here is very simple: the storage bin name is placed into the property list. At the same time, a comparison is made with the keytypes defined earlier: if there is a mismatch then an error is raised.

```
\cs_new_protected:Npn \__template_parse_vars_elt:nnn #1#2#3
 {
    \tl_set:Ne \l__template_key_name_tl
      { \tl_trim_spaces:e { \tl_to_str:n {#2} } }
```

```
    \prop_get:NVNTF \l__template_keytypes_prop
      \l__template_key_name_tl
      \l__template_keytype_tl
      {
        \__template_split_keytype_arg:V \l__template_keytype_tl
        \__template_parse_vars_elt_aux:nn {#1} {#3}
        \prop_remove:NV \l__template_keytypes_prop \l__template_key_name_tl
      }
      { \msg_error:nnn { template } { unknown-key } {#2} }
  }
```

Split off any leading `global` and they look for the way to implement.

```
\cs_new_protected:Npn \__template_parse_vars_elt_aux:nn #1#2
  {
    \__template_parse_vars_elt_aux:nw {#1} #2 global global \s__template_stop
  }
\cs_new_protected:Npn \__template_parse_vars_elt_aux:nw
  #1#2 global #3 global #4 \s__template_stop
  {
    \tl_if_blank:nTF {#4}
      { \__template_parse_vars_elt_aux:nnn {#1} { } {#2} }
      {
        \tl_if_blank:nTF {#2}
          {
            \__template_parse_vars_elt_aux:nne
              {#1} { global } { \tl_trim_spaces:n {#3} }
          }
          { \msg_error:nnn { template } { bad-variable } { #2 global #3 } }
      }
  }
\cs_new_protected:Npn \__template_parse_vars_elt_aux:nnn #1#2#3
  {
    \str_case:VnF \l__template_keytype_tl
      {
        { choice } { \__template_implement_choices:nn {#1} {#3} }
        { function }
          {
            \cs_if_exist:NF #3
              { \cs_new:Npn #3 { } }
            \__template_parse_vars_elt_key:nn {#1}
              {
                .code:n =
                  {
                    \cs_generate_from_arg_count:NNnn
                      \exp_not:N #3
                      \exp_not:c
                        { cs_ \str_if_eq:nnT {#1} { global } { g } set:Npn }
                      { \exp_not:V \l__template_keytype_arg_tl }
                      {##1}
                  }
              }
            \prop_put:NVn \l__template_vars_prop
              \l__template_key_name_tl {#2#3}
          }
        { instance }
```

```
          {
            \__template_parse_vars_elt_key:nn {#1}
              {
                .code:n =
                  {
                    \exp_not:c
                      { cs_ \str_if_eq:nnT {#1} { global } { g } set:Npn }
                      \exp_not:N #3 { \UseInstance {##1} }
                  }
              }
            \prop_put:NVn \l__template_vars_prop
              \l__template_key_name_tl {#2#3}
          }
      }
      {
        \tl_if_single:nTF {#3}
          {
            \cs_if_exist:NF #3
              { \use:c { \__template_map_var_type: _new:N } #3 }
            \__template_parse_vars_elt_key:nn {#1}
              {
                . \__template_map_var_type:
                  _ \str_if_eq:nnT {#1} { global } { g } set:N
                    = \exp_not:N #3
              }
            \prop_put:NVn \l__template_vars_prop
              \l__template_key_name_tl {#2#3}
          }
          { \msg_error:nnn { template } { bad-variable } {#2#3} }
      }
  }
\cs_generate_variant:Nn \__template_parse_vars_elt_aux:nnn { nne }
\cs_new_protected:Npn \__template_parse_vars_elt_key:nn #1#2
  {
    \keys_define:ne { template / #1 }
      { \l__template_key_name_tl #2 }
  }
```

(*End of definition for* `\__template_parse_vars_elt:nnn` *and others.*)

`\__template_map_var_type:` Turn a "friendly" variable type into an `expl3` one.

```
\cs_new:Npn \__template_map_var_type:
  {
    \str_case:Vn \l__template_keytype_tl
      {
        { boolean }   { bool }
        { commalist } { clist }
        { integer }   { int }
        { length }    { dim }
        { muskip }    { muskip }
        { real }      { fp }
        { skip }      { skip }
        { tokenlist } { tl }
      }
  }
```

(*End of definition for* `\__template_map_var_type:.`)

__template_implement_choices:nn
__template_implement_choices_default:

Implementing choices requires a second key–value loop. So after a little set-up, the standard parser is called.

```
\cs_new_protected:Npn \__template_implement_choices:nn #1#2
  {
    \clist_set:NV \l__template_tmp_clist \l__template_keytype_arg_tl
    \prop_put:NVn \l__template_vars_prop \l__template_key_name_tl { }
    \keys_define:ne { template / #1 } { \l__template_key_name_tl .choice: }
    \keyval_parse:nnn
      { \__template_implement_choice_elt:n }
      { \__template_implement_choice_elt:nnn {#1} }
      {#2}
    \prop_get:NVNT \l__template_values_prop \l__template_key_name_tl
      \l__template_tmp_tl
      { \__template_implement_choices_default: }
    \clist_if_empty:NF \l__template_tmp_clist
      {
        \clist_map_inline:Nn \l__template_tmp_clist
          { \msg_error:nnn { template } { choice-not-implemented } {##1} }
      }
  }
```

A sanity check for the default value, so that an error is raised now and not when converting to assignments.

```
\cs_new_protected:Npn \__template_implement_choices_default:
  {
    \tl_set:Ne \l__template_tmp_tl
      { \l__template_key_name_tl \c_space_tl \l__template_tmp_tl }
    \prop_if_in:NVF \l__template_vars_prop \l__template_tmp_tl
      {
        \tl_set:Ne \l__template_tmp_tl
          { \l__template_key_name_tl \c_space_tl \l__template_tmp_tl }
        \prop_if_in:NVF \l__template_vars_prop \l__template_tmp_tl
          {
            \prop_get:NVN \l__template_keytypes_prop \l__template_key_name_tl
              \l__template_tmp_tl
            \__template_split_keytype_arg:V \l__template_tmp_tl
            \prop_get:NVN \l__template_values_prop \l__template_key_name_tl
              \l__template_tmp_tl
            \msg_error:nnVV { template } { unknown-default-choice }
              \l__template_key_name_tl
              \l__template_key_name_tl
          }
      }
  }
```

(*End of definition for* `\__template_implement_choices:nn` *and* `\__template_implement_choices_default:.`)

__template_implement_choice_elt:nnn
__template_implement_choice_elt_aux:nnn
__template_implement_choice_elt_aux:n
__template_implement_choice_elt:n

The actual storage of the implementation of a choice is mainly about error checking. The code here ensures that all choices have to have been declared, apart from the special `unknown` choice, which must come last. The code for each choice is stored along with the key name in the variables property list.

```
\cs_new_protected:Npn \__template_implement_choice_elt:nnn #1#2#3
  {
    \clist_if_empty:NTF \l__template_tmp_clist
      {
        \str_if_eq:nnTF {#2} { unknown }
          { \__template_implement_choice_elt_aux:nnn {#1} {#2} {#3} }
          { \__template_implement_choice_elt_aux:n {#2} }
      }
      {
        \clist_if_in:NnTF \l__template_tmp_clist {#2}
          {
            \clist_remove_all:Nn \l__template_tmp_clist {#2}
            \__template_implement_choice_elt_aux:nnn {#1} {#2} {#3}
          }
          { \__template_implement_choice_elt_aux:n {#2} }
      }
  }
\cs_new_protected:Npn \__template_implement_choice_elt_aux:n #1
  {
    \prop_get:NVN \l__template_keytypes_prop \l__template_key_name_tl
      \l__template_tmp_tl
    \__template_split_keytype_arg:V \l__template_tmp_tl
    \msg_error:nnVn { template } { unknown-choice } \l__template_key_name_tl {#1}
  }
\cs_new_protected:Npn \__template_implement_choice_elt_aux:nnn #1#2#3
  {
    \keys_define:ne { template / #1 }
      { \l__template_key_name_tl / #2 .code:n = { \exp_not:n {#3} } }
    \tl_set:Ne \l__template_tmp_tl
      { \l__template_key_name_tl \c_space_tl #2 }
    \prop_put:NVn \l__template_vars_prop \l__template_tmp_tl {#3}
  }
\cs_new_protected:Npn \__template_implement_choice_elt:n #1
  {
    \msg_error:nnVn { template } { choice-requires-code }
      \l__template_key_name_tl {#1}
  }
```

(*End of definition for* `\__template_implement_choice_elt:nnn` *and others.*)

12.7 Editing template defaults

`\__template_edit_defaults:nnn`

Editing the template defaults means getting the values back out of the store, then parsing the list of new values before putting the updated list back into storage.

```
\cs_new_protected:Npn \__template_edit_defaults:nnn #1#2#3
  {
    \__template_if_keys_exist:nnT {#1} {#2}
      {
        \__template_recover_defaults:nn {#1} {#2}
        \__template_parse_values:nnn {#1} {#2} {#3}
        \__template_store_defaults:nn {#1} {#2}
      }
  }
```

(*End of definition for* `\__template_edit_defaults:nnn`.)

`\__template_parse_values:nnn` The routine to parse values is the same for both editing a template and setting up an instance. So the code here does only the minimum necessary for reading the values.

```
\cs_new_protected:Npn \__template_parse_values:nnn #1#2#3
  {
    \__template_recover_keytypes:nn {#1} {#2}
    \keyval_parse:NNn
      \__template_parse_values_elt:n \__template_parse_values_elt:nn {#3}
  }
```

(*End of definition for* `\__template_parse_values:nnn`.)

`\__template_parse_values_elt:n` Every key needs a value, so this is just an error routine.

```
\cs_new_protected:Npn \__template_parse_values_elt:n #1
  {
    \bool_set_true:N \l__template_error_bool
    \msg_error:nnn { template } { key-no-value } {#1}
  }
```

(*End of definition for* `\__template_parse_values_elt:n`.)

`\__template_parse_values_elt:nn` `\__template_parse_values_elt_aux:n` To store the value, find the keytype then call the saving function. These need the current key name saved as `\l__template_key_name_tl`.

```
\cs_new_protected:Npn \__template_parse_values_elt:nn #1#2
  {
    \tl_set:Ne \l__template_key_name_tl
      { \tl_trim_spaces:e { \tl_to_str:n {#1} } }
    \prop_get:NVNTF \l__template_keytypes_prop \l__template_key_name_tl
      \l__template_tmp_tl
      { \__template_parse_values_elt_aux:n {#2} }
      { \msg_error:nnV { template } { unknown-key } \l__template_key_name_tl }
  }
\cs_new_protected:Npn \__template_parse_values_elt_aux:n #1
  {
    \__template_split_keytype_arg:V \l__template_tmp_tl
    \use:c { __template_store_value_ \l__template_keytype_tl :n } {#1}
  }
```

(*End of definition for* `\__template_parse_values_elt:nn` *and* `\__template_parse_values_elt_aux:n`.)

`\__template_template_set_eq:nnn` To copy a template, each of the lists plus the code has to be copied across. To keep this independent of the list storage system, it is all done with two-part shuffles.

```
\cs_new_protected:Npn \__template_template_set_eq:nnn #1#2#3
  {
    \__template_recover_defaults:nn {#1} {#3}
    \__template_store_defaults:nn {#1} {#2}
    \__template_recover_keytypes:nn {#1} {#3}
    \__template_store_keytypes:nn {#1} {#2}
    \__template_recover_vars:nn {#1} {#3}
    \__template_store_vars:nn {#1} {#2}
    \cs_if_exist:cT { \c__template_code_root_tl #1 / #2 }
      { \msg_info:nnnn { template } { declare-template-code } {#1} {#2} }
    \cs_gset_eq:cc { \c__template_code_root_tl #1 / #2 }
      { \c__template_code_root_tl #1 / #3 }
  }
```

(*End of definition for* `\__template_template_set_eq:nnn`.)

12.8 Creating instances of templates

`\__template_declare_instance:nnnn`
`\__template_declare_instance_aux:nnnn`

Making an instance has two distinct parts. First, the keys given are parsed to transfer the values into the structured data format used internally. This allows the default and given values to be combined with no repetition. In the second step, the structured data is converted to pre-defined variable assignments, and these are stored in the function for the instance.

```
\cs_new_protected:Npn \__template_declare_instance:nnnn #1#2#3#4
  {
    \__template_execute_if_code_exist:nnT {#1} {#2}
      {
        \__template_recover_defaults:nn {#1} {#2}
        \__template_recover_vars:nn {#1} {#2}
        \__template_declare_instance_aux:nnnn {#1} {#2} {#3} {#4}
      }
  }
\cs_new_protected:Npn \__template_declare_instance_aux:nnnn #1#2#3#4
  {
    \bool_set_false:N \l__template_error_bool
    \__template_parse_values:nnn {#1} {#2} {#4}
    \bool_if:NF \l__template_error_bool
      {
        \prop_put:Nnn \l__template_values_prop { from~template } {#2}
        \__template_store_values:nn {#1} {#3}
        \__template_convert_to_assignments:
        \cs_if_exist:cT { \c__template_instances_root_tl #1 / #3 }
          { \msg_info:nnnn { template } { declare-instance } {#3} {#1} }
        \cs_set_protected:cpe { \c__template_instances_root_tl #1 / #3 }
          {
            \exp_not:N \__template_assignments_push:n
              { \exp_not:V \l__template_assignments_tl }
            \exp_not:c { \c__template_code_root_tl #1 / #2 }
          }
      }
  }
```

(*End of definition for* `\__template_declare_instance:nnnn` *and* `\__template_declare_instance_aux:nnnn`.)

`\__template_instance_set_eq:nnn`

Copy–paste an instance.

```
\cs_new_protected:Npn \__template_instance_set_eq:nnn #1#2#3
  {
    \__template_if_instance_exist:nnTF {#1} {#3}
      {
        \__template_recover_values:nn {#1} {#3}
        \__template_store_values:nn {#1} {#2}
        \cs_if_exist:cT { \c__template_instances_root_tl #1 / #2 }
          { \msg_info:nnnn { template } { declare-instance } {#2} {#1} }
        \cs_set_eq:cc { \c__template_instances_root_tl #1 / #2 }
          { \c__template_instances_root_tl #1 / #3 }
      }
      { \msg_error:nnnn { template } { unknown-instance } {#1} {#3} }
```

```
  }
```

(*End of definition for* `\__template_instance_set_eq:nnn`.)

__template_edit_instance:nnn
__template_edit_instance_aux:nnnnn
__template_edit_instance_aux:nVnnn

Editing an instance is almost identical to declaring one. The only variation is the source of the values to use. When editing, they are recovered from the previous instance run.

```
\cs_new_protected:Npn \__template_edit_instance:nnn #1#2#3
  {
    \__template_if_instance_exist:nnTF {#1} {#2}
      {
        \__template_recover_values:nn {#1}  {#2}
        \prop_get:NnN \l__template_values_prop { from~template }
          \l__template_tmp_tl
        \__template_edit_instance_aux:nVnn
          {#1} \l__template_tmp_tl {#2} {#3}
      }
      { \msg_error:nnnn { template } { unknown-instance } {#1} {#2} }
  }
\cs_new_protected:Npn \__template_edit_instance_aux:nnnn #1#2#3#4
  {
    \__template_recover_vars:nn {#1} {#2}
    \__template_declare_instance_aux:nnnn {#1} {#2} {#3} {#4}
  }
\cs_generate_variant:Nn \__template_edit_instance_aux:nnnn { nV }
```

(*End of definition for* `\__template_edit_instance:nnn` *and* `\__template_edit_instance_aux:nnnnn`.)

__template_convert_to_assignments:
__template_convert_to_assignments_aux:n
__template_convert_to_assignments_aux:nn
__template_convert_to_assignments_aux:nV

The idea on converting to a set of assignments is to loop over each key, so that the loop order follows the declaration order of the keys. This is done using a sequence as property lists are not "ordered".

```
\cs_new_protected:Npn \__template_convert_to_assignments:
  {
    \tl_clear:N \l__template_assignments_tl
    \seq_map_function:NN \l__template_key_order_seq
      \__template_convert_to_assignments_aux:n
  }
\cs_new_protected:Npn \__template_convert_to_assignments_aux:n #1
  {
    \prop_get:NnN \l__template_keytypes_prop {#1} \l__template_tmp_tl
    \__template_convert_to_assignments_aux:nV {#1} \l__template_tmp_tl
  }
```

The second auxiliary function actually does the work. The arguments here are the key name (#1) and the keytype (#2). From those, the value to assign and the name of the appropriate variable are recovered. A bit of work is then needed to sort out keytypes with arguments (for example instances), and to look for global assignments. Once that is done, a hand-off can be made to the handler for the relevant keytype.

```
\cs_new_protected:Npn \__template_convert_to_assignments_aux:nn #1#2
  {
    \prop_get:NnNT \l__template_values_prop {#1} \l__template_value_tl
      {
        \prop_get:NnNTF \l__template_vars_prop {#1} \l__template_var_tl
          {
            \__template_split_keytype_arg:n {#2}
```

```
            \str_if_eq:VnF \l__template_keytype_tl { choice }
              {
                \str_if_eq:VnF \l__template_keytype_tl { code }
                  { \__template_find_global: }
              }
            \tl_set:Nn \l__template_key_name_tl {#1}
            \cs_if_exist_use:cF { __template_assign_ \l__template_keytype_tl : }
              { \__template_assign_variable: }
          }
          { \msg_error:nnn { template } { unknown-attribute } {#1} }
      }
  }
\cs_generate_variant:Nn \__template_convert_to_assignments_aux:nn { nV }
```

(*End of definition for* `\__template_convert_to_assignments:`, `\__template_convert_to_assignments_aux:n`, *and* `\__template_convert_to_assignments_aux:nn`.)

`\__template_find_global:`
`\__template_find_global_aux:w`

Global assignments should have the phrase `global` at the front. This is pretty easy to find: no other error checking, though.

```
\cs_new_protected:Npn \__template_find_global:
  {
    \bool_set_false:N \l__template_global_bool
    \tl_if_in:onT \l__template_var_tl { global }
      {
        \exp_after:wN \__template_find_global_aux:w \l__template_var_tl \s__template_stop
      }
  }
\cs_new_protected:Npn \__template_find_global_aux:w  #1 global #2 \s__template_stop
  {
    \tl_set:Nn \l__template_var_tl {#2}
    \bool_set_true:N \l__template_global_bool
  }
```

(*End of definition for* `\__template_find_global:` *and* `\__template_find_global_aux:w`.)

12.9 Using templates directly

`\__template_use_template:nnn`

Directly use a template with a particular parameter setting. This is also picked up if used in a nested fashion inside a parameter list. The idea is essentially the same as creating an instance, just with no saving of the result.

```
\cs_new_protected:Npn \__template_use_template:nnn #1#2#3
  {
    \__template_execute_if_code_exist:nnT {#1} {#2}
      {
        \__template_recover_defaults:nn {#1} {#2}
        \__template_recover_vars:nn {#1} {#2}
        \__template_parse_values:nnn {#1} {#2} {#3}
        \__template_convert_to_assignments:
        \use:c { \c__template_code_root_tl #1 / #2  }
      }
  }
```

(*End of definition for* `\__template_use_template:nnn`.)

12.10 Assigning values to variables

__template_assign_boolean:
__template_assign_boolean_aux:n

Setting a Boolean value is slightly different to everything else as the value can be used to work out which `set` function to call. As long as there is no need to recover things from another variable, everything is pretty easy. If there is, then we need to allow for the fact that the recovered value here will *not* be expandable, so needs to be converted to something that is.

```
\cs_new_protected:Npn \__template_assign_boolean:
  {
    \bool_if:NTF \l__template_global_bool
      { \__template_assign_boolean_aux:n { bool_gset } }
      { \__template_assign_boolean_aux:n { bool_set } }
  }
\cs_new_protected:Npn \__template_assign_boolean_aux:n #1
  {
    \__template_if_key_value:VTF \l__template_value_tl
      {
        \__template_key_to_value:
        \tl_put_right:Ne \l__template_assignments_tl
          {
            \exp_not:c { #1 _eq:NN }
            \exp_not:V \l__template_var_tl
            \exp_not:V \l__template_value_tl
          }
      }
      {
        \tl_put_right:Ne \l__template_assignments_tl
          {
            \exp_not:c { #1 _ \l__template_value_tl :N }
            \exp_not:V \l__template_var_tl
          }
      }
  }
```

(*End of definition for* `\__template_assign_boolean:` *and* `\__template_assign_boolean_aux:n`.)

__template_assign_choice:
__template_assign_choice_aux:nF
__template_assign_choice_aux:eF

The idea here is to find either the choice as-given or else the special `unknown` choice, and to copy the appropriate code across.

```
\cs_new_protected:Npn \__template_assign_choice:
  {
    \__template_assign_choice_aux:eF
      { \l__template_key_name_tl \c_space_tl \l__template_value_tl }
      {
        \__template_assign_choice_aux:eF
          { \l__template_key_name_tl \c_space_tl unknown }
          {
            \prop_get:NVN \l__template_keytypes_prop \l__template_key_name_tl
              \l__template_tmp_tl
            \__template_split_keytype_arg:V \l__template_tmp_tl
            \msg_error:nnVV { template } { unknown-choice }
              \l__template_key_name_tl
              \l__template_value_tl
          }
      }
```

```
  }
\cs_new_protected:Npn \__template_assign_choice_aux:nF #1
  {
    \prop_get:NnNTF \l__template_vars_prop {#1} \l__template_tmp_tl
      { \tl_put_right:NV \l__template_assignments_tl \l__template_tmp_tl }
  }
\cs_generate_variant:Nn \__template_assign_choice_aux:nF { e }
```

(*End of definition for* `\__template_assign_choice:` *and* `\__template_assign_choice_aux:nF`.)

`\__template_assign_function:`
`\__template_assign_function_aux:N`

This looks a bit messy but is only actually one function.

```
\cs_new_protected:Npn \__template_assign_function:
  {
    \bool_if:NTF \l__template_global_bool
      { \__template_assign_function_aux:N \cs_gset:Npn }
      { \__template_assign_function_aux:N \cs_set:Npn  }
  }
\cs_new_protected:Npn \__template_assign_function_aux:N #1
  {
    \tl_put_right:Ne \l__template_assignments_tl
      {
        \cs_generate_from_arg_count:NNnn
          \exp_not:V \l__template_var_tl
          \exp_not:N #1
          { \exp_not:V \l__template_keytype_arg_tl }
          { \exp_not:V \l__template_value_tl }
      }
  }
```

(*End of definition for* `\__template_assign_function:` *and* `\__template_assign_function_aux:N`.)

`\__template_assign_instance:`
`\__template_assign_instance_aux:N`

Using an instance means adding the appropriate function creation to the tl. No checks are made at this stage, so if the instance is not valid then errors will arise later.

```
\cs_new_protected:Npn \__template_assign_instance:
  {
    \bool_if:NTF \l__template_global_bool
      { \__template_assign_instance_aux:N \cs_gset_protected:Npn }
      { \__template_assign_instance_aux:N \cs_set_protected:Npn  }
  }
\cs_new_protected:Npn \__template_assign_instance_aux:N #1
  {
    \tl_put_right:Ne \l__template_assignments_tl
      {
        \exp_not:N #1 \exp_not:V \l__template_var_tl
          {
            \__template_use_instance:nn
              { \exp_not:V \l__template_keytype_arg_tl }
              { \exp_not:V \l__template_value_tl }
          }
      }
  }
```

(*End of definition for* `\__template_assign_instance:` *and* `\__template_assign_instance_aux:N`.)

__template_assign_variable:
__template_assign_variable:n

A general-purpose function for all of the other assignments. As long as the value is not coming from another variable, the stored value is simply transferred for output. We use V-type expansion for the \KeyValue case: for token lists this is essential, whilst for register-based variables, it does no harm and avoids needing a low-level test.

```
\cs_new_protected:Npn \__template_assign_variable:
  {
    \exp_args:Ne \__template_assign_variable:n
      {
        \__template_map_var_type:
        _
        \bool_if:NT \l__template_global_bool { g }
        set:N
      }
  }
\cs_new_protected:Npn \__template_assign_variable:n #1
  {
    \__template_if_key_value:VTF \l__template_value_tl
      {
        \__template_key_to_value:
        \tl_put_right:Ne \l__template_assignments_tl
          {
            \exp_not:c { #1 V } \exp_not:V \l__template_var_tl
             \exp_not:V \l__template_value_tl
          }
      }
      {
        \tl_put_right:Ne \l__template_assignments_tl
          {
            \exp_not:c { #1 n } \exp_not:V \l__template_var_tl
             { \exp_not:V \l__template_value_tl }
          }
      }
  }
```

(*End of definition for* __template_assign_variable: *and* __template_assign_variable:n.)

__template_key_to_value:
__template_key_to_value_auxi:w
__template_key_to_value_auxii:w

The idea here is to recover the attribute value of another key. To do that, the marker is removed and a look up takes place. If this is successful, then the name of the variable of the attribute is returned. This assumes that the value will be used in context where it will be converted to a value, for example when setting a number. There is also a need to check in case the copied value happens to be global.

```
\cs_new_protected:Npn \__template_key_to_value:
  { \exp_after:wN \__template_key_to_value_auxi:w \l__template_value_tl }
\cs_new_protected:Npn \__template_key_to_value_auxi:w \KeyValue #1
  {
    \tl_set:Ne \l__template_tmp_tl { \tl_trim_spaces:e { \tl_to_str:n {#1} } }
    \prop_get:NVNTF \l__template_vars_prop \l__template_tmp_tl
      \l__template_value_tl
      {
        \exp_after:wN \__template_key_to_value_auxii:w \l__template_value_tl
          \s__template_mark global \q__template_nil \s__template_stop
      }
      { \msg_error:nnV { template } { unknown-attribute } \l__template_tmp_tl }
```

```
  }
\cs_new_protected:Npn \__template_key_to_value_auxii:w #1 global #2#3 \s__template_stop
  {
    \__template_quark_if_nil:NF #2
      { \tl_set:Nn \l__template_value_tl {#2} }
  }
```

(*End of definition for* `\__template_key_to_value:`, `\__template_key_to_value_auxi:w`, *and* `\__template_key_to_value_auxii:w`.)

12.11 Using instances

`\__template_use_instance:nn`
`\__template_use_instance_aux:nNnnn`
`\__template_use_instance_aux:nn`

Using an instance is just a question of finding the appropriate function. If nothing is found, an error is raised. One complication is that if the first token of argument `#2` is `\UseTemplate` then that is also valid. There is an error-test to make sure that the types agree, and if so the template is used directly.

```
\cs_new_protected:Npn \__template_use_instance:nn #1#2
  {
    \__template_if_use_template:nTF {#2}
      { \__template_use_instance_aux:nNnnn {#1} #2 }
      { \__template_use_instance_aux:nn {#1} {#2} }
  }
\cs_new_protected:Npn \__template_use_instance_aux:nNnnn #1#2#3#4#5
  {
    \str_if_eq:nnTF {#1} {#3}
      { \__template_use_template:nnn {#3} {#4} {#5} }
      { \msg_error:nnnn { template } { type-mismatch } {#1} {#3} }
}
\cs_new_protected:Npn \__template_use_instance_aux:nn #1#2
  {
    \__template_if_instance_exist:nnTF {#1} {#2}
      { \use:c { \c__template_instances_root_tl #1 / #2 } }
      { \msg_error:nnnn { template } { unknown-instance } {#1} {#2} }
  }
```

(*End of definition for* `\__template_use_instance:nn`, `\__template_use_instance_aux:nNnnn`, *and* `\__template_use_instance_aux:nn`.)

12.12 Assignment manipulation

A few functions to transfer assignments about, as this is needed by `\AssignTemplateKeys`.

`\__template_assignments_pop:`

To actually use the assignments.

```
\cs_new:Npn \__template_assignments_pop: { \l__template_assignments_tl }
```

(*End of definition for* `\__template_assignments_pop:`.)

`\__template_assignments_push:n`

Here, the assignments are stored for later use.

```
\cs_new_protected:Npn \__template_assignments_push:n #1
  { \tl_set:Nn \l__template_assignments_tl {#1} }
```

(*End of definition for* `\__template_assignments_push:n`.)

12.13 Showing templates and instances

__template_show_code:nn

Showing the code for a template is just a translation of \cs_show:c.

```
\cs_new_protected:Npn \__template_show_code:nn #1#2
  { \cs_show:c { \c__template_code_root_tl #1 / #2 } }
```

(*End of definition for* __template_show_code:nn.)

__template_show_defaults:nn
__template_show_keytypes:nn
__template_show_vars:nn
__template_show:Nnnn

A modified version of the property-list printing code, such that the output refers to templates and instances rather than to the underlying structures.

```
\cs_new_protected:Npn \__template_show_defaults:nn #1#2
  {
    \__template_if_keys_exist:nnT {#1} {#2}
      {
        \__template_recover_defaults:nn {#1} {#2}
        \__template_show:Nnnn \l__template_values_prop
          {#1} {#2} { default~values }
      }
  }
\cs_new_protected:Npn \__template_show_keytypes:nn #1#2
  {
    \__template_if_keys_exist:nnT {#1} {#2}
      {
        \__template_recover_keytypes:nn {#1} {#2}
        \__template_show:Nnnn \l__template_keytypes_prop
          {#1} {#2} { interface }
      }
  }
\cs_new_protected:Npn \__template_show_vars:nn #1#2
  {
     \__template_execute_if_code_exist:nnT {#1} {#2}
      {
        \__template_recover_vars:nn {#1} {#2}
        \__template_show:Nnnn \l__template_vars_prop
          {#1} {#2} { variable~mapping }
      }
  }
\cs_new_protected:Npn \__template_show:Nnnn #1#2#3#4
  {
    \msg_show:nneeee { template } { show-attribute }
      { \tl_to_str:n {#2} }
      { \tl_to_str:n {#3} }
      { \tl_to_str:n {#4} }
      { \prop_map_function:NN #1 \msg_show_item_unbraced:nn }
  }
```

(*End of definition for* __template_show_defaults:nn *and others.*)

__template_show_values:nn

Instance values are a little more complex, as is the template to consider.

```
\cs_new_protected:Npn \__template_show_values:nn #1#2
  {
    \__template_if_instance_exist:nnT {#1} {#2}
      {
        \__template_recover_values:nn {#1} {#2}
```

```
        \msg_show:nneee { template } { show-values }
          { \tl_to_str:n {#1} }
          { \tl_to_str:n {#2} }
          {
            \prop_map_function:NN \l__template_values_prop
              \msg_show_item_unbraced:nn
          }
      }
  }
```

(*End of definition for* `\__template_show_values:nn`.)

12.14 Messages

The text for error messages: short and long text for all of them.

```
\msg_new:nnnn { template } { argument-number-mismatch }
  { Template~type~'#1'~takes~#2~argument(s). }
  {
    Templates~of~type~'#1'~require~#2~argument(s).\\
    You~have~tried~to~make~a~template~for~'#1'~
    with~#3~argument(s),~which~is~not~possible:~
    the~number~of~arguments~must~agree.
  }
\msg_new:nnnn { template } { bad-number-of-arguments }
  { Bad~number~of~arguments~for~template~type~'#1'. }
  {
    A~template~may~accept~between~0~and~9~arguments.\\
    You~asked~to~use~#2~arguments:~this~is~not~supported.
  }
\msg_new:nnnn { template } { bad-variable }
  { Incorrect~variable~description~'#1'. }
  {
    The~argument~'#1'~is~not~of~the~form \\
    ~~'<variable>'\\
    ~or~\\
    ~~'global~<variable>'.\\
    It~must~be~given~in~one~of~these~formats~to~be~used~in~a~template.
  }
\msg_new:nnnn { template } { choice-not-implemented }
  { The~choice~'#1'~has~no~implementation. }
  {
    Each~choice~listed~in~the~interface~for~a~template~must~
    have~an~implementation.
  }
\msg_new:nnnn { template } { choice-no-code }
  { The~choice~'#1'~requires~implementation~details. }
  {
    When~creating~template~code~using~\DeclareTemplateCode,~
    each~choice~name~must~have~an~associated~implementation.\\
    This~should~be~given~after~a~'='~sign:~LaTeX~did~not~find~one.
  }
\msg_new:nnnn { template } { choice-requires-code }
  { The~choice~'#2'~for~key~'#1'~requires~an~implementation. }
  {
```

```
    You~should~have~put:\\
    \ \ #1~:~choice~{~#2 = <code> ~} \\
    but~LaTeX~did~not~find~any~<code>.
  }
\msg_new:nnnn { template } { duplicate-key-interface }
  { Key~'#1'~appears~twice~in~interface~definition~\msg_line_context:. }
  {
    Each~key~can~only~have~one~interface~declared~in~a~template.\\
    LaTeX~found~two~interfaces~for~'#1'.
  }
\msg_new:nnnn { template } { keytype-requires-argument }
  { The~key~type~'#1'~requires~an~argument~\msg_line_context:. }
  {
    You~should~have~put:\\
    \ \ <key-name>~:~#1~{~<argument>~} \\
    but~LaTeX~did~not~find~an~<argument>.
  }
\msg_new:nnnn { template } { invalid-keytype }
  { The~key~'#1'~is~missing~a~key-type~\msg_line_context:. }
  {
    Each~key~in~a~template~requires~a~key-type,~given~in~the~form:\\
    \ \ <key>~:~<key-type>\\
    LaTeX~could~not~find~a~<key-type>~in~your~input.
  }
\msg_new:nnnn { template } { key-no-value }
  { The~key~'#1'~has~no~value~\msg_line_context:. }
  {
    When~creating~an~instance~of~a~template~
    every~key~listed~must~include~a~value:\\
    \ \ <key>~=~<value>
  }
\msg_new:nnnn { template } { key-no-variable }
  { The~key~'#1'~requires~implementation~details~\msg_line_context:. }
  {
    When~creating~template~code~using~\DeclareTemplateCode,~
    each~key~name~must~have~an~associated~implementation.\\
    This~should~be~given~after~a~'='~sign:~LaTeX~did~not~find~one.
  }
\msg_new:nnnn { template } { key-not-implemented }
  { Key~'#1'~has~no~implementation~\msg_line_context:. }
  {
    The~definition~of~key~implementations~for~template~'#2'~
    of~template~type~'#3'~does~not~include~any~details~for~key~'#1'.\\
    The~key~was~declared~in~the~interface~definition,~
    and~so~an~implementation~is~required.
  }
\msg_new:nnnn { template } { missing-keytype }
  { The~key~'#1'~is~missing~a~key-type~\msg_line_context:. }
  {
    Key~interface~definitions~should~be~of~the~form\\
    \ \ #1~:~<key-type>\\
    but~LaTeX~could~not~find~a~<key-type>.
  }
\msg_new:nnnn { template } { no-template-code }
```

```
  {
    The~template~'#2'~of~type~'#1'~is~unknown~
    or~has~no~implementation.
  }
  {
    There~is~no~code~available~for~the~template~name~given.\\
    This~should~be~given~using~\DeclareTemplateCode.
  }
\msg_new:nnnn { template } { type-already-defined }
  { Template~type~'#1'~already~defined. }
  {
    You~have~used~\NewTemplateType~
    with~a~template~type~that~has~already~been~defined.
  }
\msg_new:nnnn { template } { type-mismatch }
  { Template~types~'#1'~and~'#2'~do~not~agree. }
  {
    You~are~trying~to~use~a~template~directly~with~\UseInstance
    (or~a~similar~function),~but~the~template~types~do~not~match.
  }
\msg_new:nnnn { template } { unknown-attribute }
  { The~template~attribute~'#1'~is~unknown. }
  {
    There~is~a~definition~in~the~current~template~reading\\
    \ \ \token_to_str:N \KeyValue {~#1~} \\
    but~there~is~no~key~called~'#1'.
  }
\msg_new:nnnn { template } { unknown-choice }
  { The~choice~'#2'~was~not~declared~for~key~'#1'. }
  {
    The~key~'#1'~takes~a~fixed~list~of~choices~
    and~this~list~does~not~include~'#2'.
  }
\msg_new:nnnn { template } { unknown-default-choice }
  { The~default~choice~'#2'~was~not~declared~for~key~'#1'. }
  {
    The~key~'#1'~takes~a~fixed~list~of~choices~
    and~this~list~does~not~include~'#2'.
  }
\msg_new:nnnn { template } { unknown-instance }
  { The~instance~'#2'~of~type~'#1'~is~unknown. }
  {
    You~have~asked~to~use~an~instance~'#2',~
    but~this~has~not~been~created.
  }
\msg_new:nnnn { template } { unknown-key }
  { Unknown~template~key~'#1'. }
  {
    The~key~'#1'~was~not~declared~in~the~interface~
    for~the~current~template.
  }
\msg_new:nnnn { template } { unknown-keytype }
  { The~key-type~'#1'~is~unknown. }
  {
```

```
    Valid~key-types~are:\\
    -~boolean;\\
    -~choice;\\
    -~commalist;\\
    -~function;\\
    -~instance;\\
    -~integer;\\
    -~length;\\
    -~muskip;\\
    -~real;\\
    -~skip;\\
    -~tokenlist.
  }
\msg_new:nnnn { template } { unknown-type }
  { The~template~type~'#1'~is~unknown. }
  {
    A~template~type~needs~to~be~defined~with~\NewTemplateType
    prior~to~using~it.
  }
\msg_new:nnnn { template } { unknown-template }
  { The~template~'#2'~of~type~'#1'~is~unknown. }
  {
    No~interface~has~been~declared~for~a~template~
    '#2'~of~template~type~'#1'.
  }
```

Information messages only have text: more text should not be needed.

```
\msg_new:nnn { template } { declare-instance }
  { Declaring~instance~~'#1'~of~type~#2~\msg_line_context:. }
\msg_new:nnn { template } { declare-template-code }
  { Declaring~code~for~template~'#2'~of~template~type~'#1'~\msg_line_context:. }
\msg_new:nnn { template } { declare-template-interface }
  {
    Declaring~interface~for~template~'#2'~of~template~type~'#1'~
    \msg_line_context:.
  }
\msg_new:nnn { template } { declare-type }
  { Declaring~template~type~'#1'~taking~#2~argument(s)~\msg_line_context:. }
\msg_new:nnn { template } { show-attribute }
  {
    The~template~'#2'~of~type~'#1'~has~
    \tl_if_empty:nTF {#4} { no~#3. } { #3 : #4 }
  }
\msg_new:nnn { template } { show-values }
  {
    The~instance~'#2'~of~type~'#1'~has~
    \tl_if_empty:nTF {#3} { no~values. } { values: #3 }
  }
```

Also add template to the LaTeX messages.

```
\prop_gput:Nnn \g_msg_module_type_prop { template } { LaTeX }
```

12.15 User functions

\NewTemplateType
\DeclareTemplateInterface
\DeclareTemplateCode
\DeclareTemplateCopy
\EditTemplateDefaults
\UseTemplate
\DeclareInstance
\DeclareInstanceCopy
\EditInstance
\UseInstance

All simple translations.

```
\cs_new_protected:Npn \NewTemplateType #1#2
  { \__template_define_type:nn {#1} {#2} }
\cs_new_protected:Npn \DeclareTemplateInterface #1#2#3#4
  { \__template_declare_template_keys:nnnn {#1} {#2} {#3} {#4} }
\cs_new_protected:Npn \DeclareTemplateCode #1#2#3#4#5
  { \__template_declare_template_code:nnnnn {#1} {#2} {#3} {#4} {#5} }
\cs_new_protected:Npn \DeclareTemplateCopy #1#2#3
  { \__template_template_set_eq:nnn {#1} {#2} {#3} }
\cs_new_protected:Npn \EditTemplateDefaults #1#2#3
  { \__template_edit_defaults:nnn {#1} {#2} {#3} }
\cs_new_protected:Npn \UseTemplate #1#2#3
  { \__template_use_template:nnn {#1} {#2} {#3} }
\cs_new_protected:Npn \DeclareInstance #1#2#3#4
  { \__template_declare_instance:nnnn {#1} {#3} {#2} {#4} }
\cs_new_protected:Npn \DeclareInstanceCopy #1#2#3
  { \__template_instance_set_eq:nnn {#1} {#2} {#3} }
\cs_new_protected:Npn \EditInstance #1#2#3
  { \__template_edit_instance:nnn {#1} {#2} {#3} }
\cs_new_protected:Npn \UseInstance #1#2
  { \__template_use_instance:nn {#1} {#2} }
```

(*End of definition for* `\NewTemplateType` *and others. These functions are documented on page* *343*.)

\ShowTemplateCode
\ShowTemplateDefaults
\ShowTemplateInterface
\ShowTemplateVariables
\ShowInstanceValues

The show functions are again just translation.

```
\cs_new_protected:Npn \ShowTemplateCode #1#2
  { \__template_show_code:nn {#1} {#2} }
\cs_new_protected:Npn \ShowTemplateDefaults #1#2
  { \__template_show_defaults:nn {#1} {#2} }
\cs_new_protected:Npn \ShowTemplateInterface #1#2
  { \__template_show_keytypes:nn {#1} {#2} }
\cs_new_protected:Npn \ShowTemplateVariables #1#2
  { \__template_show_vars:nn {#1} {#2} }
\cs_new_protected:Npn \ShowInstanceValues #1#2
  { \__template_show_values:nn {#1} {#2} }
```

(*End of definition for* `\ShowTemplateCode` *and others. These functions are documented on page* *350*.)

\IfInstanceExistsT
\IfInstanceExistsF
\IfInstanceExistsTF

More direct translation.

```
\cs_new:Npn \IfInstanceExistsTF #1#2
  { \__template_if_instance_exist:nnTF {#1} {#2} }
\cs_new:Npn \IfInstanceExistsT #1#2
  { \__template_if_instance_exist:nnT {#1} {#2} }
\cs_new:Npn \IfInstanceExistsF #1#2
  { \__template_if_instance_exist:nnF {#1} {#2} }
```

(*End of definition for* `\IfInstanceExistsT`, `\IfInstanceExistsF`, *and* `\IfInstanceExistsTF`. *These functions are documented on page* *348*.)

\KeyValue

Simply dump the argument when executed: this should not happen.

```
\cs_new_protected:Npn \KeyValue #1 {#1}
```

(*End of definition for* `\KeyValue`. *This function is documented on page* *345*.)

\AssignTemplateKeys A short call to use a token register by proxy.

```
\cs_new_protected:Npn \AssignTemplateKeys { \__template_assignments_pop: }
```

(*End of definition for* `\AssignTemplateKeys`. *This function is documented on page* 345.)

\SetTemplateKeys A friendly wrapper

```
\cs_new_protected:Npn \SetTemplateKeys #1#2#3
  { \keys_set_known:nnN { template / #1 / #2 } {#3} \l__template_tmp_clist }
```

(*End of definition for* `\SetTemplateKeys`. *This function is documented on page* 349.)

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{lttemplates}%
⟨latexrelease⟩                  {Prototype~document~commands}%
⟨latexrelease⟩
⟨latexrelease⟩\EndModuleRelease
```

```
\ExplSyntaxOff
```

```
⟨/2ekernel | latexrelease⟩
```

We need to stop DocStrip treating `@@` in a special way at this point.

```
⟨@@=⟩
```

File 12
ltalloc.dtx

1 Counters

This section deals with counter and other variable allocation.

```
⟨*2ekernel⟩
```

The following are from plain TEX:

`\z@` A zero dimen or number. It's more efficient to write `\parindent\z@` than `\parindent 0pt`.

`\@ne` The number 1.

`\m@ne` The number −1.

`\tw@` The number 2.

`\sixt@@n` The number 16.

`\@m` The number 1000.

`\@MM` The number 20000.

`\@xxxii` The constant 32.

```
\chardef\@xxxii=32
```

(*End of definition for* `\@xxxii`.)

`\@Mi` `\@Mii` `\@Miii` `\@Miv` Constants 10001–10004.

```
\mathchardef\@Mi=10001
\mathchardef\@Mii=10002
\mathchardef\@Miii=10003
\mathchardef\@Miv=10004
```

(*End of definition for* `\@Mi` *and others.*)

`\@tempcnta` `\@tempcntb` Scratch count registers used by LATEX kernel commands.

```
\newcount\@tempcnta
\newcount\@tempcntb
```

(*End of definition for* `\@tempcnta` *and* `\@tempcntb`.)

`\if@tempswa` General boolean switch used by LATEX kernel commands.

```
\newif\if@tempswa
```

(*End of definition for* `\if@tempswa`.)

`\@tempdima` `\@tempdimb` `\@tempdimc` Scratch dimen registers used by LATEX kernel commands.

```
\newdimen\@tempdima
\newdimen\@tempdimb
\newdimen\@tempdimc
```

(*End of definition for* `\@tempdima`, `\@tempdimb`, *and* `\@tempdimc`.)

`\@tempboxa` Scratch box register used by LaTeX kernel commands.

```
\newbox\@tempboxa
```

(*End of definition for* `\@tempboxa`.)

`\@tempskipa` `\@tempskipb` Scratch skip registers used by LaTeX kernel commands.

```
\newskip\@tempskipa
\newskip\@tempskipb
```

(*End of definition for* `\@tempskipa` *and* `\@tempskipb`.)

`\@temptokena` Scratch token register used by LaTeX kernel commands.

```
\newtoks\@temptokena
```

(*End of definition for* `\@temptokena`.)

`\@flushglue` Glue used for `\right-` & `\leftskip` = 0pt plus 1fil

```
\newskip\@flushglue \@flushglue = 0pt plus 1fil
```

(*End of definition for* `\@flushglue`.)

⟨/2ekernel⟩

File 13
ltcntrl.dtx

1 Program control structure

This section defines a number of control structure macros, such as while-loops and for-loops.
Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
⟨*2ekernel⟩
\message{control,}
```

```
\@whilenum TEST \do {BODY}
\@whiledim TEST \do {BODY}  : These implement the loop
          while  TEST  do  BODY  od
    where  TEST  is a TeX \ifnum or \ifdim test, respectively.
    They are optimized for the normal case of TEST initially false.

\@whilesw SWITCH \fi {BODY} : Implements the loop
               while SWITCH do BODY od
    Optimized for normal case of SWITCH initially false.

\@for NAME := LIST \do {BODY} : Assumes that LIST expands to A1,A2,
     ... ,An .
     Executes  BODY  n  times, with  NAME = Ai  on the i-th iteration.
     Optimized for the normal case of n = 1.  Works for n=0.

\@tfor NAME := LIST \do {BODY}
     if, before expansion, LIST = T1 ... Tn  where each Ti is a
     token or {...}, then executes  BODY  n  times, with  NAME = Ti
     on the i-th iteration.  Works for n=0.

 NOTES: 1. These macros use no \@temp sequences.
        2. These macros do not work if the body contains anything that
        looks syntactically to TeX like an improperly balanced \if
        \else \fi.

\@whilenum TEST \do {BODY} ==
 BEGIN
   if  TEST
     then  BODY
           \@iwhilenum{TEST \relax BODY}
 END

\@iwhilenum {TEST BODY} ==
 BEGIN
   if  TEST
     then  BODY
```

```
            \@nextwhile = def(\@iwhilenum)
      else  \@nextwhile = def(\@whilenoop)
    fi
    \@nextwhile {TEST BODY}
  END

 \@whilesw SWITCH \fi {BODY} ==
  BEGIN
    if SWITCH
      then BODY
           \@iwhilesw {SWITCH BODY}\fi
    fi
  END

 \@iwhilesw {SWITCH BODY} \fi ==
  BEGIN
    if SWITCH
      then BODY
           \@nextwhile = def(\@iwhilesw)
      else \@nextwhile = def(\@whileswnoop)
    fi
    \@nextwhile {SWITCH BODY} \fi
  END
```

End of historical LaTeX 2.09 comments.

\@whilenoop
\@whilenum
\@iwhilenum

```
\long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
     #2\relax}\fi}
\long\def\@iwhilenum#1{\ifnum #1\expandafter\@iwhilenum
         \else\expandafter\@gobble\fi{#1}}
```

(*End of definition for* `\@whilenoop`, `\@whilenum`, *and* `\@iwhilenum`.)

\@whiledim
\@iwhiledim

```
\long\def\@whiledim#1\do #2{\ifdim #1\relax#2\@iwhiledim{#1\relax#2}\fi}
\long\def\@iwhiledim#1{\ifdim #1\expandafter\@iwhiledim
        \else\expandafter\@gobble\fi{#1}}
```

(*End of definition for* `\@whiledim` *and* `\@iwhiledim`.)

\@whileswnoop
\@whilesw
\@iwhilesw

```
\long\def\@whilesw#1\fi#2{#1#2\@iwhilesw{#1#2}\fi\fi}
\long\def\@iwhilesw#1\fi{#1\expandafter\@iwhilesw
         \else\@gobbletwo\fi{#1}\fi}
```

(*End of definition for* `\@whileswnoop`, `\@whilesw`, *and* `\@iwhilesw`.)

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\@for NAME := LIST \do {BODY} ==
   BEGIN \@forloop expand(LIST),\@nil,\@nil \@@ NAME {BODY} END

\@forloop CAR, CARCDR, CDRCDR \@@ NAME {BODY} ==
  BEGIN
    NAME = CAR
    if def(NAME) = def(\@nnil)
      else BODY;
            NAME = CARCDR
            if def(NAME) = def(\@nnil)
              else BODY
                    \@iforloop CDRCDR \@@ NAME \do {BODY}
            fi
    fi
  END

\@iforloop CAR, CDR \@@ NAME {BODY} =
    NAME = CAR
    if def(NAME) = def(\@nnil)
       then  \@nextwhile = def(\@fornoop)
       else  BODY ;
              \@nextwhile = def(\@iforloop)
    fi
    \@nextwhile name cdr {body}

\@tfor NAME := LIST \do {BODY}
   =  \@tforloop LIST \@nil \@@ NAME {BODY}

\@tforloop car cdr \@@ name {body} =
    name = car
    if def(name) = def(\@nnil)
       then  \@nextwhile == \@fornoop
       else  body ;
              \@nextwhile == \@forloop
    fi
    \@nextwhile name cdr {body}
```

End of historical LaTeX 2.09 comments.

`\@nnil`

```
\def\@nnil{\@nil}
```

(*End of definition for* `\@nnil`.)

`\@empty`

```
\def\@empty{}
```

(*End of definition for* `\@empty`.)

\@fornoop

```
\long\def\@fornoop#1\@@#2#3{}
```

(*End of definition for* \@fornoop.)

\@for

```
\long\def\@for#1:=#2\do#3{%
  \expandafter\def\expandafter\@fortmp\expandafter{#2}%
  \ifx\@fortmp\@empty \else
    \expandafter\@forloop#2,\@nil,\@nil\@@#1{#3}\fi}
```

(*End of definition for* \@for.)

\@forloop

```
\long\def\@forloop#1,#2,#3\@@#4#5{\def#4{#1}\ifx #4\@nnil \else
       #5\def#4{#2}\ifx #4\@nnil \else#5\@iforloop #3\@@#4{#5}\fi\fi}
```

(*End of definition for* \@forloop.)

\@iforloop

```
\long\def\@iforloop#1,#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
       \expandafter\@fornoop \else
      #4\relax\expandafter\@iforloop\fi#2\@@#3{#4}}
```

(*End of definition for* \@iforloop.)

\@tfor

```
\def\@tfor#1:={\@tf@r#1 }
\long\def\@tf@r#1#2\do#3{\def\@fortmp{#2}\ifx\@fortmp\space\else
    \@tforloop#2\@nil\@nil\@@#1{#3}\fi}
\long\def\@tforloop#1#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
       \expandafter\@fornoop \else
      #4\relax\expandafter\@tforloop\fi#2\@@#3{#4}}
```

(*End of definition for* \@tfor.)

\@break@tfor Break out of a \@tfor loop. This should be called *inside* the scope of an \if. See \@iffileonpath for an example.

```
\long\def\@break@tfor#1\@@#2#3{\fi\fi}
```

(*End of definition for* \@break@tfor.)

\@removeelement Removes an element from a comma-separated list and puts it into a control sequence, called as \@removeelement{⟨*element*⟩}{⟨*list*⟩}{⟨*cs*⟩}. Due to the implementation method the ⟨*element*⟩ is not allowed to contain braces.

```
\def\@removeelement#1#2#3{%
  \def\reserved@a##1,#1,##2\reserved@a{##1,##2\reserved@b}%
  \def\reserved@b##1,\reserved@b##2\reserved@b{%
    \ifx,##1\@empty\else##1\fi}%
  \edef#3{%
    \expandafter\reserved@b\reserved@a,#2,\reserved@b,#1,\reserved@a}}
```

(*End of definition for* \@removeelement.)

⟨/2ekernel⟩

File 14
lterror.dtx

1 Error handling and tracing

This section defines LaTeX's error commands.

```
⟨*2ekernel⟩
```

The '2ekernel' code ensures that a `\usepackage{autoerr}` is essentially ignored if a 'full' format is being used that has the error messages already in the format.

These days we don't support autoloading approach any longer, but this part bit is kept in case it is used in old documents.

```
\expandafter\let\csname ver@autoerr.sty\endcsname\fmtversion
```

1.1 General commands

`\MessageBreak` This command prints a new-line inside a message, followed by a continuation line begun with `\@msg@continuation`. Normally it is defined to be `\relax`, but inside messages, it is let to `\@message@break`.

```
\let\MessageBreak\relax
```

(*End of definition for* `\MessageBreak`.)

`\GenericInfo` This takes two arguments: a continuation and a message, and sends the result to the log file.

```
\DeclareRobustCommand{\GenericInfo}[2]{%
   \begingroup
      \def\MessageBreak{^^J#1}%
      \set@display@protect
      \immediate\write\m@ne{#2\on@line.}%
   \endgroup
}
```

(*End of definition for* `\GenericInfo`.)

`\GenericWarning` This takes two arguments: a continuation and a message, and sends the result to the screen.

```
\DeclareRobustCommand{\GenericWarning}[2]{%
   \begingroup
      \def\MessageBreak{^^J#1}%
      \set@display@protect
      \immediate\write\@unused{^^J#2\on@line.^^J}%
   \endgroup
}
```

(*End of definition for* `\GenericWarning`.)

`\GenericError` This macro takes four arguments: a continuation, an error message, where to go for further information, and the help information. It displays the error message, and sets the error help (the result of typing `h` to the prompt), and does a horrible hack to turn the last context line (which by default is the only context line) into just three dots. This could be made more efficient.

```
\bgroup
\lccode`\@=`\ %
\lccode`\~=`\ %
\lccode`\}=`\ %
\lccode`\{=`\ %
\lccode`\T=`\T%
\lccode`\H=`\H%
\catcode`\ =11\relax%
\lowercase{%
\egroup%
```

Unfortunately TEX versions older than 3.141 have a bug which means that `^^J` does not force a linebreak in `\message` and `\errmessage` commands. So for these old TEX's we use `\typeout` to produce the message, and then have an empty `\errmessage` command. This causes an extra line of the form

.

To appear on the terminal, but if you do not like it, you can always upgrade your TEX! In order for your format to use this version, you must define the macro `\@TeXversion` to be the version number, e.g., 3.14 of the underlying TEX. See the comments in `ltdircheck.dtx`.

```
\dimen@\ifx\@TeXversion\@undefined4\else\@TeXversion\fi\p@%
\ifdim\dimen@>3.14\p@%
```

First the 'standard case'.

```
\DeclareRobustCommand{\GenericError}[4]{%
\begingroup%
\immediate\write\@unused{}%
\def\MessageBreak{^^J}%
\set@display@protect%
\edef%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                  %
{{#4}}%
\errhelp
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                  %
\let
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                  %
\@empty
\def\MessageBreak{^^J#1}%
\def~{\errmessage{%
#2.^^J^^J%
#3^^J%
Type  H <return>  for immediate help%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                  %
```

```
}}%
~%
\endgroup}%
\else%
```

Secondly the version for old TEX's.

```
\DeclareRobustCommand{\GenericError}[4]{%
\begingroup%
\immediate\write\@unused{}%
\def\MessageBreak{^^J}%
\set@display@protect%
\edef%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                  %
{{#4}}%
\errhelp
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                  %
\let
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                  %
\errmessage
\def\MessageBreak{^^J#1}%
\def~{\typeout{! %
#2.^^J^^J%
#3^^J%
Type  H <return>  for immediate help.}%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                  %
{}}%
~%
\endgroup}%
\fi}%
```

(*End of definition for* `\GenericError`.)

`\PackageError`
`\PackageWarning`
`\PackageWarningNoLine`
`\PackageInfo`
`\ClassError`
`\ClassWarning`
`\ClassWarningNoLine`
`\ClassInfo`

These commands are intended for use by package and class writers, to give information to authors. The syntax is:

```
\PackageError{⟨package⟩}{⟨error⟩}{⟨help⟩}
\PackageWarning{⟨package⟩}{⟨warning⟩}
\PackageWarningNoLine{⟨package⟩}{⟨warning⟩}
\PackageInfo{⟨package⟩}{⟨info⟩}
```

and similarly for classes. The `Error` commands print the ⟨*error*⟩ message, and present the interactive prompt; if the author types `h`, then the ⟨*help*⟩ information is displayed. The `Warning` commands produce a warning but do not present the interactive prompt. The `WarningNoLine` commands do the same, but don't print the input line number. The `Info` commands write the message to the `log` file. Within the messages, the command `\MessageBreak` can be used to break a line, `\protect` can be used to protect command names, and `\space` is a space, for example:

```
\newcommand{\foo}{FOO}
\PackageWarning{ethel}{%
   Your hovercraft is full of eels,\MessageBreak
   and \protect\foo\space is \foo}
```

produces:

```
Package ethel warning: Your hovercraft is full of eels,
(ethel)                and \foo is FOO on input line 54.
```

```
\gdef\PackageError#1#2#3{%
   \GenericError{%
      (#1)\@spaces\@spaces\@spaces\@spaces
   }{%
      Package #1 Error: #2%
   }{%
      See the #1 package documentation for explanation.%
   }{#3}%
}
\def\PackageWarning#1#2{%
   \GenericWarning{%
      (#1)\@spaces\@spaces\@spaces\@spaces
   }{%
      Package #1 Warning: #2%
   }%
}
\def\PackageWarningNoLine#1#2{%
   \PackageWarning{#1}{#2\@gobble}%
}
\def\PackageInfo#1#2{%
   \GenericInfo{%
      (#1) \@spaces\@spaces\@spaces
   }{%
      Package #1 Info: #2%
   }%
}
\gdef\ClassError#1#2#3{%
   \GenericError{%
      (#1) \space\@spaces\@spaces\@spaces
   }{%
      Class #1 Error: #2%
   }{%
      See the #1 class documentation for explanation.%
   }{#3}%
}
\def\ClassWarning#1#2{%
   \GenericWarning{%
      (#1) \space\@spaces\@spaces\@spaces
   }{%
      Class #1 Warning: #2%
   }%
}
\def\ClassWarningNoLine#1#2{%
```

```
   \ClassWarning{#1}{#2\@gobble}%
}
\def\ClassInfo#1#2{%
   \GenericInfo{%
      (#1) \space\space\@spaces\@spaces
   }{%
      Class #1 Info: #2%
   }%
}
```

(*End of definition for* `\PackageError` *and others.*)

`\ClassNote`
`\ClassNoteNoLine`
`\PackageNote`
`\PackageNoteNoLine`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                  {\ClassNote}{Notes for classes/packages}%
\def\ClassNote#1#2{%
   \GenericWarning{%
      (#1) \space\space\@spaces\@spaces
   }{%
      Class #1 Info: #2%
   }%
}
\def\ClassNoteNoLine#1#2{\ClassNote{#1}{#2\@gobble}}
\def\PackageNote#1#2{%
   \GenericWarning{%
      (#1) \@spaces\@spaces\@spaces
   }{%
      Package #1 Info: #2%
   }%
}
\def\PackageNoteNoLine#1#2{\PackageNote{#1}{#2\@gobble}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

We don't roll back, because if this code is used by packages then most often they will not have rollback code implemented, so they would immediately break even if they otherwise would work fine.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\ClassNote}{Notes for classes/packages}%
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\ClassNote` *and others.*)

`\@latex@error`
`\@latex@warning`
`\@latex@warning@no@line`
`\@latex@info`
`\@latex@info@no@line`

Errors and other info, for use in the LaTeX core.

```
\gdef\@latex@error#1#2{%
   \GenericError{%
      \space\space\space\@spaces\@spaces\@spaces
   }{%
      LaTeX Error: #1%
   }{%
```

```
      See the LaTeX manual or LaTeX Companion for explanation.%
   }{#2}%
}
\def\@latex@warning#1{%
   \GenericWarning{%
      \space\space\space\@spaces\@spaces\@spaces
   }{%
      LaTeX Warning: #1%
   }%
}
\def\@latex@warning@no@line#1{%
   \@latex@warning{#1\@gobble}}
\def\@latex@info#1{%
   \GenericInfo{%
      \@spaces\@spaces\@spaces
   }{%
      LaTeX Info: #1%
   }%
}
\def\@latex@info@no@line#1{%
  \@latex@info{#1\@gobble}}
```

`\@font@warning` and `\@font@info` are defined later since they have to be redefined by the `tracefnt` package.

```
def\@font@warning#1{%
  \GenericWarning{%
     {(font)\@spaces\@spaces}%
     {Font Warning: #1}%
}
def\@font@info#1{%
  \GenericInfo{%
     (font)\space\@spaces
  }{%
     Font Info: #1%
  }%
}
```

(*End of definition for* `\@latex@error` *and others.*)

`\@latex@note`
`\@latex@note@no@line`

These are "info" messages that display on the terminal not just in the transcript.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                    {\@latex@note}{Display notes}%
\def\@latex@note#1{%
   \GenericWarning{%
      \@spaces\@spaces\@spaces
   }{%
      LaTeX Info: #1%
   }%
}
```

```
\def\@latex@note@no@line#1{%
  \@latex@note{#1\@gobble}}
```

We don't make them undefined but rather point to `\@latex@info` because that's what they replace. This way we can change `\@latex@info` elsewhere without the need to further rollback sections.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@latex@note}{Display notes}%
⟨latexrelease⟩
⟨latexrelease⟩\let\@latex@note\@latex@info
⟨latexrelease⟩\let\@latex@note@no@line\@latex@info@no@line
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@latex@note` *and* `\@latex@note@no@line`.)

`\c@errorcontextlines` `\errorcontextlines` as a LaTeX counter, so that it may be manipulated with `\setcounter` (once it is defined :-)

```
\let\c@errorcontextlines\errorcontextlines
\c@errorcontextlines=-1
```

(*End of definition for* `\c@errorcontextlines`.)

`\on@line` The message ' on input line n'.

```
\def\on@line{ on input line \the\inputlineno}
```

(*End of definition for* `\on@line`.)

`\@warning` `\@@warning` `\@latexerr` Older LaTeX messages. For the moment, these `\let` to the new message commands. They may be changed later, once only obsolete packages and classes contain them.

```
\let\@warning\@latex@warning
\let\@@warning\@latex@warning@no@line
\global\let\@latexerr\@latex@error
```

(*End of definition for* `\@warning`, `\@@warning`, *and* `\@latexerr`.)

`\@spaces` Four spaces.

```
\def\@spaces{\space\space\space\space}
```

(*End of definition for* `\@spaces`.)

1.2 Specific errors

`\@eha` `\@ehb` `\@ehc` `\@ehd` The more common error help messages.

```
\gdef\@eha{%
  Your command was ignored.\MessageBreak
  Type \space I <command> <return> \space to replace it %
  with another command,\MessageBreak
  or \space <return> \space to continue without it.}
\gdef\@ehb{%
  You've lost some text. \space \@ehc}
\gdef\@ehc{%
  Try typing \space <return> %
```

```
  \space to proceed.\MessageBreak
  If that doesn't work, type \space X <return> \space to quit.}
\gdef\@ehd{%
  You're in trouble here.  \space\@ehc}
```

(*End of definition for* `\@eha` *and others.*)

`\@notdefinable` Error message generated in `\@ifdefinable` from calls to one of the commands `\newcommand`, `\newlength` or `\newtheorem` specifying an already-defined command name or one that begins `\end`....

```
\gdef\@notdefinable{%
 \@latex@error{%
   Command \@backslashchar\reserved@a\space
   already defined.\MessageBreak
   Or name \@backslashchar\@qend... illegal,
   see p.192 of the manual}\@eha}
```

(*End of definition for* `\@notdefinable`.)

`\@nolnerr` Generated by `\newline` and `\\` when called in vertical mode.

```
\gdef\@nolnerr{%
  \@latex@error{There's no line here to end}\@eha}
```

(*End of definition for* `\@nolnerr`.)

`\@nocounterr` Generated by `\setcounter`, `\addtocounter` or `\newcounter` if applied to an undefined counter ⟨*cnt*⟩.

`\@nocnterr` Obsolete error message generated in LaTeX2.09 by `\setcounter`, `\addtocounter` or `\newcounter` for undefined counter. DO NOT use for LaTeX 2ε it MIGHT vanish! Use `\@nocounterr{`⟨*cnt*⟩`}` instead.

```
\gdef\@nocounterr#1{%
  \@latex@error{No counter '#1' defined}\@eha}
\gdef\@nocnterr{\@nocounterr?}
```

(*End of definition for* `\@nocounterr` *and* `\@nocnterr`.)

`\@ctrerr` Called when trying to print the value of a counter numbered by letters that's greater than 26.

```
\gdef\@ctrerr{%
  \@latex@error{Counter too large}\@ehb}
```

(*End of definition for* `\@ctrerr`.)

`\@nodocument` Error produced if paragraphs are typeset in the preamble.

```
\gdef\@nodocument{%
  \@latex@error{Missing \protect\begin{document}}\@ehd}
```

(*End of definition for* `\@nodocument`.)

\@badend Called by \end that doesn't match its \begin. RmS 1992/08/24: added code to \@badend to display position of non-matching \begin. FMi 1993/01/14: missing space added.

The environment name has to literally match, i.e., what is stored in \@currenvir (after one expansion) must match what is passed to \end (without expansion). If not we complain. Not the absolute best solution but at least it avoids getting \begin{foo} ended by \end{foo} which was possible in the past.

```
\gdef\@badend#1{%
  \@latex@error{\protect\begin
     {\detokenize\expandafter{\@currenvir}}\@currenvline
     \space ended by \protect\end{\detokenize{#1}}}\@eha}
```

(*End of definition for* \@badend.)

\@badmath Called by \[, \], \(or \) when used in wrong mode.

```
\gdef\@badmath{%
  \@latex@error{Bad math environment delimiter}\@eha}
```

(*End of definition for* \@badmath.)

\@toodeep Called by a list environment nested more than six levels deep, or an enumerate or itemize nested more than four levels.

```
\gdef\@toodeep{%
  \@latex@error{Too deeply nested}\@ehd}
```

(*End of definition for* \@toodeep.)

\@badpoptabs Called by \endtabbing when not enough \poptabs have occurred, or by \poptabs when too many have occurred.

```
\gdef\@badpoptabs{%
  \@latex@error{\protect\pushtabs\space and \protect\poptabs
      \space don't match}\@ehd}
```

(*End of definition for* \@badpoptabs.)

\@badtab Called by \>, \+ , \- or \< when stepping to an undefined tab.

```
\gdef\@badtab{%
 \@latex@error{Undefined tab position}\@ehd}
```

(*End of definition for* \@badtab.)

\@preamerr This error is special: it appears in places where we normally have to \protect expansions. However, to prevent a protection of the error message itself (which would result in the message getting printed not issued on the terminal) we need to locally reset \protect to \relax.

```
\gdef\@preamerr#1{%
  \begingroup
    \let\protect\relax
    \@latex@error{\ifcase #1 Illegal character\or
     Missing @-exp\or Missing p-arg\fi\space
     in array arg}\@ehd
  \endgroup}
```

(*End of definition for* \@preamerr.)

\@badlinearg Occurs in \line and \vector command when a bad slope argument is encountered.

```
\gdef\@badlinearg{%
  \@latex@error{%
      Bad \protect\line\space or \protect\vector
      \space argument}\@ehb}
```

(*End of definition for* \@badlinearg.)

\@LRmoderr A command is not allowed in restricted horizontal mode, i.e., in LR-mode in LaTeX terminology.

```
\gdef\@LRmoderr{%
  \@latex@error{Not allowed in LR mode}\@ehb}
```

(*End of definition for* \@LRmoderr.)

\@parmoderr Occurs in a float environment or a \marginpar when encountered in inner vertical mode.

```
\gdef\@parmoderr{%
  \@latex@error{Not in outer par mode}\@ehb}
```

(*End of definition for* \@parmoderr.)

\@fltovf Occurs in float environment or \marginpar when there are no more free boxes for storing floats.

```
\gdef\@fltovf{%
  \@latex@error{Too many unprocessed floats}\@ehb}
```

(*End of definition for* \@fltovf.)

\@latexbug Occurs in output routine. This is bad news.

```
\gdef\@latexbug{%
  \@latex@error{This may be a LaTeX bug}{Call for help}}
```

(*End of definition for* \@latexbug.)

\@badcrerr This error was removed and replaced by \@nolnerr.

```
%\def\@badcrerr {\@latex@error{Bad use of \protect\\}\@ehc}
```

(*End of definition for* \@badcrerr.)

\@noitemerr \addvspace or \addpenalty was called when not in vmode. Probably caused by a missing \item.

```
\gdef\@noitemerr{%
  \@latex@error{Something's wrong--perhaps a missing %
      \protect\item}\@ehc}
```

(*End of definition for* \@noitemerr.)

\@notprerr A command that can be used only in the preamble appears after the command \begin{document}.

```
\gdef\@notprerr{%
  \@latex@error{Can be used only in preamble}\@eha}
```

(*End of definition for* \@notprerr.)

\@inmatherr Issued by commands that don't work correctly within math (like \item). There is no real error recovery happening, e.g., the user might get additional errors afterwards.

```
\gdef\@inmatherr#1{%
   \relax
   \ifmmode
    \@latex@error{Command \protect#1 invalid in math mode}\@ehc
   \fi}
```

(*End of definition for* \@inmatherr.)

\@invalidchar An error for use with invalid characters. This is commented out, since we decided to use catcode 15 instead.

```
%\def\@invalidchar{\@latex@error{Invalid character in input}\@ehc}
```

(*End of definition for* \@invalidchar.)

As well as the above error commands some error messages are directly coded to save space. The messages already present in LaTeX2.09 include:

`Environment --- undefined`

Issued by \begin for undefined environment.

`Tab overflow`

Occurs in \= when maximum number of tabs exceeded.

`\< in mid line`

Occurs in \< when it appears in middle of line.

`Float(s) lost`

In output routine, caused by a float environment or \marginpar occurring in inner vertical mode.

1.3 Tracing

The trace package implements the commands \traceon and \traceoff that work similar to \tracingall but skip certain code blocks that produce a lot of tracing output being of no interest during debugging (for example loading a font). Code blocks that should be hidden during tracing need to be surrounded by the macros \conditionally@traceoff and \contionally@traceon.

For the kernel code the trace package then redefines a number of macros to include this tracing support.

However, in order to allow any macro package to react to \traceon we also provide dummy definitions for the two commands in the kernel so that they can be used by external packages without the need to distinguish between trace being loaded or not.

\conditionally@traceon
\conditionally@traceoff

These are only dummy definitions. For details see the trace package.

```
\let\conditionally@traceon\@empty
\let\conditionally@traceoff\@empty
```

(*End of definition for* \conditionally@traceon *and* \conditionally@traceoff.)

⟨/2ekernel⟩

File 15
ltpar.dtx

1 Paragraphs

This section of the kernel declares the commands used to set \par and \everypar when ever their function needs to be changed for a long time.

This file here describes the interfaces that have been in the kernel forever, used to implement the scenarios described below. They remain valid but are now augmented in the next file (ltpara.dtx) to add hooks to paragraphs. At some point we will consolidate the two files further.

There are two situations in which \par may be changed:

- Long-term changes, in which the new value is to remain in effect until the current environment is left. The environments that change \par in this way are the following:
  - All list environments (itemize, quote, etc.)
  - Environments that turn \par into a noop: tabbing, array and tabular.
- Temporary changes, in which \par is restored to its previous value the next time it is executed. The following are all such uses.
  - \end when preceded by \@endparenv, which is called by \endtrivlist
  - The mechanism for avoiding page breaks and getting the spacing right after section heads.

1.1 Implementation

\@setpar To permit the proper interaction of these two situations, long-term changes are made by the \@setpar{⟨*VAL*⟩} command. It's function is:

To set \par. It \def's \par and \@par to ⟨*VAL*⟩.

\@restorepar Short-term changes are made by the usual \def\par commands. The original values are restored after a short-term change by the \@restorepar commands.

\@@par \@@par always is defined to be the original TEX \par.

\everypar \everypar is changed only for the short term. Whenever \everypar is set non-null, it should restore itself to null when executed.

The following commands change \everypar in this way:

- \item
- \end when preceded by \@endparenv, which is called by endtrivlist
- \minipage

When dealing with \par and \everypar remember the following two warnings:

1. Commands that make short-term changes to \par and \everypar must take account of the possibility that the new commands and the ones that do the restoration may be executed inside a group. In particular, \everypar is executed inside a group whenever a new paragraph begins with a left brace. The \everypar command that restores its definition should be local to the current group (in case the command is inside a minipage used inside someplace where \everypar has been redefined). Thus, if \everypar is redefined to do an \everypar{} it could take several executions of \everypar before the restoration "holds". This usually causes no problem. However, to prevent the extra executions from doing harm, use a global switch to keep anything harmful in the new \everypar from being done twice.

2. Commands that change \everypar should remember that \everypar might be supposed to set the following switches false:

   - @nobreak
   - @minipage

   they should do the setting if necessary.

```
⟨*2ekernel⟩
\message{par,}
```

\@setpar
\@par

Initiate a long-term change to \par.

```
\def\@setpar#1{\def\par{#1}\def\@par{#1}}
```

The default definition of \@par will ensure that if \@restorepar defines \par to execute \@par it will redefine itself to the primitive \@@par after one iteration.

```
\def\@par{\let\par\@@par\par}
```

(*End of definition for* \@setpar *and* \@par.)

\@restorepar

Restore from a short-term change to \par.

```
\def\@restorepar{\def\par{\@par}}
⟨/2ekernel⟩
```

(*End of definition for* \@restorepar.)

File 16
ltpara.dtx

Abstract

This code defines four special kernel hooks to support paragraph tagging as well as four public hooks which can be occasionally useful.

1 Introduction

The building of paragraphs in the TEX engine(s) has a number of peculiarities that makes it on one hand fairly flexible but on the other hand somewhat awkward to control or reliably to extend. Thus to better understand the code below we start with a brief introduction of the mechanism; for more details refer to the TEXbook [**?**, chap. 14] (for the full truth you may even have to study the program code).

1.1 The default processing done by the engine

TEX automatically starts building a paragraph when it is currently in vertical mode and encounters anything that can only live in horizontal mode. Most often this is a character, but there are also many commands that can be used only in horizontal mode. If any of them is encountered, TEX will immediately back up (i.e., the character or command is read later again), adds a `\parskip` glue to the current vertical list unless the list is empty, switches to horizontal mode, starts its special "start of paragraph processing" and only then rereads the character or command that caused the mode change.[23]

This "start of paragraph processing" first adds an empty box at the start of the horizontal list of width `\parindent` (which represents the paragraph indentation) unless the paragraph was started with `\noindent` in which case no such box is added[24]. It then reads and processes all tokens stored in the special engine token register `\everypar`. After that it reads and processes whatever has caused the paragraph to start.

Thus out of the box, TEX offers the possibility to put some special code into `\everypar` to gain control at (more or less) the start of the paragraph. For example, in LaTeX and a number of packages, special code like the following is sometimes used:

```
\everypar{{\setbox\z@\lastbox}\everypar{} ...}
```

This removes the paragraph indentation box again (that was already placed by TEX), then resets `\everypar` so that it doesn't do anything on the next paragraph start and then does whatever it wants to do, e.g., in an `\item` of a list it will typeset the label in front of the paragraph text. However, there is only one such `\everypar` token register and if different packages and/or the kernel all attempt to add their own code here, coordination is very difficult if not impossible.

The process when the paragraph ends has different mechanisms and interfaces. A paragraph ends when the engine primitive `\par` is called while TEX is in unrestricted horizontal mode, i.e., is building a paragraph. At other times this primitive does nothing or generates as an error depending on the mode TEX is in, e.g., the `\par` in `\hbox{a\par b}` is ignored, but `$a\par b$` would complain.

[23] Already not quite true: the command `\noindent` starts the paragraph but influences the special processing by suppressing the paragraph indentation box normally inserted by it.

[24] That's a bit different from placing a zero-sized box!

If this primitive ends the paragraph it does some special "end of horizontal list" processing, then calls TEX's paragraph builder; this breaks the horizontal list into lines and then these lines are added as boxes to the enclosing vertical list and TEX returns to vertical mode.

This \par command can be given explicitly, but there are also situations in which TEX is generating it on the fly. Most often this happens when TEX encounters a blank line which is automatically changed to a \par command which is then executed. The other possibility is that TEX encounters a command which is incompatible with horizontal processing, e.g., \vskip (a request for adding vertical space). In such cases it silently backs up, and inserts a \par in the hope that this gets it out of horizontal mode and makes the vertical command acceptable.

The important point to note here is that TEX really inserts the command with the name \par, which can be redefined. Thus, it may not have its original "primitive" meaning and therefore may not end the horizontal list and call the paragraph builder. This approach offers some flexibility but also allows you to easily produce a TEX document that loops forever, for example, the simple line

```
A \let\par\relax \vskip
```

will start a horizontal list at A, redefines \par, then sees \vskip and inserts \par to end the paragraph. But this now only runs \relax so nothing changes and \vskip is read again, issues a \par which .... In short, it only takes a plain TEX document with five tokens to run forever (since no memory is consumed and therefore eventually exhausted).

There is no way other than changing \par to gain control at the end of a paragraph, i.e., there is no token list like \everypar that is inserted. Hence the only way to change the default behavior is to modify the action that \par executes, with similar issues as outlined before: different processes need to ensure that they do not overwrite their modifications or worse, think that the \par in front of them is the engine primitive while in fact it has already been changed by other code.

To make matters slightly worse there are a few places where TEX handles the situation differently (most likely for speed reasons back when computers were much slower). If TEX finds itself in unrestricted horizontal mode at the end of building a vertical box (for an \insert, \vadjust or executing the output routine code), it will finish the horizontal list not by issuing a \par command (which would be consistent with all other places) but by simply executing the primitive meaning of \par, regardless of the actual definition that \par has at the time.

Thus, if you have carefully crafted a redefined \par to execute some special actions at the end of a paragraph and you write something like

```
\vbox{Some paragraph ... text.}
```

you will find that your code does not get run for the last paragraph in that box. LATEX avoids this problem, by making sure that its boxes (such as \parbox or the minipage environment, etc.) all internally add an explicit \par at the end so that such code is run and TEX finds itself in vertical mode already without the need to start up the paragraph builder internally. But, of course, this only works for boxes under direct control of the LATEX kernel; if some package uses low-level \vboxes without adding this precaution the TEX optimization kicks in and no special \par code is executed.

And there is another optimization that is painful: if a paragraph is interrupted by a mathematical display, e.g., \[...\] in LATEX or $$...$$ in plain TEX, then TEX will resume horizontal mode afterward, i.e., it will start to build a new horizontal list

without inserting an indentation box or \everypar at that point. However, if that list immediately ends with an explicit or implicit \par then TEX will simply throw away this "null" paragraph and not do its usual "end of horizontal list" processing, so this special case also needs to be accounted for when introducing any extended processing.

2 The new mechanism implemented for LATEX

To improve the situation (and also to support automatic tagging of PDF documents) we now offer public as well as private hooks at the start and end of the paragraph processing. The public hooks can be used by packages (or by the user in the preamble or within the document) and using the hook mechanisms it is possible to reorder or arrange code from different packages in such a way that these can safely coexist.

To make that happen we have to make use of the basic functionality that is offered by TEX, e.g., we install special code inside \everypar to provide hooks at the beginning and we redefine \par to do some special processing when appropriate to install hooks at the end of the paragraph.

In order to make this work, we have to ensure that package use of \everypar is not overwriting our code. This is done through a trick: we basically hide the real \everypar from the packages and offer them a new token register (with the same name). So if they install their own code it doesn't overwrite ours. Our code then inserts the new \everypar at the right place inside the process so that it looks as if it was the primitive \everypar.[25]

At the end of the paragraph it would be great if we could use a similar trick. However, due to the fact that TEX inserts the token \par (that doesn't have a defined meaning) we can't hide "the real thing™" and offer the package an indistinguishable alternate.

Fortunately, LATEX has already redefined \par for its own purposes. As a result there aren't many packages that attempt to change \par, because without a lot of extra care that would fail miserably. But the bottom line is that, if you load a package that alters \par then the end of paragraph hooks are most likely not executing while that redefinition is active.[26]

[25] Ideally, \everypar wouldn't be used at all by packages and instead they would simply write their code into the hooks now offered by the kernel. However, while this is the longterm goal and clearly an improvement (because then the packages do no longer need to worry about getting their code overwritten or needing to account for already existing code in \everypar), this will not happen overnight. For that reason support for this legacy method is retained.

[26] Similarly to the \everypar situation, the remedy is that such packages stop doing this and instead add their alterations into the paragraph hooks now provided.

2.1 The provided hooks

para/before
para/begin
para/end
para/after

The following four public hooks are defined and executed for each paragraph:

para/before This hook is executed after the kernel hook `\@kernel@before@para@before` (discussed below) in vertical mode immediately after TEX has contributed `\parskip` to the vertical list and before the actual paragraph processing in horizontal mode starts.

This hook should either not produce any typeset material or add only vertical material. If it starts a paragraph an error is generated. The reason is that we are in the starting process of processing a paragraph and so this would lead to endless recursion.[27]

para/begin This hook is executed after the kernel hook `\@kernel@before@para@begin` (discussed below) in horizontal mode immediately before the indentation box is placed (if there is any, i.e., if the paragraph hasn't been started with `\noindent`).

The indentation box to be typeset is available to the hook as `\IndentBox` and its automatic placement (after the hook is executed) can be prevented through `\OmitIndent`. More precisely `\OmitIndent` voids the box.

The indentation box is then typeset directly after the hook execution by something equivalent to `\box\IndentBox` followed by the current content of the token register `\everypar` that it is available to the kernel or to packages (that run some legacy code).

One has to be careful not to add any code to the hook that starts its own paragraph (e.g., by adding a `\parbox` or a `\marginpar` inside) because that would call the hook inside again (as a new paragraph is started there) and thus lead to an endless recursion ending only after exhausting the available memory. This can only be done by making sure that is not executed for the inner paragraphs (or at least not recursively forever).

para/end This hook is executed at the end of a paragraph when TEX is ready to return to vertical mode and after it has removed the last horizontal glue (but not any kerns) placed on the horizontal list. The code is still executed in horizontal mode so it is possible to add further horizontal material at this point, but it should not alter the mode (even a temporary exit from horizontal mode would create chaos—any attempt will cause an error message)! After the hook has ended the kernel hook `\@kernel@after@para@end` is executed and then TEX returns to vertical mode.

The hook is offered as public hook, but because of the requirement to stay within horizontal mode one needs to be careful in what is placed into the hook.[28]

This hook is implemented as a reversed hook.

para/after This hook is executed directly after TEX has returned to vertical mode and after any material that migrated out of the horizontal list (e.g., from a `\vadjust`) has processed.

[27] One could allow it but only if the newly started paragraph is processed without any hooks. Furthermore correct spacing would be a bit of a nightmare so for now this is forbidden.

[28] Maybe we should guard against that, but it would be rather tricky to implement as mode changes can happen across group boundaries so one would need to keep a private stack just for that. Well, something to ponder.

This hook should either not produce any typeset material or add only vertical material. However, for this hook starting a new paragraph is not a disaster so that it isn't prevented.

This hook is implemented as a reversed hook.

Once that hook code has been processed the kernel hook `\@kernel@after@para@after` is executed as the final action of the paragraph processing.

`\@kernel@before@para@before`
`\@kernel@after@para@after`
`\@kernel@before@para@begin`
`\@kernel@after@para@end`

As already mentioned above there are also four kernel hooks that are executed at the start and end of the processing.

`\@kernel@before@para@before` For future extensions, not currently used by the kernel.

`\@kernel@after@para@after` For future extensions, not currently used by the kernel.

`\@kernel@before@para@begin` Used by the kernel to implement tagging. This hook is executed at the very beginning of a paragraph after TEX has switched to horizontal mode but before any indentation box got added or any `\everypar` was run.

It should not generate typeset material that could alter the position. Note that it should never leave hmode, otherwise you will end with a loop! We could guard against this, but since it is an internal kernel hook that shouldn't be touched this isn't checked.

`\@kernel@after@para@end` Used by the kernel to implement tagging. It is executed directly after the public `para/end` hook. After it there is a quick check that we are still in horizontal mode, i.e., that the public hook has not mistakenly ended horizontal mode prematurely (this is an incomplete check just testing the mode and could perhaps be improved (at the cost of speed)).

2.2 Altered and newly provided commands

`\par`
`\endgraf`
`\para_end:`

An explicit request for ending a paragraph is provided in plain TEX under the name `\endgraf`, which simply uses the primitive meaning (regardless of what `\par` may have as its current definition). In LATEX `\endgraf` (with that behavior) was originally also available.

With the new paragraph handling in LATEX, ending a paragraph means a bit more than just calling the engine's paragraph builder: the process also has to add any hook code for the end of a paragraph. Thus `\endgraf` was changed to provide this additional functionality (along with `\par` remaining subject to its current meaning).

The expl3 name for this functionality is `\para_end:`.

Note: *The next two commands are still under discussion and may slightly change their semantics (as described in the document) and/or their names between now and the 2021 Spring release!*

\OmitIndent
\para_omit_indent:

Inside the para/begin hook one can use this command to suppress the indentation box at the start of the paragraph. (Technically it is possible to use this command outside the hook as well, but this should not be relied upon.) The box itself remains available for use.

The expl3 name for the function is \para_omit_indent:.

\IndentBox
\g_para_indent_box

The box register holding the indentation box for the paragraph is available for inspection (or changes) inside hooks. It remains available even if the \OmitIndent command was used; in that case it will just not be automatically placed.

The expl3 name for the box register is \g_para_indent_box.

\RawIndent
\para_raw_indent:
\RawNoindent
\para_raw_noindent:
\RawParEnd
\para_raw_end:

\RawIndent *hmode material* \RawParEnd
\RawNoindent *hmode material* \RawParEnd

The commands \RawIndent and \RawNoindent are not meant for normal paragraph building (where the result is a textual paragraph in the traditional meaning of the word), but for special cases where TEX's low-level algorithm is used to achieve special effects, but where the result is not a "paragraph".

They are called "raw", because they bypass LATEX's hook mechanism for paragraphs and simply invoke the low-level TEX algorithm. I.e., they are like the original TEX primitives \indent and \noindent (that is they execute no hooks other than \everypar) except that they can only be used in vertical mode and generate an error if found elsewhere.

To avoid issues a paragraph started by them should always be ended by \RawParEnd[29] and not by \par (or a blank line), because the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired. This also means that one should not put arbitrary user content between these commands if that content could contain stray \pars.

The expl3 names for the functions are \para_raw_indent:, \para_raw_indent: and \para_raw_end:.

2.3 Examples

None of the examples in this section are meant for real use as they are far too simple-minded but they should give some ideas of what could be possible if a bit more care is applied.

2.3.1 Testing the mechanism

The idea is to output for each paragraph encountered some information: a paragraph sequence number, a level number in roman numerals, the environment in which this paragraph appears, and the line number where the start or end of the paragraph is, e.g., something like

[29] Technical note for those who know their *TEXbook*: the \RawParEnd command invokes the original TEX engine definition of \par that (solely) triggers the paragraph builder in TEX when found inside unrestricted horizontal mode and does nothing in other processing modes.

```
PARA: 1-i start (document env. on input line 38)
PARA: 1-i end   (document env. on input line 38)
PARA: 2-i start (document env. on input line 40)
PARA: 3-ii start (minipage env. on input line 40)
PARA: 3-ii end   (minipage env. on input line 40)
PARA: 2-i end   (document env. on input line 41)
```

As you can see paragraph 2 starts on line 40 and ends on 41 and inside a minipage started paragraph 3 (start and end on line 40). If you run this on some document you will find that LaTeX considers more things "a paragraph" than you have probably thought.

This was generated by the following hook code:

```
\newcounter{paracnt}         % sequence counter
\newcounter{paralevel}       % level counter
```

To support paragraph nesting we need to maintain a stack of the sequence numbers. This is most easily done using expl3 functions, so we switch over. This is not a very general implementation, just enough for what we need and a bit of LaTeX 2_ε thrown in as well. When popping, the result gets stored in `\paracntvalue` and the `\ERROR` should never happen because it means we have tried to pop from an empty stack.

```
\ExplSyntaxOn
\seq_new:N \g_para_seq
\cs_new:Npn \ParaPush
  {\seq_gpush:No  \g_para_seq {\the\value{paracnt}}}
\cs_new:Npn \ParaPop  {\seq_gpop:NNF  \g_para_seq \paracntvalue \ERROR }
\ExplSyntaxOff
```

At the start of the paragraph increment both sequence counter and level and also save the then current sequence number on our stack.

```
\AddToHook{para/begin}{%
  \stepcounter{paracnt}\stepcounter{paralevel}%
  \ParaPush
```

To display the sequence number we `\typeout` the current sequence and level number. The command `\@currenvir` gives us the current environment and `\on@line` produces a space and the current input line number.

```
  \typeout{PARA: \arabic{paracnt}-\roman{paralevel} start
    (\@currenvir\space env.\on@line)}%
```

We also typeset the sequence number as a tiny red number in a box that takes up no horizontal space. This helps us seeing where LaTeX sees the start and end of the paragraphs in the document.

```
  \llap{\color{red}\tiny\arabic{paracnt}\ }%
}
```

At the end of the paragraph we display sequence number and level again. The level counter has the correct value but we need to retrieve the right sequence value by popping it off the stack after which it is available in `\paracntvalue` the way we have set this up above.

```
\AddToHook{para/end}{%
  \ParaPop
  \typeout{PARA: \paracntvalue-\roman{paralevel} end \space\space
    (\@currenvir\space env.\on@line)}%
```

We also typeset again a tiny red number with that value, this time sticking out to the right.[30] We also decrement the level counter since our level has finished.

```
  \rlap{\color{red}\tiny\ \paracntvalue}%
  \addtocounter{paralevel}{-1}%
}
\makeatother
```

2.3.2 Mark the first paragraph of each `itemize`

The code for this is rather simple. We supply some code that is executed only once inside a hook at the start of each `itemize`. We explicitly change the color back and forth so that we don't introduce grouping around the paragraph.

```
\AddToHook{env/itemize/begin}{%
  \AddToHookNext{para/begin}{\color{blue}}%
  \AddToHookNext{para/end}{\color{black}}%
}
```

As a result the first paragraph of each `itemize` will appear in blue.

2.4 Some technical notes

The code tries hard to be transparent for package code, but of course any change means that there is a potential for breaking other code. So in section we collect a few cases that may be of importance if low-level code is dealing with paragraphs that are now behaving slightly differently. The notes are from issues we observed and will probably grow over time.

2.4.1 Glue items between paragraphs (found with fancypar)

In the past LaTeX placed two glue items between two consecutive paragraphs, e.g.,

```
text1 \par text2 \par
```

would show something like

```
\glue(\parskip) 0.0 plus 1.0
\glue(\baselineskip) 5.16669
```

but now there is another `\parskip` glue (that is always 0pt):

```
\glue(\parskip) 0.0 plus 1.0
\glue(\parskip) 0.0
\glue(\baselineskip) 5.16669
```

[30] Note that this can alter the document pagination, because a paragraph ending in a display (e.g., an equation) will get an extra line—in that case our tiny number has an effect even though it doesn't take up any space, because it paragraph is no longer empty and thus isn't dropped!

The reason is that we generate a "fake" paragraph to gain control and safely add the early hooks, but this generates an additional glue item. That item doesn't contribute anything vertically but if somebody writes code that unravels a constructed list using `\lastbox`, `\unskip` and `\unpenalty` then the code has to remove one additional glue item or else it will fail.

3 The Implementation

```
⟨@@=para⟩
⟨*2ekernel | latexrelease⟩
\ExplSyntaxOn
⟨latexrelease⟩\NewModuleRelease{2021/06/01}{ltpara}
⟨latexrelease⟩                    {Paragraph~handling~and~hooks}
```

3.1 Providing hooks for paragraphs

para/before
para/after
para/begin
para/end

The public hooks. They are implemented as a paired set of hooks.

```
\hook_new_pair:nn{para/before}{para/after}
\hook_new_pair:nn{para/begin}{para/end}
```

(*End of definition for* `para/before` *and others. These functions are documented on page 404.*)

\@kernel@before@para@before
\@kernel@after@para@after
\@kernel@before@para@begin
\@kernel@after@para@end

The corresponding kernel hooks (for tagging and future extensions).

```
\let \@kernel@before@para@before \@empty
\let \@kernel@before@para@begin  \@empty
\let \@kernel@after@para@end     \@empty
\let \@kernel@after@para@after   \@empty
```

(*End of definition for* `\@kernel@before@para@before` *and others. These functions are documented on page 405.*)

\g__para_standard_everypar_tl

Whenever TEX starts a paragraph it inserts first an indentation box and then executes the tokens stored in `\tex_everypar:D` (known to LATEX as `\everypar`). We alter this behavior slightly here, so that hooks are added into the right place. Otherwise the process change remains transparent to any legacy code for this space.

We keep the standard code to be used by `\tex_everypar:D` in a separate token list because we have to switch back and forth for error recovery and so altering `\tex_everypar:D` all the time should be a tiny bit faster.

```
⟨latexrelease⟩\IncludeInRelease{2023/06/01}
⟨latexrelease⟩        {\g__para_standard_everypar_tl}{minipage~ fix}
\tl_new:N \g__para_standard_everypar_tl
```

Here is now its definition:

```
\tl_gset:Nn \g__para_standard_everypar_tl {
```

First we remove the indentation box and store it in `\g_para_indent_box`. If there was none because the paragraph was started by `\noindent` the box register will be void.

```
  \box_gset_to_last:N \g_para_indent_box
```

This will make the newly started horizontal list empty, so if we stop it now and return to vertical mode it will be dropped by TEX. We do that but inside a group so that any `\parshape` settings will not get lost as we need them for later.

```
  \group_begin:
    \tex_par:D
  \group_end:
```

We then change \tex_everypar:D to generate an error so that we can detect and report if the para/before hook illegally changed out of vmode.

```
\tex_everypar:D { \msg_error:nnnn { hooks }{ para-mode }{before}{vertical} }
\@kernel@before@para@before
\hook_use:n {para/before}
```

Assuming the hooks have been well behaved it is time to return to horizontal mode and start the paragraph in earnest. We already have the indentation box saved away so we now have to restart the paragraph with an empty \tex_everypar:D and with \tex_noindent:D. And we need to make sure not to get another \parskip or rather (since we can't prevent that) that it is of zero size.

```
\group_begin:
  \tex_everypar:D {}
```

There has been a long-standing problem with LaTeX's minipages in that invisible material at the beginning of a minipage (such as a \color setting) would result in \parskip being added in front of the first paragraph—something that is not done by TeX if a vertical list is completely empty. As this is happening on a very low-level in the engine it wasn't really possible to find out if this \parskip was added or if a space we see in front of the current point is legitimate. However, with the new paragraph handling we are in a better position: while we still don't know if there is such a space or not, we do know if we have just created an empty paragraph. Thus, if we now set \parskip to -\parskip the two will cancel each other if present and if the first was ignored because the vertical list was empty, then the second will be ignored too because it is still empty. Of course, we don't want to cancel always but only at the start of a minipage and that is signaled with the @minipage switch.

```
  \skip_set:Nn \tex_parskip:D
       { \if@minipage -\tex_parskip:D \else: \c_zero_skip \fi: }
  \tex_noindent:D
\group_end:
```

That brings us back to the start of the horizontal list but we need to change \tex_everypar:D back to its normal content in case there are nested paragraphs coming up.

```
\tex_everypar:D{\g__para_standard_everypar_tl}
```

This is followed by executing the kernel and the public hook. The kernel hook is there to enable tagging.

```
\@kernel@before@para@begin
\hook_use:n {para/begin}
```

If we aren't in horizontal mode any longer the hooks above misbehaved.

```
\if_mode_horizontal: \else:
  \msg_error:nnnn { hooks }{ para-mode }{begin}{horizontal} \fi:
```

Finally we reinsert the indentation box (unless suppressed) and then call \everypar the way legacy LaTeX code expects it.

However, adding the public \everypar is a bit tricky (see below) so we add that later, and indirectly.

```
\__para_handle_indent:
% \the \everypar           % <--- done differently below
}
```

```
⟨latexrelease⟩\cs_set:Npn \__para_tmp:w #1#2#3#4#5 { }
⟨latexrelease⟩\tl_gput_right:Nx \g__para_standard_everypar_tl {
⟨latexrelease⟩  \exp_not:N \the
⟨latexrelease⟩  \exp_not:N \toks
⟨latexrelease⟩  \exp_after:wN \__para_tmp:w \token_to_meaning:N \everypar
⟨latexrelease⟩  \c_space_tl
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/06/01}
⟨latexrelease⟩       {\g__para_standard_everypar_tl}{minipage~ fix}
⟨latexrelease⟩
⟨latexrelease⟩\tl_gset:Nn \g__para_standard_everypar_tl {
⟨latexrelease⟩  \box_gset_to_last:N \g_para_indent_box
⟨latexrelease⟩  \group_begin:
⟨latexrelease⟩    \tex_par:D
⟨latexrelease⟩  \group_end:
⟨latexrelease⟩  \tex_everypar:D { \msg_error:nnnn { hooks }{ para-mode }{before}{vertical} }
⟨latexrelease⟩  \@kernel@before@para@before
⟨latexrelease⟩  \hook_use:n {para/before}
⟨latexrelease⟩  \group_begin:
⟨latexrelease⟩    \tex_everypar:D {}
⟨latexrelease⟩    \skip_zero:N \tex_parskip:D
⟨latexrelease⟩    \tex_noindent:D
⟨latexrelease⟩  \group_end:
⟨latexrelease⟩  \tex_everypar:D{\g__para_standard_everypar_tl}
⟨latexrelease⟩  \@kernel@before@para@begin
⟨latexrelease⟩  \hook_use:n {para/begin}
⟨latexrelease⟩  \if_mode_horizontal: \else:
⟨latexrelease⟩    \msg_error:nnnn { hooks }{ para-mode }{begin}{horizontal} \fi:
⟨latexrelease⟩  \__para_handle_indent:
⟨latexrelease⟩}
```

We also have to add the `\everypar` toks register at the end. In case of rollback this is already allocated and we have to find out the correct number (hope this is correctly done)

```
⟨latexrelease⟩\cs_set:Npn \__para_tmp:w #1#2#3#4#5 { }
⟨latexrelease⟩\tl_gput_right:Nx \g__para_standard_everypar_tl {
⟨latexrelease⟩    \exp_not:N \the
⟨latexrelease⟩    \exp_not:N \toks
⟨latexrelease⟩    \exp_after:wN \__para_tmp:w \token_to_meaning:N \everypar
⟨latexrelease⟩    \c_space_tl
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\g__para_standard_everypar_tl`.)

`\tex_everypar:D` `\tex_everypar:D` then only has to execute `\g__para_standard_everypar_tl` by default.

```
\tex_everypar:D{\g__para_standard_everypar_tl}
```

(*End of definition for* `\tex_everypar:D`.)

`\everypar` Tokens inserted at the beginning of the paragraph are placed into `\everypar` inside legacy LaTeX code, e.g., by the list environments or by headings to handle `\clubpenalty`, etc.

Now this isn't any longer the primitive but simply a toks register used in the code above but to legacy LaTeX code that is transparent.

There is, however, a problem: a handful packages use exactly the same trick and replace the primitive with a token register and call the token register inside the renamed primitive. That is they assume that `\everypar` is the primitive and that it will still be called at the start of the paragraph even if renamed.

But if we have already replaced it by a token register then all they do is to give that token register a new name. Thus our code in `\tex_everypar:D` would call `\everypar` (which is now their token register) and the code that they added ends up in our token register which is then never used at all. A bit mind boggling I guess.

So what we have to do is not to call the token register `\everypar` by its name inside `\tex_everypar:D` but by using its actual register number.

```
\newtoks \everypar
```

After we have allocated a new toks register with the name `\everypar` the actual register number is available (briefly) inside `\allocationnumber`. So instead of `\the\everypar` we have to put `\the\toks`⟨*allocated number*⟩ at the end of `\tex_everypar:D`.

So what remains doing is to append a few tokens to the token list `\g__para_standard_everypar_tl` which we do now. We use `x` expansion here to get the value of `\allocationnumber` in, all the other tokens should not be expanded at this point.

One important point here is to terminate the register allocation number with a real space. This space will get swallowed up when the number is read. Anything else, such as `\scan_stop:` would remain in the input and that would mean that it would interfere with `\everypar` code that attempts to scan ahead to see how the paragraph text starts.

```
\tl_gput_right:Nx \g__para_standard_everypar_tl {
  \exp_not:N \the
  \exp_not:N \toks
  \the \allocationnumber
  \c_space_tl
}
```

(*End of definition for* `\everypar`.)

`\g_para_indent_box` For managing the indentation we need to provide a public accessible box register

```
\box_new:N \g_para_indent_box
```

(*End of definition for* `\g_para_indent_box`. *This function is documented on page* 406.)

`\__para_handle_indent:` Adding (typesetting) the indent box is straight forward. If it was emptied before it does nothing.

```
\cs_new:Npn \__para_handle_indent: {
  \box_use_drop:N \g_para_indent_box
}
```

The declaration `\para_omit_indent:` (or `\OmitIndent`) changes that to do nothing.

```
\cs_new:Npn \para_omit_indent: {
  \box_gclear:N \g_para_indent_box
}
```

(*End of definition for* `\__para_handle_indent:`.)

\IndentBox
\OmitIndent

The LaTeX 2ε names for the indentation box and for suppressing it for use in the para/begin hook.

```
\cs_set_eq:NN \IndentBox \g_para_indent_box
\cs_set_eq:NN \OmitIndent \para_omit_indent:
```

(*End of definition for* `\IndentBox` *and* `\OmitIndent`*. These functions are documented on page 406.*)

\para_end:

Adding hooks to the end of a paragraph is similar but here we need to alter the command that is used by TeX to end horizontal mode and return to vertical mode, i.e., \par.

This is a bit more complicated as this command can appear anywhere either explicitly or implicitly added by TeX in certain situations:

- when using \par in the code or the document
- when using a blank line (which is converted to \par)
- when TeX finds any commands incompatible with horizontal mode it issues a \par and then rereads the command.

Unfortunately, TeX has some (these days) unnecessary optimizations: if a \vbox ends and TeX is still in horizontal mode it simply exercises the paragraph builder instead of issuing a \par. It is therefore necessary for LaTeX to ensure that this case doesn't happen and all boxes internally have a \par command at their end.

This \par may or may not run the "par primitive" (which is always available as \tex_par:D in expl3); it is permissible to have a changed meaning and it is in fact changed by LaTeX in various ways at various points inside latex.ltx. For this LaTeX 2ε code has the following conventions: \@@par and \endgraf both refer to the default meaning (in the past this was the initex primitive) while \par is the current meaning which maybe does something else.

We are now going to change this default meaning to instead run \para_end:, which ultimately executes the initex primitive but additionally adds our hooks when appropriate. This way the change is again transparent to the legacy LaTeX 2ε code.

In most cases \para_end: should behave exactly like the primitive and we achieve this by simply expanding it to the primitive which is available to us as \tex_par:D. This way we don't have to care about whether TeX just does nothing (e.g., if in vertical mode already) or generates an error, etc.

```
\cs_new_protected:Npn \para_end: {
```

CCC Maybe needs more explanation. TEMP NOTE: What should happen if in outer hmode with an empty hlist?

The only case we care about is when we are in horizontal mode (i.e., doing typesetting) and not also in inner mode (i.e., making paragraphs and not building an \hbox.

```
\bool_lazy_and:nnT
    { \mode_if_horizontal_p: }
    { \bool_not_p:n { \mode_if_inner_p: } }
    { ...
```

Since this is executed for each and every paragraph in a document we try to stay as fast as possible, so we do not use the above construct but two conditionals instead. Using low-level \if_mode... conditions would be even faster but has the danger to conflict with conditionals in the user hooks.

If \para_end: is executed while TeX is currently doing a low-level assignment the test for horizontal mode may get executed as part of the assignment. That is normally not an issue but we just found one case where it is:

```
\afterassignment\lst@vskip\@tempskipa \z@ \par
```

If TEX is in hmode while that assignment happens then the `\par` is seen in hmode because in the above case the assignment may not be finished (one should have used `\z@skip`) and the `\lst@vskip` will get inserted into the middle of the conditional. The `\lst@vskip` then changes to vmode and you get a surprising error about the `para/end` hook having changed modes even if you don't have any hook code(!): it is the inserted `\lst@vskip` that is actually causing the change of mode.This is what happened when the output routines got started while a `lstlisting` environment (that redefines `\vskip` in this way) was active. This is really faulty coding, but we try to be proactive and guard the conditional so that any scanning is first stopped, thus:

```
\scan_stop:
\mode_if_horizontal:TF {
  \mode_if_inner:F {
```

In that case the action of the primitive would be to remove the last glue (but no kerns) from the horizontal list (constructed to form a paragraph) and then to append a penalty of 10000 and the `\parfillskip`; it then passes the whole list to the paragraph builder, which breaks it into lines and TEX then returns to vertical mode.

What we want to do is to add this hook code at the end of the horizontal list before any of the above happens. If there was a glue item at the end of the list then it should get removed before the hook code gets added so we have to arrange for this removal.

As in other similar cases, it may be best to add here a `\nobreak` in case the hook itself adds glue and thus creates a non-explicit and unwanted potential breakpoont. On the other hand (as has been argued) the code in the hook should perhaps have the responsibility for adding such a guard penalty in this casse. This needs further analysis and decisions (as in emails).

In either case, good documentation of these hooks is essential, covering what the hook may or should provide and all such related considerations convernimg the content.

There is not much point in checking if there was really a glue item at the end of the horizontal list, instead we simply try to remove one using `\tex_unskip:D`: if there wasn't one this will do nothing.

```
        \tex_unskip:D
```

We then execute the public hook (which may add some final typeset material) followed by the kernel hook that we need for adding tagging support. None of this is supposed to change the mode—at the moment we make only a very simple test for this, more devious changes go unnoticed, but too bad as they will then probably backfire badly.

```
        \hook_use:n{para/end}
        \@kernel@after@para@end
        \mode_if_horizontal:TF {
```

The final action (before getting to the point where `\tex_par:D` is called) is to add an extra glue item so that the primitive is prevented from removing intended glue (if there was some). If we don't do this and the horizontal list ends in several glue items we would end up removing two glue items instead of just the last one, which would be wrong. We use glue (rather than a kern) as that will be removed by the primitive.

There is however one other TEX optimization that hurts: in a sequence like this `$$ ... $$ \par` (with `\par` being the primitive) TEX will be in horizontal mode after the display, ready to receive further paragraph text, but since the `\par` follows immediately there is a "null" paragraph at the end and TEX simply throws that away. The space between `$$` and `\par` got already dropped during the display processing so the `\par` is

not removing any space and appending \parfillskip, instead it simply goes silently to vmode.

Now if we would have added something (to prevent glue removal) that would look to TEX like material after the display and so we would end up with an empty paragraph just containing a penalty and \parfillskip.

We therefore check if the current hlist does end in glue (\tex_lastnodetype:D has the value 11) and if so we add a zero-length guard skip which will be removed by the following \tex_par:D.

```
            \if_int_compare:w 11 = \tex_lastnodetype:D
              \tex_hskip:D \c_zero_dim
            \fi:
```

To run the para/after hook we first end the paragraph. This means that the \tex_par:D at the very end is unnecessary but executing it there unnecessarily is better than having code that tests for all the different mode possibilities.

```
            \tex_par:D
            \hook_use:n{para/after}
            \@kernel@after@para@after
          }
```

If we were not horizontal mode (the F case from above) then the earlier hook para/end must have been at fault, so we report that.

```
          { \msg_error:nnnn { hooks }{ para-mode }{end}{horizontal} }
```

Finally close out the nested conditionals.

```
      }
    }
```

And then we can use the primitive to truly end the paragraph.

```
    \tex_par:D
  }
```

(*End of definition for* \para_end:. *This function is documented on page* 405.)

\para_raw_indent:
\para_raw_noindent:
\para_raw_end:

The commands \para_raw_indent: and \para_raw_noindent: are like the primitives \indent and \noindent except that they can only be used in vertical mode.

To avoid issues a paragraph started by them should always be ended by \para_raw_end: and not by \para_end: or \par as the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired.

```
\cs_new:Npn \para_raw_indent: {
  \mode_if_vertical:TF
      {
        \tex_everypar:D {
          \box_gset_to_last:N \g_para_indent_box
          \tex_everypar:D { \g__para_standard_everypar_tl }
          \__para_handle_indent:
          \the\everypar }
      }
      { \msg_error:nn { latex2e }{ raw-para } }
  \tex_indent:D
}
```

```
\cs_new:Npn \para_raw_noindent: {
  \mode_if_vertical:TF
       {
         \tex_everypar:D {
           \tex_everypar:D { \g__para_standard_everypar_tl }
           \the\everypar }
       }
       { \msg_error:nn { latex2e }{ raw-para } }
  \tex_noindent:D
}
```

```
\cs_new_eq:NN \para_raw_end: \tex_par:D
```

(*End of definition for* `\para_raw_indent:`, `\para_raw_noindent:`, *and* `\para_raw_end:`. *These functions are documented on page 406.*)

`\RawIndent` `\RawNoIndent` `\RawParEnd`

The LaTeX 2_ε names for starting and ending a paragraph without adding any hooks.

```
\cs_set_eq:NN \RawIndent   \para_raw_indent:
\cs_set_eq:NN \RawNoindent \para_raw_noindent:
\cs_set_eq:NN \RawParEnd   \para_raw_end:
```

(*End of definition for* `\RawIndent`, `\RawNoIndent`, *and* `\RawParEnd`. *These functions are documented on page 406.*)

This ends the `para` module code.

```
⟨@@=⟩
```

`\par` `\endgraf` `\@@par`

Having the new default definition for `\par` we also have to set it up so that it gets used. This involves three commands: `\par`, `\@@par` (to which LaTeX resets `\par` occasionally) and `\endgraf`, which is another name for the "default" action of `\par`.

```
\cs_set_eq:NN \par     \para_end:
\cs_set_eq:NN \@@par   \para_end:
\cs_set_eq:NN \endgraf \para_end:
```

(*End of definition for* `\par`, `\endgraf`, *and* `\@@par`. *These functions are documented on page 405.*)

While this is not integrated properly into the format we have to redo the `\everypar` setting from the kernel, otherwise that gets lost (as it happens before that file is loaded).

```
\everypar{\@nodocument} %% To get an error if text appears before the \document
```

3.2 The error messages

This one is used when we detect that some hook code has changed the mode where it shouldn't, e.g., by starting or ending a paragraph. The first argument is the hook name second the mode it should have stayed in but didn't.

```
\msg_new:nnnn { hooks } { para-mode }
  {
    Illegal~mode~ change~ in~ hook~ 'para/#1'.\\
    Hook~ code~ did~ not~ remain~ in~ #2~ mode.
  }
  {
    Paragraph~ hooks~ cannot~ change~ the~ TeX~ mode~ without~ causing~
    endless~ recursion.~ The~ hook~ code~ in~ 'para/#1'~ needs~ to~ stay~
    in~ #2~ mode,~ but~ it~ didn't.~ Examine~ the~ hook~
    code~ with~ \iow_char:N \\ShowHook~ to~ find~ the~ issue.
  }
```

And here is one used in the “raw” commands when they are used outside of vertical mode.

```
\msg_new:nnnn { latex2e } { raw-para }
  {
     Not~ in~ vertical~ mode.
  }
  {
    Starting~ a~ paragraph~ with~ \iow_char:N \\RawIndent~ or~
    \iow_char:N \\RawNoindent \\
    (or~ \iow_char:N \\para_raw_indent:~ or~
    \iow_char:N \\para_raw_noindent:)~ is~ only~ allowed \\
    if~ LaTeX~ is~ in~ vertical~ mode.
  }
```

```
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {ltpara}{Undo~hooks~for~paragraphs}
⟨latexrelease⟩
⟨latexrelease⟩\let \OmitIndent  \@undefined
⟨latexrelease⟩\let \IndentBox   \@undefined
⟨latexrelease⟩\let \RawIndent   \@undefined
⟨latexrelease⟩\let \RawNoindent \@undefined
⟨latexrelease⟩\let \RawParEnd   \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\cs_set_eq:NN \par     \tex_par:D
⟨latexrelease⟩\cs_set_eq:NN \@@par   \tex_par:D
⟨latexrelease⟩\cs_set_eq:NN \endgraf \tex_par:D
⟨latexrelease⟩
```

We also need to clean up the primitive “everypar” as that should no longer execute any code by default. And, of course, make `\everypar` become the primitive again.

```
⟨latexrelease⟩\tex_everypar:D {}
⟨latexrelease⟩\cs_set_eq:NN \everypar \tex_everypar:D
⟨latexrelease⟩
⟨latexrelease⟩\EndModuleRelease
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
```

File 17
ltmeta.dtx

Abstract

This code defines the `\DocumentMetadata` interface.

1 Introduction

In the past there was no dedicated location to declare settings concerning a document as a whole. Settings are placed somewhere in the preamble or with the class options or even with some package options. For some settings this can be too late, for example the pdf version can no longer be changed if a package has used code which already opened the PDF.

`\DocumentMetadata` as a new command unifies such settings in one place. It must be used before `\documentclass` but can be issued more than once there.

At the moment most of the code run by `\DocumentMetadata` is external to the format and subject to change. This includes the supported key/values.

For that reason all that happens right now in the format is to look for suitable support files and if found, to redirect the processing to them.

1.1 `\DocumentMetadata`

`\DocumentMetadata` `\DocumentMetadata{`⟨*key-value list*⟩`}`

The keys defined for `\DocumentMetadata` currently allow to set the PDF version, to set the PDF `/Lang`, to uncompress a PDF, to set the language and to declare a few PDF standards and some color profiles.

`\DocumentMetadata` is also used to activate the new PDF management code and it loads a number of required files for the PDF management code. As this forces the loading of the backend files, a backend which can't be detected automatically like `dvipdfmx`, must be set in the first `\DocumentMetadata` call (if there is more than one).

The full set of keys currently supported is documented in `documentmetadata-support.pdf` for now.

2 The Implementation

```
⟨*2ekernel | latexrelease⟩
```

Not needed yet but ...

```
%\ExplSyntaxOn
⟨latexrelease⟩\NewModuleRelease{2022/06/01}{ltmeta}
⟨latexrelease⟩                  {Document Metadata handling}
\let \IfDocumentMetadataTF \@secondoftwo
\protected\def\DocumentMetadata{%
  \InputIfFileExists{documentmetadata-support.ltx}%
     {}%
```

The above file is changing `\DocumentMetadata` to a suitable definition (or so we hope), so that we can try again — if not tough.

If the file can't be found we say so and carry on without it.

```
    {%
      \@latex@error{No support files for
                    \noexpand\DocumentMetadata found}
        {Is the 'LaTeX-lab' bundle installed?%
        \MessageBreak
        Without it, the declaration is ignored.}%
```

No point in trying this more than once if there are several calls in the document.

```
     \let\DocumentMetadata\@gobble
    }%
  \let \IfDocumentMetadataTF \@firstoftwo
  \DocumentMetadata
}
```

To allow package and class author to support for document links we provide also the new interface commands of the hyperref package for the creation of targets.

`\MakeLinkTarget`
`\LinkTargetOn`
`\LinkTargetOff`
`\NextLinkTarget`

```
⟨latexrelease⟩\IncludeInRelease{2024/11/01}%
⟨latexrelease⟩                 {\MakeLinkTarget}{Record target name for tagging support}%
\ExplSyntaxOn
\int_new:N\g__kernel_target_int
\NewDocumentCommand\MakeLinkTarget{sO{}m}{%
  \ifvmode
    \special{}%
  \else
    \@savsf\spacefactor
    \smash{}%
    \spacefactor\@savsf
  \fi
  \IfBooleanTF {#1}
   {
     \tl_gset:Ne \@currentHref {#3}
   }
   {
     \int_gincr:N\g__kernel_target_int
     \tl_gset:Ne \@currentHref {target*.\int_use:N\g__kernel_target_int}
   }
  \UseTaggingSocket{recordtarget}
   }
\ExplSyntaxOff
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                 {\MakeLinkTarget}{Record target name for tagging support}%
⟨latexrelease⟩\NewDocumentCommand\MakeLinkTarget{sO{}m}{%
⟨latexrelease⟩  \ifvmode
⟨latexrelease⟩    \special{}%
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@savsf\spacefactor
⟨latexrelease⟩    \smash{}%
⟨latexrelease⟩    \spacefactor\@savsf
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\EndIncludeInRelease
```

```
  \NewDocumentCommand\LinkTargetOn{}{}
\NewDocumentCommand\LinkTargetOff{}{}
\NewDocumentCommand\NextLinkTarget{m}{}
```

(*End of definition for* `\MakeLinkTarget` *and others.*)

We do not undo `\MakeLinkTarget` and friends if we roll back, in case they are used in packages that themselves do not offer rollback. This way a roll forward adds them, but the dummies remain if you roll back and you don't get missing csname errors if they are used.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{ltmeta}%
⟨latexrelease⟩                  {Undo Document Metadata handling}
⟨latexrelease⟩
⟨latexrelease⟩\let\DocumentMetadata\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndModuleRelease
```

Again for the future . . .

```
%\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
```

Restore module prefix (if any):

```
⟨@@=⟩
```

File 18
ltspace.dtx

1 Spacing

This section deals with spacing, and line- and page-breaking.

1.1 User Commands

`\nopagebreak` `[`⟨*i*⟩`]` : ⟨*i*⟩ = 0,...,4.

Default argument = 4. Puts a penalty into the vertical list output as follows:

0 : penalty = 0
1 : penalty = `\@lowpenalty`
2 : penalty = `\@medpenalty`
3 : penalty = `\@highpenalty`
4 : penalty = 10000

`\pagebreak` `[`⟨*i*⟩`]` : same as except negatives of its penalty

`\linebreak` `[`⟨*i*⟩`]` : analog of the above

`\nolinebreak` `[`⟨*i*⟩`]` : analog of the above

`\samepage` : inhibits page breaking most places by setting the following penalties to 10000:

```
\interlinepenalty
\predisplaypenalty
\postdisplaypenalty
\interdisplaylinepenalty
\@beginparpenalty
\@endparpenalty
\@itempenalty
\@secpenalty
\interfootnotelinepenalty
```

`\\` : initially defined to be `\newline`

`\\[`⟨*length*⟩`]` : initially defined to be `\vspace{`⟨*length*⟩`}\newline`

Note: `\\*` adds a `\vadjust{\penalty 10000}`

OBSOLETE COMMANDS (which never made it into the manual):

`\obeycr` : defines <CR> == `\\\relax`

`\restorecr` : restores <CR> to its usual meaning.

1.2 Chris' comments

There are several aspects of the handling of space in horizontal mode that are inconsistent or do not work well in some cases. These are largely concerned with ignoring the effect of space tokens that would otherwise typeset an inter-word space.

Negating the effect of such space tokens is achieved by two mechanisms:

- `\unskip` is used to remove the glue just added by a space that has already had its effect; it is sometimes invoked after an `\ifdim` test on `\lastskip` (see below);
- `\ignorespaces` is used to ignore space-tokens yet to come.

The test done on \lastskip is sometimes for equality with zero and sometimes for being positive. Recall also that the test is only on the natural length of the glue and that no glue cannot be distinguished from glue whose natural length is zero: to summarise, a pretty awful test. It is not clear why these tests are not all the same; I think that they should all be for equality. One place where \unskip is often used is just before a \par (which itself internally does an \unskip) and one bit of code (in \@item) even has two \unskips before a \par. These uses may be fossil code but if they are necessary, maybe \@killglue would be even safer.

Such removal of glue by \unskip may sometimes have the wrong result, removing not the glue from a space-token but other explicit glue; this is sometimes not what is intended.

A common way to prevent such removal is to add an \hskip\z@ after the glue that should not be removed. This protects that glue against one \unskip with no test but not against more than one. It does work for 'tested \unskips'. This is used by \hspace* but not by \hspace; this is inconsistent as the star is supposed to prevent removal only at the beginning of a line, not at the end, or in a tabular, etc.

If this reason for removing glue were the only consideration then a tested-\unskip and protection by \hskip\z@ would suffice but would need to be consistently implemented.

However, the class of invisibles, commands and environments tries to be even cleverer: one of these tries to leave only one inter-word space whenever there is one before it and one after it; and it does this quite well.

But problems can arise when there is not a space-token on both sides of it; in particular, when an invisible appears at the beginning or end of a piece of text the method still leaves one space token whereas usually in these cases it should leave none.

Also, the current rules do not work well when more than one such command appears consecutively, separated by space-tokens; it leaves glue between every other invisible.

There is also a question about what these commands should do when they occur next to spaces that do not come from space tokens but, for example, from \hspace. Should they still produce 'just one space'? If so, which one? It is good to note that the manual is sufficiently cautious about invisibles that we are not obliged to make anything work.

Another interesting side-road to explore is whether the space-tokens either side of an \hspace{...} should be ignored.

One alternative to the current algorithm that is often suggested is that all glue around the invisible should be consolidated into a space after it (usually without stating how much glue should be put there). The command \nolinebreak is implemented this way (and \linebreak should also be). This does not work correctly for the following common case:

```
... some text
\index{some-word}
some-word and more text.
```

This is optimal coding since it is normal to index a word that gets split across a page-break on its starting page. This would, on the other hand, fix another common (and documented) failure of the current system: when the invisible is the last thing in a paragraph the space before it is not removed and, worse, it is also hidden from the paragraph-ending mechanism so that an 'empty' line can be created at the end of the paragraph.

Another deficiency (I think) of the current system is that the following is treated as having the `\index` command between the paragraphs, which is probably not what the author intended (since there is no empty line after it).

```
\index{beginnings}
Beginnings of paragraphs ...
```

I know of no algorithm that will handle satisfactorily even all the most common cases; note that it could be that the best algorithm may be different for different invisibles since, for example, the common uses and expected behaviour of `\index`, `\marginpar`, `\linebreak`, `\pagebreak` and `\vspace` are somewhat different. [For example, is `\vspace` ever used in the middle of a paragraph?]

One method that can (and is) used to make invisible commands produce no space when used at the beginning of text is to put in some glue that is nearly enough the same as no glue or glue of zero length in all respects except for the precise test for not being exactly equal to zero; examples of such glue are `\hskip 1sp` and, possibly better but more complex, `\hskip -1sp \hskip 1sp`. However, this only works when it is known that user-supplied text is about to start.

Some similar concerns apply to the handling of space and penalties in vertical mode; there is an extra hurdle here as `\unskip` does not work on the main vertical list. The complexity of the tests done by `\addvspace` have never been explained.

The implementation of space hacks etc for vertical mode is another major area that needs further attention; my earlier experiments did not produce much improvement over the current unsatisfactory situation.

One particular problem is what happens when the following very natural coding is used (part of the problem here is that this looks like an hmode problem, but it is not):

```
... end of text.

\begin{enumerate}
  \item \label{item:xxx} Item text.
\end{enumerate}
```

1.3 Some immediate actions

- Fix bug in `\linebreak`.
- Fix bug in `\\*`.
- Reimplement `\\`, etc, removing extra `\vadjust`s and getting better error trapping (this seems to involve a lot more tokens).
- Investigate whether `\\`, etc need to be errors in vmode; I think that they could be noops (maybe with a warning).
- Make all(?) `\unskips` include test for zero skip (rather than other tests or no test).
- Consider replacing `\hskip 1sp` by something better (here called an 'infinitesimal' skip).
- Look at all `\hskip\z@` (or similar) to see if they should be changed to an 'infinitesimal' skip.

- Resolve the inconsistency between `\hspace` and `\hspace*`.
- Remove unnecessary `\unskips`.
- Investigate and rationalise the 'newline' code.
- Find better algorithms for all sorts of things or, easier(?), fix TEX itself.

1.4 The code

```
⟨*2ekernel⟩
\message{spacing,}
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\pagebreak}{Make commands robust}%
```

`\pagebreak`
`\nopagebreak`

```
\DeclareRobustCommand\pagebreak{\@testopt{\@no@pgbk-}4}
\DeclareRobustCommand\nopagebreak{\@testopt\@no@pgbk4}
```

(*End of definition for* `\pagebreak` *and* `\nopagebreak`.)

`\linebreak`
`\nolinebreak`

```
\DeclareRobustCommand\linebreak{\@testopt{\@no@lnbk-}4}
\DeclareRobustCommand\nolinebreak{\@testopt\@no@lnbk4}
```

(*End of definition for* `\linebreak` *and* `\nolinebreak`.)

`\samepage`

```
\DeclareRobustCommand\samepage{\interlinepenalty\@M
   \predisplaypenalty\@M
   \postdisplaypenalty\@M
   \interdisplaylinepenalty\@M
   \@beginparpenalty\@M
   \@endparpenalty\@M
   \@itempenalty\@M
   \@secpenalty\@M
   \interfootnotelinepenalty\@M}
```

(*End of definition for* `\samepage`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\pagebreak}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\pagebreak
⟨latexrelease⟩\kernel@make@fragile\nopagebreak
⟨latexrelease⟩\kernel@make@fragile\linebreak
⟨latexrelease⟩\kernel@make@fragile\nolinebreak
⟨latexrelease⟩\kernel@make@fragile\samepage
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@no@pgbk

```
\def\@no@pgbk #1[#2]{%
  \ifvmode
    \penalty #1\@getpen{#2}%
  \else
    \@bsphack
    \vadjust{\penalty #1\@getpen{#2}}%
    \@esphack
  \fi}
```

(*End of definition for* `\@no@pgbk`.)

\@no@lnbk

```
\def\@no@lnbk #1[#2]{%
  \ifvmode
    \@nolnerr
  \else
    \@tempskipa\lastskip
    \unskip
    \penalty #1\@getpen{#2}%
    \ifdim\@tempskipa>\z@
      \hskip\@tempskipa
      \ignorespaces
    \fi
  \fi}
```

(*End of definition for* `\@no@lnbk`.)

\\ The purpose of the new code is to fix a few bugs; however, it also attempts to optimize the following, in order of priority:

1. efficient execution of plain `\\`;
2. efficient execution of `\\[...]`;
3. memory use;
4. name-space use.

The changes should make no difference to the typeset output. It appears to be safe to use `\reserved@e` and `\reserved@f` here (other reserved macros are somewhat disastrous).

These changes made `\newline` even less robust than it had been, so now it is explicitly robust, like `\\`.

The internal definition of the 'normal' definition of `\\`.

\@normalcr

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {\@normalcr}{Make robust}%
\protected\def\@normalcr{%
  \let \reserved@e \relax
  \let \reserved@f \relax
  \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
             \@xnewline}%
  \@xnewline}
```

```
\let\\\@normalcr
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@normalcr}{Make robust}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\\{%
⟨latexrelease⟩   \let \reserved@e \relax
⟨latexrelease⟩   \let \reserved@f \relax
⟨latexrelease⟩   \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
⟨latexrelease⟩             \@xnewline}%
⟨latexrelease⟩  \@xnewline}
⟨latexrelease⟩\expandafter\let\expandafter\@normalcr
⟨latexrelease⟩     \csname\expandafter\@gobble\string\\ \endcsname
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\\` *and* `\@normalcr`.)

`\@vspace@calcify` Helper command to produce a `\vskip` that is first run through `\setlength`. This way the `calc` package can operate on the argument value.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@vspace@calcify}{Add calc support}%
\def\@vspace@calcify#1{\setlength\sp@ce@skip{#1}\vskip\sp@ce@skip}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@vspace@calcify}{Add calc support}%
⟨latexrelease⟩
⟨latexrelease⟩\let\@vspace@calcify\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@vspace@calcify`.)

`\newline` A simple form of the 'normal' definition of `\\`.

```
\DeclareRobustCommand\newline{\@normalcr\relax}
```

(*End of definition for* `\newline`.)

`\@xnewline`

```
\def\@xnewline{\@ifnextchar[% ] bracket matching
                   \@newline
                  {\@gnewline\relax}}
```

(*End of definition for* `\@xnewline`.)

`\@newline`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
```

```
⟨latexrelease⟩                  {\@newline}{\newline calc support}%
\def\@newline[#1]{\let \reserved@e \vadjust
                    \@gnewline {\@vspace@calcify{#1}}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@newline}{\newline calc support}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@newline[#1]{\let \reserved@e \vadjust
⟨latexrelease⟩                    \@gnewline {\vskip #1}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@newline`.)

`\@gnewline` The `\nobreak` added to prevent null lines when `\\` ends an overfull line. Change made 24 May 89 as suggested by Frank Mittelbach and Rainer Schöpf

```
\def\@gnewline #1{%
  \ifvmode
    \@nolnerr
  \else
    \unskip \reserved@e {\reserved@f#1}\nobreak \hfil \break
  \fi}
```

(*End of definition for* `\@gnewline`.)

`\@getpen`

```
\def\@getpen#1{\ifcase #1 \z@ \or \@lowpenalty\or
         \@medpenalty \or \@highpenalty
         \else \@M \fi}
```

(*End of definition for* `\@getpen`.)

`\if@nobreak` Switch used to avoid page breaks caused by `\label` after a section heading, etc. It should be **GLOBALLY** set true after the `\nobreak` and **globally** set false by the next invocation of `\everypar`.

Commands that reset `\everypar` should globally set it false if appropriate.

```
\def\@nobreakfalse{\global\let\if@nobreak\iffalse}
\def\@nobreaktrue {\global\let\if@nobreak\iftrue}
\@nobreakfalse
```

(*End of definition for* `\if@nobreak`.)

`\@savsk` `\@savsf` Registers used to save the space factor and last skip.

```
\newdimen\@savsk
\newcount\@savsf
```

(*End of definition for* `\@savsk` *and* `\@savsf`.)

\@bsphack

\@bsphack and \@esphack used by macros such as \index and \begin{@float} ... \end{@float} that want to be invisible — i.e., not leave any extra space when used in the middle of text. Such a macro should begin with \@bsphack and end with \@esphack. The macro in question should not create any text, nor change the mode.

Before giving the current definition we give an extended definition that is currently not used (because it doesn't work as advertised:-)

These are generalised hacks which attempt to do sensible things when 'invisible commands' appear in vmode too.

They need to cope with space in both hmode (plus spacefactor) and vmode, and also cope with breaks etc. In vmode this means ensuring that any following \addvspace, etc sees the correct glue in \lastskip.

In fact, these improved versions should be used for other cases of 'whatsits, thingies etc' which should be invisible. They are only for commands, not environments (see notes on \@Esphack).

BTW, anyone know why the standard hacks are surrounded by \ifmmode\else rather than simply \ifhmode?

And are there any cases where saving the spacefactor is essential? I have some extensions where it is, but it does not appear to be so in the standard uses.

```
def \@bsphack{%
 \relax \ifvmode
   \@savsk \lastskip
   \ifdim \lastskip=\z@
   \else
     \vskip -\lastskip
   \fi
 \else
   \ifhmode
     \@savsk \lastskip
     \@savsf \spacefactor
   \fi
 \fi
```

I think that, in vmode, it is the safest to put in a \nobreak immediately after such things since writes, inserts etc followed by glue give valid breakpoints and, in general, it is possible to create breaks but impossible to destroy them.

```
def \@esphack{%
  \relax \ifvmode
    \nobreak
    \ifdim \@savsk=\z@
    \else
      \vskip\@savsk
    \fi
  \else
    \ifhmode
      \spacefactor \@savsf
      \ifdim \@savsk>\z@
        \ignorespaces
      \fi
```

```
    \fi
  \fi
```

For the moment we are going to ignore the vertical versions until they are correct.

```
\def\@bsphack{%
  \relax
  \ifhmode
    \@savsk\lastskip
    \@savsf\spacefactor
  \fi}
```

(*End of definition for* `\@bsphack`.)

`\@esphack` Companion to `\@bsphack`. If this command is not properly paired with `\@bsphack` one might end up with a low-level TEX error: "BAD spacefactor". One possible cause is calling `\@bsphack` in vertical mode, then doing something that gets you (sometimes) into horizontal mode and finally calling `\@esphack`. Even if no error is generated that is wrong, because `\@esphack` will then use the saved values for `\@savsk` and `\@savsf` from some earlier invocation of `\@bsphack` which will have nothing to do with the current situation.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2018/10/10}%
⟨latexrelease⟩                  {\@esphack}{hyphenation and nobreak after space hack}%
⟨*2ekernel | latexrelease⟩
\def\@esphack{%
  \relax
  \ifhmode
    \spacefactor\@savsf
    \ifdim\@savsk>\z@

      \ifdim\lastskip=\z@
        \nobreak \hskip\z@skip
      \fi
      \ignorespaces
    \fi

  \else
    \ifvmode
      \if@nobreak\nobreak\else\if@noskipsec\nobreak\fi\fi
    \fi

  \fi}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/10/01}%
⟨latexrelease⟩                  {\@esphack}{hyphenation and nobreak after space hack}%
⟨latexrelease⟩\def\@esphack{%
⟨latexrelease⟩  \relax
⟨latexrelease⟩  \ifhmode
⟨latexrelease⟩    \spacefactor\@savsf
⟨latexrelease⟩    \ifdim\@savsk>\z@
⟨latexrelease⟩      \ifdim\lastskip=\z@
⟨latexrelease⟩        \nobreak \hskip\z@skip
⟨latexrelease⟩      \fi
```

```
⟨latexrelease⟩      \ignorespaces
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\@esphack}{hyphenation and nobreak after space hack}%
⟨latexrelease⟩\def\@esphack{%
⟨latexrelease⟩  \relax
⟨latexrelease⟩  \ifhmode
⟨latexrelease⟩    \spacefactor\@savsf
⟨latexrelease⟩    \ifdim\@savsk>\z@
⟨latexrelease⟩      \nobreak \hskip\z@skip
⟨latexrelease⟩      \ignorespaces
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@esphack}{hyphenation and nobreak after space hack}%
⟨latexrelease⟩\def\@esphack{%
⟨latexrelease⟩  \relax
⟨latexrelease⟩  \ifhmode
⟨latexrelease⟩    \spacefactor\@savsf
⟨latexrelease⟩    \ifdim\@savsk>\z@
⟨latexrelease⟩      \ignorespaces
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@esphack`.)

`\@Esphack` A variant of `\@esphack` that sets the `@ignore` switch to true (as `\@esphack` used to do previously). This is currently used only for floats and similar environments.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\@Esphack}{hyphenation after space hack}%
⟨*2ekernel | latexrelease⟩
\def\@Esphack{%
  \relax
  \ifhmode
    \spacefactor\@savsf
    \ifdim\@savsk>\z@
      \nobreak \hskip\z@skip
      \@ignoretrue
      \ignorespaces
    \fi
   \fi}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@Esphack}{hyphenation after space hack}%
⟨latexrelease⟩\def\@Esphack{%
⟨latexrelease⟩  \relax
⟨latexrelease⟩  \ifhmode
```

```
⟨latexrelease⟩     \spacefactor\@savsf
⟨latexrelease⟩     \ifdim\@savsk>\z@
⟨latexrelease⟩       \@ignoretrue
⟨latexrelease⟩       \ignorespaces
⟨latexrelease⟩     \fi
⟨latexrelease⟩    \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@Esphack`.)

`\@vbsphack` Another variant which is useful for invisible things which should not live in vmode (this is how some people feel about marginals).

If it occurs in vmode then it enters hmode and ensures that `\@savsk` is nonzero so that the `\ignorespaces` is put in later. It is not used at present.

```
\def \@vbsphack{ %
   \relax \ifvmode
     \leavevmode
     \@savsk 1sp
     \@savsf \spacefactor
   \else
     \ifhmode
       \@savsk \lastskip
       \@savsf \spacefactor
     \fi
   \fi
}
```

(*End of definition for* `\@vbsphack`.)

1.5 Vertical spacing

LaTeX supports the plain TeX commands `\smallskip`, `\medskip` and `\bigskip`. However, it redefines them using `\vspace` instead of `\vskip`.

Extra vertical space is added by the command `\addvspace{`⟨*skip*⟩`}`, which adds a vertical skip of ⟨*skip*⟩ to the document. The sequence
`\addvspace{`⟨*s1*⟩`}` `\addvspace{`⟨*s2*⟩`}` is equivalent to
`\addvspace{`⟨*maximum of s1, s2*⟩`}`.

`\addvspace` should be used only in vertical mode, and gives an error if it's not. The `\addvspace` command does *not* add vertical space if `@minipage` is true. The minipage environment uses this to inhibit the addition of extra vertical space at the beginning.

Penalties are put into the vertical list with the `\addpenalty{`⟨*penalty*⟩`}` command. It works properly when `\addpenalty` and `\addvspace` commands are mixed.

The `@nobreak` switch is set true used when in vertical mode and no page break should occur. (Right now, it is used only by the section heading commands to inhibit page breaking after a heading.)

`\@xaddvskip` Internal macro for `\vspace` handling the case that space has previously been added.

```
\def\@xaddvskip{%
  \ifdim\lastskip<\@tempskipb
    \vskip-\lastskip
```

```
    \vskip\@tempskipb
  \else
    \ifdim\@tempskipb<\z@
      \ifdim\lastskip<\z@
      \else
        \advance\@tempskipb\lastskip
        \vskip-\lastskip
        \vskip \@tempskipb
      \fi
    \fi
  \fi}
```

(*End of definition for* `\@xaddvskip`.)

`\addvspace` Add vertical space taking into account space already added, as described above.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/11/01}%
⟨latexrelease⟩                  {\addvspace}{drop unnecessary no-item error}%
\protected\def\addvspace#1{%
```

When this is encountered in hmode, we check whether we are in an hbox and if so generate a LaTeX error, as otherwise this would cause a bunch of low-level errors. In unrestricted hmode we simply switch to vmode by issuing a `\par`.

```
  \ifhmode \ifinner \@LRmoderr \else \par \fi \fi
  \if@minipage\else
    \ifdim \lastskip =\z@
      \@vspace@calcify{#1}%
    \else
    \setlength\@tempskipb{#1}%
      \@xaddvskip
    \fi
  \fi
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\addvspace}{\addvspace calc support}%
⟨latexrelease⟩\def\addvspace#1{%
⟨latexrelease⟩  \ifvmode
⟨latexrelease⟩     \if@minipage\else
⟨latexrelease⟩       \ifdim \lastskip =\z@
⟨latexrelease⟩         \@vspace@calcify{#1}%
⟨latexrelease⟩       \else
⟨latexrelease⟩       \setlength\@tempskipb{#1}%
⟨latexrelease⟩         \@xaddvskip
⟨latexrelease⟩       \fi
⟨latexrelease⟩     \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@noitemerr
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\addvspace}{\addvspace calc support}%
⟨latexrelease⟩
⟨latexrelease⟩\def\addvspace#1{%
⟨latexrelease⟩  \ifvmode
⟨latexrelease⟩     \if@minipage\else
⟨latexrelease⟩       \ifdim \lastskip =\z@
⟨latexrelease⟩         \vskip #1\relax
⟨latexrelease⟩       \else
⟨latexrelease⟩       \@tempskipb#1\relax
⟨latexrelease⟩         \@xaddvskip
⟨latexrelease⟩       \fi
⟨latexrelease⟩     \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@noitemerr
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\addvspace`.)

`\addpenalty`

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2024/11/01}%
⟨latexrelease⟩                 {\addpenalty}{\addpenalty drop error}%
⟨*2ekernel | latexrelease⟩
```

```
\protected\def\addpenalty#1{%
```

See description of `\addvspace` for documentation of the next line of code.

```
  \ifhmode \ifinner \@LRmoderr \else \par \fi \fi
```

Fix provided by Donald (though the original fix was not good enough). In 2005 Plamen Tanovski discovered that this fix wasn't good enough either as the `\vskip` kept getting bigger if several `\addpenalty` commands followed each other. Donald kindly send a new fix.

```
  \if@minipage
  \else
    \if@nobreak
    \else
      \ifdim\lastskip=\z@
        \penalty#1\relax
      \else
        \@tempskipb\lastskip
```

We have to make sure the final `\vskip` seen by TEX is the correct one, namely `\@tempskipb`. However, we may have to adjust for `\prevdepth` when placing the penalty; that should not affect the skip we pass on to TEX.

```
        \begingroup
          \@tempskipa\@tempskipb
          \advance \@tempskipb
            \ifdim\prevdepth>\maxdepth\maxdepth\else
```

If \prevdepth is -1000pt due to \nointerlineskip we better not add it!

```
                \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
              \fi
            \vskip -\@tempskipb
            \penalty#1%
            \ifdim\@tempskipa=\@tempskipb
```

Do nothing if the \prevdepth check made no adjustment.

```
            \else
```

Combine the prevdepth adjustment into a single skip.

```
              \advance\@tempskipb -\@tempskipa
              \vskip \@tempskipb
            \fi
```

The final skip is always the specified length.

```
            \vskip \@tempskipa
          \endgroup
        \fi
      \fi
    \fi
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\addpenalty}{\addpenalty}%
⟨latexrelease⟩\def\addpenalty#1{%
⟨latexrelease⟩  \ifvmode
⟨latexrelease⟩    \if@minipage
⟨latexrelease⟩    \else
⟨latexrelease⟩      \if@nobreak
⟨latexrelease⟩      \else
⟨latexrelease⟩        \ifdim\lastskip=\z@
⟨latexrelease⟩          \penalty#1\relax
⟨latexrelease⟩        \else
⟨latexrelease⟩          \@tempskipb\lastskip
⟨latexrelease⟩          \begingroup
⟨latexrelease⟩            \@tempskipa\@tempskipb
⟨latexrelease⟩            \advance \@tempskipb
⟨latexrelease⟩              \ifdim\prevdepth>\maxdepth\maxdepth\else
⟨latexrelease⟩                 \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
⟨latexrelease⟩              \fi
⟨latexrelease⟩             \vskip -\@tempskipb
⟨latexrelease⟩             \penalty#1%
⟨latexrelease⟩             \ifdim\@tempskipa=\@tempskipb
⟨latexrelease⟩             \else
⟨latexrelease⟩               \advance\@tempskipb -\@tempskipa
⟨latexrelease⟩               \vskip \@tempskipb
⟨latexrelease⟩             \fi
⟨latexrelease⟩             \vskip \@tempskipa
⟨latexrelease⟩          \endgroup
⟨latexrelease⟩        \fi
⟨latexrelease⟩      \fi
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
```

```
⟨latexrelease⟩     \@noitemerr
⟨latexrelease⟩   \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\addpenalty}{\addpenalty}%
⟨latexrelease⟩\def\addpenalty#1{%
⟨latexrelease⟩   \ifvmode
⟨latexrelease⟩     \if@minipage
⟨latexrelease⟩     \else
⟨latexrelease⟩       \if@nobreak
⟨latexrelease⟩       \else
⟨latexrelease⟩         \ifdim\lastskip=\z@
⟨latexrelease⟩           \penalty#1\relax
⟨latexrelease⟩         \else
⟨latexrelease⟩           \@tempskipb\lastskip
⟨latexrelease⟩           \vskip -\lastskip
⟨latexrelease⟩           \penalty#1%
⟨latexrelease⟩           \vskip\@tempskipb
⟨latexrelease⟩         \fi
⟨latexrelease⟩       \fi
⟨latexrelease⟩     \fi
⟨latexrelease⟩   \else
⟨latexrelease⟩     \@noitemerr
⟨latexrelease⟩   \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\addpenalty`.)

`\vspace`
`\@vspace`
`\@vspacer`

The new code for these commands depends on the following facts:

- The value of prevdepth is changed only when a box or rule is created and added to a vertical list;
- The value of prevdepth is used only when a box is created and added to a vertical list;
- The value of prevdepth is always local to the building of one vertical list.

```
\DeclareRobustCommand\vspace{\@ifstar\@vspacer\@vspace}
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@vspace}{Support calc in \vspace}%
```

We support calc syntax in the argument and therefore use `\setlength`.

```
\def\@vspace #1{%
  \ifvmode
    \@vspace@calcify{#1}%
    \vskip\z@skip
   \else
     \@bsphack
     \vadjust{\@restorepar
              \@vspace@calcify{#1}%
              \vskip\z@skip
```

```
              }%
     \@esphack
   \fi}
\def\@vspacer#1{%
  \ifvmode
    \dimen@\prevdepth
    \hrule \@height\z@
    \nobreak
    \@vspace@calcify{#1}%
    \vskip\z@skip
    \prevdepth\dimen@
  \else
    \@bsphack
    \vadjust{\@restorepar
             \hrule \@height\z@
             \nobreak
             \@vspace@calcify{#1}%
             \vskip\z@skip}%
    \@esphack
  \fi}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@vspace}{Support calc in \vspace}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@vspace #1{%
⟨latexrelease⟩  \ifvmode
⟨latexrelease⟩    \vskip #1
⟨latexrelease⟩    \vskip\z@skip
⟨latexrelease⟩   \else
⟨latexrelease⟩     \@bsphack
⟨latexrelease⟩     \vadjust{\@restorepar
⟨latexrelease⟩              \vskip #1
⟨latexrelease⟩              \vskip\z@skip
⟨latexrelease⟩              }%
⟨latexrelease⟩     \@esphack
⟨latexrelease⟩   \fi}
⟨latexrelease⟩\def\@vspacer#1{%
⟨latexrelease⟩  \ifvmode
⟨latexrelease⟩    \dimen@\prevdepth
⟨latexrelease⟩    \hrule \@height\z@
⟨latexrelease⟩    \nobreak
⟨latexrelease⟩    \vskip #1
⟨latexrelease⟩    \vskip\z@skip
⟨latexrelease⟩    \prevdepth\dimen@
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@bsphack
⟨latexrelease⟩    \vadjust{\@restorepar
⟨latexrelease⟩             \hrule \@height\z@
⟨latexrelease⟩             \nobreak
⟨latexrelease⟩             \vskip #1
⟨latexrelease⟩             \vskip\z@skip}%
⟨latexrelease⟩    \@esphack
⟨latexrelease⟩  \fi}
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \vspace*,* \@vspace*, and* \@vspacer*.*)

\smallskip
\medskip
\bigskip

```
\def\smallskip{\vspace\smallskipamount}
\def\medskip{\vspace\medskipamount}
\def\bigskip{\vspace\bigskipamount}
```

(*End of definition for* \smallskip*,* \medskip*, and* \bigskip*.*)

\smallskipamount
\medskipamount
\bigskipamount

```
\newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
\newskip\medskipamount   \medskipamount  =6pt plus 2pt minus 2pt
\newskip\bigskipamount   \bigskipamount =12pt plus 4pt minus 4pt
```

(*End of definition for* \smallskipamount*,* \medskipamount*, and* \bigskipamount*.*)

1.6 Horizontal space (and breaks)

\nobreakdashes

This idea is borrowed from the amsmath package but here we define a robust command.

This command is a low-level command designed for use only before hyphens or dashes (such as `-`, `--`, or `---`).

It could probably be better implemented: it may need its own private token register and temporary command.

Setting the hyphen in a box and then unboxing it means that the normal penalty will not be added after it—and if the penalty is not there a break will not be taken (unless an explicit penalty or glue follows, thus the final `\nobreak`).

Note that even if it is not followed by a '-', it still leaves vmode and sets the space-factor; so use it carefully!

```
\DeclareRobustCommand{\nobreakdashes}{%
  \leavevmode
  \toks@{}%
  \def\reserved@a##1{\toks@\expandafter{\the\toks@-}%
                     \futurelet\@let@token \reserved@b}%
  \def\reserved@b   {\ifx\@let@token -%
                         \expandafter\reserved@a
                     \else
                       \setbox\z@ \hbox{\the\toks@\nobreak}%
                       \unhbox\z@
                       \spacefactor\sfcode`\-
                     \fi}%
  \futurelet\@let@token \reserved@b
}
```

(*End of definition for* \nobreakdashes*.*)

\nobreakspace
\@xobeysp

This is a robust command that produces a horizontal space at which, in paragraph-mode, a line-break is not possible. We then define an active `~` to expand to it since this is the documented behaviour of `~`. One reason for introducing this is that some 8-bit input encodings have a slot for such a space and we do not want to use active characters as the LaTeX internal commands.

The braces in the definition of ~ are needed to ensure that a following space is preserved when reading to/from internal files.

We need to keep \@xobeysp as it is widely used; so here it is let to the non-robust command \nobreakspace .

The fragile version of ~ needs a brace group after \nobreakspace to prevent loss of spaces if it occurs in an expansion context. That's not an issue with the updated \protected definition, so we keep the code shorter and avoid that.

```
\DeclareRobustCommand{\nobreakspace}{%
   \leavevmode\nobreak\ }
\catcode `\~=13
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                  {\tilde}{Protected tilde}%
⟨*2ekernel | latexrelease⟩
\protected\edef~{%
  \noexpand\ifincsname\noexpand\expandafter\string~%
  \noexpand\else
    \noexpand\expandafter\noexpand\nobreakspace
  \noexpand\fi
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\tilde}{Protected tilde}%
⟨latexrelease⟩\def~{\nobreakspace{}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
\expandafter\let\expandafter\@xobeysp\csname nobreakspace \endcsname
```

(*End of definition for* \nobreakspace *and* \@xobeysp.)

\@xobeytab Equivalent to the space case with the default settings.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                  {\@xobeytab}{Obeyed tabs}%
⟨*2ekernel | latexrelease⟩
\let\@xobeytab\@xobeysp
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@xobeytab}{Obeyed tabs}%
⟨latexrelease⟩\let\@xobeytab\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \@xobeytab.)

\@ Placed before a ", makes it a sentence-ending period. Does the right thing for other punctuation marks as well. Does this by setting spacefactor to 1000.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\@}{Space after \@}%
⟨*2ekernel | latexrelease⟩
```

```
\def\@{\spacefactor\@m{}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@}{Space after \@}%
⟨latexrelease⟩\def\@{\spacefactor\@m}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@`.)

`\hspace`

```
\DeclareRobustCommand\hspace{\@ifstar\@hspacer\@hspace}
```

(*End of definition for* `\hspace`.)

`\@hspace`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@hspace}{Support calc with \hspace}%
```

We use a private register to calculate the space (if calc is used). Previously we used a group but that results in `\everypar` etc. being executed inside the group if the `\hspace` starts a paragraph. This is a bug fix so we do not provide rollback to the incorrect intermediate version.

```
\newskip\sp@ce@skip
\def\@hspace#1{\setlength\sp@ce@skip{#1}\hskip\sp@ce@skip}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@hspace}{Support calc with \hspace}%

⟨latexrelease⟩
⟨latexrelease⟩\def\@hspace#1{\hskip #1\relax}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@hspace`.)

`\@hspacer` Extra `\hskip 0pt` added 1985/17/12 to guard against a following `\unskip` `\relax` added 13 Oct 88 for usual TEX lossage replaced both changes by `\hskip\z@skip` 27 Nov 91

```
\def\@hspacer#1{\vrule \@width\z@\nobreak
                \@hspace{#1}\hskip \z@skip}
```

(*End of definition for* `\@hspacer`.)

`\fill`

```
\newskip\fill
\fill = 0pt plus 1fill
```

(*End of definition for* `\fill`.)

`\stretch`

```
\def\stretch#1{\z@ \@plus #1fill\relax}
```

(*End of definition for* `\stretch`.)

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                  {\thinspace}{Start LR-mode}%
```

`\enspace`

```
\DeclareRobustCommand\enspace{\leavevmode@ifvmode\kern.5em }
```

(*End of definition for* `\enspace`.)

`\leavevmode@ifvmode` Leave vmode but only if we are really in vmode, otherwise the expansion is empty (which is not the case with the default definition).

```
\protected\def\leavevmode@ifvmode{\ifvmode\expandafter\indent\fi}
```

(*End of definition for* `\leavevmode@ifvmode`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\thinspace}{Start LR-mode}%
⟨latexrelease⟩\def\thinspace{\kern .16667em }
⟨latexrelease⟩\def\negthinspace{\kern-.16667em }
⟨latexrelease⟩\def\enspace{\kern.5em }
⟨latexrelease⟩\let\leavevmode@ifvmode\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\enskip`
`\quad`
`\qquad`

```
\def\enskip{\hskip.5em\relax}
\def\quad{\hskip1em\relax}
\def\qquad{\hskip2em\relax}
```

(*End of definition for* `\enskip`, `\quad`, *and* `\qquad`.)

For Unicode engines, make the Unicode soft hyphen an active character defined as `\-`.

```
\ifx\Umathcode\@undefined\else
  \catcode "AD=13
  \def^^ad{\-}
\fi
```

`\obeycr` The following definitions will probably get deleted or moved to compatibility mode soon.
`\restorecr`

```
{\catcode`\^^M=13 \gdef\obeycr{\catcode`\^^M13 \def^^M{\\\relax}%
    \@gobblecr}%
{\catcode`\^^M=13 \gdef\@gobblecr{\@ifnextchar
\@gobble\ignorespaces}}
\gdef\restorecr{\catcode`\^^M5 }}
```

(*End of definition for* `\obeycr` *and* `\restorecr`.)

```
⟨/2ekernel⟩
```

File 19
ltlogos.dtx

1 Logos

Various logos are defined here.

`\TeX` The TEX logo, adjusted so that a full stop after the logo counts as ending a sentence.

```
⟨*2ekernel⟩
\DeclareRobustCommand\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

(*End of definition for* `\TeX`.)

`\LaTeX` The LATEX logo.

```
\DeclareRobustCommand{\LaTeX}{L\kern-.36em%
        {\sbox\z@ T%
         \vbox to\ht\z@{\hbox{\check@mathfonts
                              \fontsize\sf@size\z@
                              \math@fontsfalse\selectfont
                              A}%
                         \vss}%
        }%
        \kern-.15em%
        \TeX}
```

(*End of definition for* `\LaTeX`.)

`\LaTeXe` The LATEX 2_ε logo as proposed by A-W designers.

```
\DeclareRobustCommand{\LaTeXe}{\mbox{\m@th
  \if b\expandafter\@car\f@series\@nil\boldmath\fi
  \LaTeX\kern.15em2$_{\textstyle\varepsilon}$}}
⟨/2ekernel⟩
```

(*End of definition for* `\LaTeXe`.)

File 20
ltfiles.dtx

1 File Handling

The following user commands are defined in this part:

`\document` (ie `\begin{document}`)
Reads in the .AUX files and `\catcode`'s `@` to 12.

`\nofiles`
Suppresses all file output by setting `\@filesw` false.

`\includeonly` `{`⟨*NAME1, ... ,NAMEn*⟩`}`
Causes only parts NAME1, ... ,NAMEn to be read by their `\include` commands. Works by setting partsw true and setting `\@partlist` to NAME1, ... ,NAMEn.

`\include` `{`⟨*NAME*⟩`}`
Does an `\input` NAME unless `\@partsw` is true and NAME is not in `\@partlist`. If `\@filesw` is true, then it directs .AUX output to NAME.AUX, including a checkpoint at the end.

`\input` `{`⟨*NAME*⟩`}`
The same as TeX's `\input`, except it allows optional braces around the file name. In LaTeX 2_ε, it also avoids the primitive 'missing file' error, if the file can not be found.

`\IfFileExists` `{`⟨*NAME*⟩`}{`⟨*then*⟩`}{`⟨*else*⟩`}`
If the file exists on the system, execute *then* otherwise execute *else*.

`\InputIfFileExists` `{`⟨*NAME*⟩`}{`⟨*then*⟩`}{`⟨*else*⟩`}`
If the file exists on the system, execute *then* and input *NAME* otherwise execute *else*.
Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
⟨*2ekernel⟩
\message{files,}
```

VARIABLES, SWITCHES AND INTERNAL COMMANDS:

```
  \@mainaux     : Output file number for main .AUX file.
  \@partaux     : Output file number for current part's .AUX file.
  \@auxout      : Either \@mainout or \@partout, depending on
                  which .AUX file output goes to.
  \@input{foo} : If file foo exists, then \input's it,
                  otherwise types a warning message.
  @filesw       : Switch – set false if no .AUX, .TOC, .IDX etc
                    files are to be written
  @partsw       : Set true by a \includeonly command.
  \@partlist    : Set to the argument of the \includeonly command.

  \cp@FOO       : The checkpoint for \include'd file FOO.TEX, written
                  by \@writeckpt at the end of file FOO.AUX


\includeonly{FILELIST} ==
 BEGIN
```

```
  \@partsw   := T
  \@partlist := FILELIST
 END

\include{FILE} ==
 BEGIN
  \clearpage
  if \@filesw = T
    then  \immediate\write\@mainaux{\string\@input{FILE.AUX}}
  fi
  if  \@partsw = T
    then \@tempswa := F
         \reserved@b == FILE
         for \reserved@a := \@partlist
             do if eval(\reserved@a) = eval(\reserved@b)
                   then \@tempswa := T          fi
             od
  fi

  if \@tempswa = T
     then \@auxout := \@partaux
          if \@filesw = T
            then  \immediate\openout\@partaux{FILE.AUX}
                  \immediate\write\@partaux{\relax}
          fi
          \@input{FILE.TEX}
          \clearpage
          \@writeckpt{FILE}
          if @filesw then \closeout \@partaux fi
          \@auxout := \@mainaux
     else \cp@FILE
  fi
 END

\@writeckpt{FILE} ==
 BEGIN
   if \@filesw = T
       \immediate\write on file \@partaux:
                 \@setckpt{FILE}{                  %% }
       for \reserved@a := \cl@@ckpt
          do  \immediate\write on file \@partaux:
                  \global\string\setcounter
                      {eval(\reserved@a)}{eval(\c@eval(\reserved@a))}
          od                                     %% {
       \immediate\write on file \@partaux:  }
   fi
 END

\@setckpt{FILE}{LIST} ==
 BEGIN
```

```
  G \cp@FILE := LIST
END

INITIALIZATION
  \@tempswa := T
```

End of historical LaTeX 2.09 comments.

\@mainaux
\@partaux

```
\newwrite\@mainaux
\newwrite\@partaux
```

(*End of definition for* \@mainaux *and* \@partaux.)

\if@filesw
\if@partsw

```
\newif\if@filesw \@fileswtrue
\newif\if@partsw \@partswfalse
```

(*End of definition for* \if@filesw *and* \if@partsw.)

\@clubpenalty

This stores the current normal (non-infinite) value of \clubpenalty; it should therefore be reset whenever the normal value is changed (as in the bibliography in the standard styles).

```
\newcount\@clubpenalty
\@clubpenalty \clubpenalty
```

(*End of definition for* \@clubpenalty.)

\document

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩  {\document}{Added hook to load l3backend code}%
⟨*2ekernel | latexrelease⟩
\def\document{%
```

We do cancel the grouping as part of the \begin handling (this is now done inside \begin instead) so that the env/⟨*env*⟩/begin hook is not hidden inside \begingroup ... \endgroup.

```
%  \endgroup
  \UseOneTimeHook{begindocument/before}%
  \@kernel@after@begindocument@before
```

Added hook to load l3backend code:

```
  \@expl@sys@load@backend@@
  \ifx\@unusedoptionlist\@empty\else
    \@latex@warning@no@line{Unused global option(s):^^J%
            \@spaces[\@unusedoptionlist]}%
  \fi
  \@colht\textheight
  \@colroom\textheight \vsize\textheight
  \columnwidth\textwidth
  \@clubpenalty\clubpenalty
  \if@twocolumn
    \advance\columnwidth -\columnsep
```

```
    \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
  \fi
  \hsize\columnwidth \linewidth\hsize
  \begingroup\@floatplacement\@dblfloatplacement
    \makeatletter\let\@writefile\@gobbletwo
    \global \let \@multiplelabels \relax
    \@input{\jobname.aux}%
  \endgroup
  \if@filesw
    \immediate\openout\@mainaux\jobname.aux
    \immediate\write\@mainaux{\relax}%
  \fi
```

Dateline 1991/03/26: FMi added `\process@table` to support NFSS; This will also work with old lfonts if no other style defines `\process@table`. The following line forces the initialization of the math fonts.

```
  \process@table
  \let\glb@currsize\@empty  % Force math initialization.
  \normalsize
  \everypar{}%
```

So that punctuation in headings is not disturbed by verbatim or other local changes to the space factor codes, save the document default here. This will be locally reset by the output routine. For special cases a class may want to define `\normalsfcodes` directly, in case that definition will be used. (This is an old bug, problem existed in LaTeX2.0x and plain TeX.)

```
  \ifx\normalsfcodes\@empty
    \ifnum\sfcode`\.=\@m
      \let\normalsfcodes\frenchspacing
    \else
      \let\normalsfcodes\nonfrenchspacing
    \fi
  \fi
```

For similar reasons also save the default language, this will be reset locally in the output routine. In particular it allows hyphenation in the page head even if the page break happens in verbatim. If this has already been set by a package, set to the value of `\language` at this point.

```
  \ifx\document@default@language\m@ne
    \chardef\document@default@language\language
  \fi
```

Way back in 1991 (08/26) FMi & RmS set the `\@noskipsec` switch to true in the preamble and to false here. This was done to trap lists and related text in the preamble but it does not catch everything; hence Change 1.1g was introduced.

```
  \@noskipsecfalse
  \let \@refundefined \relax
```

Just before disabling the preamble commands we execute the begin document hook which contains any code contributed by `\AtBeginDocument`. Also disable the gathering of the file list, if no `\listfiles` has been issued. `\AtBeginDocument` is redefined at this point so that and such commands that get into the hook do not chase their tail...

```
\@kernel@before@begindocument
\UseOneTimeHook{begindocument}%
\@kernel@after@begindocument
```

Most of the following assignments will be done globally in case the user adds something like `\begin{multicols}` to the document hook, i.e. starts are group in `\begin{document}`.

Since a value of exactly 0pt for `\topskip` causes `\twocolumn[]` to misbehave, we add this check, hoping that it will not cause any problems elsewhere.

```
\ifdim\topskip<1sp\global\topskip 1sp\relax\fi
\global\@maxdepth\maxdepth
\global\let\@begindocumenthook\@undefined
\ifx\@listfiles\@undefined
  \global\let\@filelist\relax
  \global\let\@addtofilelist\@gobble
\fi
```

At the very end we disable all preamble commands. This has to happen after the begin document hooks was executed so that this hook can still use such commands.

```
\gdef\do##1{\global\let ##1\@notprerr}%
\@preamblecmds
```

The next line saves tokens and also allows `\@nodocument` to be used directly to trap preamble errors.

```
\global\let \@nodocument \relax
```

The next line is a pure safety measure in case a do list is ever expanded at the wrong place. In addition it will save a few tokens to get rid of the above definition.

```
\global\let\do\noexpand
```

```
\UseOneTimeHook{begindocument/end}%
```

Use of the hook might mean that we are already in horizontal mode, so ignore the space after `\begin{document}`.

```
\ignorespaces}
```

Provide a global definition for `\do` as well, so that it is already defined in the preamble and not late as `\begin{document}` overwriting some definition given by the unsuspecting user in the preamble.

```
\let\do\noexpand
```

The `begindocument` hook already existed in the kernel since 1994 under the name `\atbegindocumenthook` the additional ones are originally from the etoolbox package under the names `\@endpreamblehook` `\afterpreamble`.

```
\NewHook{begindocument}
\NewHook{begindocument/before}
\NewHook{begindocument/end}
```

Above we used two kernel only hooks to be run after the public `begindocument/before` and after `begindocument` hooks.

\@kernel@after@begindocument@before
\@kernel@before@begindocument
`\@kernel@after@begindocument`

In `\@kernel@after@begindocument@before` we already place one action: drop the fast execution code for the `env/document/begin` hook. That hook marks the end of the preamble and should therefore only be run once. In a normal document that is anyway the case (so the code would just sit there taking up space afterwards, which these days is rather harmless), however, in more complicated scenarios where several full documents are combined to a single document it might get applied several times with harmful effects.

We therefore explicitly drop it at this point. the coding is somewhat obscure due to the name of the macro which requires constructing.

```
\edef \@kernel@after@begindocument@before {%
  \let\expandafter\noexpand\csname
       __hook env/document/begin\endcsname
  \noexpand\@empty}
```

These internal hooks are already declared earlier (in `ltexpl`) so that other modules could write to them.

```
%\let \@kernel@before@begindocument \@empty
%\let \@kernel@after@begindocument  \@empty
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2017/04/15}%
⟨latexrelease⟩  {\document}{Save language for hyphenation}%
⟨latexrelease⟩
⟨latexrelease⟩\def\document{\endgroup
⟨latexrelease⟩  \ifx\@unusedoptionlist\@empty\else
⟨latexrelease⟩    \@latex@warning@no@line{Unused global option(s):^^J%
⟨latexrelease⟩            \@spaces[\@unusedoptionlist]}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \@colht\textheight
⟨latexrelease⟩  \@colroom\textheight \vsize\textheight
⟨latexrelease⟩  \columnwidth\textwidth
⟨latexrelease⟩  \@clubpenalty\clubpenalty
⟨latexrelease⟩  \if@twocolumn
⟨latexrelease⟩    \advance\columnwidth -\columnsep
⟨latexrelease⟩    \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \hsize\columnwidth \linewidth\hsize
⟨latexrelease⟩  \begingroup\@floatplacement\@dblfloatplacement
⟨latexrelease⟩    \makeatletter\let\@writefile\@gobbletwo
⟨latexrelease⟩    \global \let \@multiplelabels \relax
⟨latexrelease⟩    \@input{\jobname.aux}%
⟨latexrelease⟩  \endgroup
⟨latexrelease⟩  \if@filesw
⟨latexrelease⟩    \immediate\openout\@mainaux\jobname.aux
⟨latexrelease⟩    \immediate\write\@mainaux{\relax}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \process@table
⟨latexrelease⟩  \let\glb@currsize\@empty  % Force math initialization.
⟨latexrelease⟩  \normalsize
⟨latexrelease⟩  \everypar{}%
⟨latexrelease⟩  \ifx\normalsfcodes\@empty
⟨latexrelease⟩    \ifnum\sfcode`\.=\@m
⟨latexrelease⟩      \let\normalsfcodes\frenchspacing
⟨latexrelease⟩    \else
⟨latexrelease⟩      \let\normalsfcodes\nonfrenchspacing
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \ifx\document@default@language\m@ne
⟨latexrelease⟩    \chardef\document@default@language\language
⟨latexrelease⟩  \fi
```

```
⟨latexrelease⟩  \@noskipsecfalse
⟨latexrelease⟩  \let \@refundefined \relax
⟨latexrelease⟩  \let\AtBeginDocument\@firstofone
⟨latexrelease⟩  \@begindocumenthook
⟨latexrelease⟩  \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
⟨latexrelease⟩  \global\@maxdepth\maxdepth
⟨latexrelease⟩  \global\let\@begindocumenthook\@undefined
⟨latexrelease⟩  \ifx\@listfiles\@undefined
⟨latexrelease⟩    \global\let\@filelist\relax
⟨latexrelease⟩    \global\let\@addtofilelist\@gobble
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \gdef\do##1{\global\let ##1\@notprerr}%
⟨latexrelease⟩  \@preamblecmds
⟨latexrelease⟩  \global\let \@nodocument \relax
⟨latexrelease⟩  \global\let\do\noexpand
⟨latexrelease⟩  \ignorespaces}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\document}{Save language for hyphenation}
⟨latexrelease⟩\def\document{\endgroup
⟨latexrelease⟩  \ifx\@unusedoptionlist\@empty\else
⟨latexrelease⟩    \@latex@warning@no@line{Unused global option(s):^^J%
⟨latexrelease⟩            \@spaces[\@unusedoptionlist]}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \@colht\textheight
⟨latexrelease⟩  \@colroom\textheight \vsize\textheight
⟨latexrelease⟩  \columnwidth\textwidth
⟨latexrelease⟩  \@clubpenalty\clubpenalty
⟨latexrelease⟩  \if@twocolumn
⟨latexrelease⟩    \advance\columnwidth -\columnsep
⟨latexrelease⟩    \divide\columnwidth\tw@ \hsize\columnwidth
⟨latexrelease⟩    \@firstcolumntrue
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \hsize\columnwidth \linewidth\hsize
⟨latexrelease⟩  \begingroup\@floatplacement\@dblfloatplacement
⟨latexrelease⟩    \makeatletter\let\@writefile\@gobbletwo
⟨latexrelease⟩    \global \let \@multiplelabels \relax
⟨latexrelease⟩    \@input{\jobname.aux}%
⟨latexrelease⟩  \endgroup
⟨latexrelease⟩  \if@filesw
⟨latexrelease⟩    \immediate\openout\@mainaux\jobname.aux
⟨latexrelease⟩    \immediate\write\@mainaux{\relax}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \process@table
⟨latexrelease⟩  \let\glb@currsize\@empty
⟨latexrelease⟩  \normalsize
⟨latexrelease⟩  \everypar{}%
⟨latexrelease⟩  \ifx\normalsfcodes\@empty
⟨latexrelease⟩    \ifnum\sfcode`\.=\@m
⟨latexrelease⟩      \let\normalsfcodes\frenchspacing
⟨latexrelease⟩    \else
⟨latexrelease⟩      \let\normalsfcodes\nonfrenchspacing
⟨latexrelease⟩    \fi
```

```
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \@noskipsecfalse
⟨latexrelease⟩  \let \@refundefined \relax
⟨latexrelease⟩  \let\AtBeginDocument\@firstofone
⟨latexrelease⟩  \@begindocumenthook
⟨latexrelease⟩  \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
⟨latexrelease⟩  \global\@maxdepth\maxdepth
⟨latexrelease⟩  \global\let\@begindocumenthook\@undefined
⟨latexrelease⟩  \ifx\@listfiles\@undefined
⟨latexrelease⟩    \global\let\@filelist\relax
⟨latexrelease⟩    \global\let\@addtofilelist\@gobble
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \gdef\do##1{\global\let ##1\@notprerr}%
⟨latexrelease⟩  \@preamblecmds
⟨latexrelease⟩  \global\let \@nodocument \relax
⟨latexrelease⟩  \global\let\do\noexpand
⟨latexrelease⟩  \ignorespaces}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\document
```

(*End of definition for* `\document` *and others.*)

`\normalsfcodes` The setting of `\@empty` is just a flag. This command may be defined in a class or package file. If it is still `\@empty` at `\begin{document}` it will be defined to be `\frenchspacing` or `\nonfrenchspacing`, depending on which of those appears to be in effect at that point.

```
\let\normalsfcodes\@empty
```

(*End of definition for* `\normalsfcodes`.)

`\nofiles` Set `\@fileswfalse` which suppresses the places where LaTeX makes `\immediate` writes. The `\makeindex` and `\makeglossary` are disabled. `\protected@write` is redefined not to write to the file specified, but rather to write a blank line to the log file. This ensures that a ⟨*whatsit*⟩ node is still created, and so spacing is not affected by the `\nofiles` command; to ensure this more generally, the `\if@nobreak` test is needed.

```
\def\nofiles{%
  \@fileswfalse
  \typeout{No auxiliary output files.^^J}%
  \long\def\protected@write##1##2##3%
    {\write\m@ne{}\if@nobreak\ifvmode\nobreak\fi\fi}%
  \let\makeindex\relax
  \let\makeglossary\relax}
\@onlypreamble\nofiles
```

(*End of definition for* `\nofiles`.)

`\protected@write` This takes three arguments: an output stream, some initialization code, and some text to write. It then writes this, with appropriate handling of `\protect` and `\thepage`.

```
\long\def \protected@write#1#2#3{%
      \begingroup
       \let\thepage\relax
       #2%
       \let\protect\@unexpandable@protect
       \edef\reserved@a{\write#1{#3}}%
```

```
      \reserved@a
    \endgroup
    \if@nobreak\ifvmode\nobreak\fi\fi
}
```

(*End of definition for* `\protected@write`.)

```
\let\@auxout=\@mainaux
```

`\include` `\includeonly`

In the definition of `\include`, `\def\reserved@b` changed to `\edef\reserved@b` to be consistent with the `\edef` in `\includeonly`. (Suggested by Rainer Schöpf & Frank Mittelbach. Change made 20 Jul 88.)

Changed definition of `\include` to allow space at end of file name — otherwise, typing `\include{foo }` would cause LaTeX to overwrite `foo.tex`. Change made 24 May 89, suggested by Rainer Schöpf and Frank Mittelbach

Made `\include` check for being used inside an `\include`'d file, as this will not work and cause surprising results.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                    {\includeonly}{Spaces in file names}%
```

```
\def\include#1{\relax
  \ifnum\@auxout=\@partaux
    \@latex@error{\string\include\space cannot be nested}\@eha
  \else
```

Here the normalization will add `.tex` for all files, (it uses the same normalization as the hooks), so we need to remove that manually. `\@strip@tex@ext` does that.

```
    \set@curr@file{#1}%
    \edef\@curr@file{\@strip@tex@ext\@curr@file}%
```

For historical reasons `\@include` expects an argument delimited by a space. This is kept (though unnecessary now) to avoid errors in other packages that use `\@include` directly.

```
    \expandafter\@include\expandafter{\@curr@file} % deliberate space
  \fi}
```

Here in `\includeonly` we also need to strip `.tex` after normalization:

```
\def\includeonly#1{%
  \@partswtrue
```

Because the argument to `\includeonly` is a comma-separated list of filenames where there may be comma's preceding some of the filenames or trailing them. Therefore we need to take the list apart, remove the unwanted spaces while leaving the spaces *in* the filenames intact.

```
  \let\@partlist\@empty
  \@for\reserved@a:=#1 \do
    {%
      \expandafter\set@curr@file\expandafter{\reserved@a}%
      \ifx\@partlist\@empty
        \edef\@partlist{\@strip@tex@ext\@curr@file}%
      \else
        \edef\@partlist{\@partlist,\@strip@tex@ext\@curr@file}%
      \fi
    }%
  }
\@onlypreamble\includeonly
```

(*End of definition for* `\include` *and* `\includeonly`.)

`\@strip@tex@ext`
`\@strip@tex@ext@aux`

These macros take a (`\detokenize`d file name and remove any `.tex` extension). Extra care is taken to not remove the string `.tex` from the middle of a file name: it is only removed if it's the very last thing in the file name.

```
\def\reserved@a#1{%
\def\@strip@tex@ext##1{%
  \expandafter\@strip@tex@ext@aux
    ##1\@nil\@nil
     #1\@nil\relax\@nnil}
\def\@strip@tex@ext@aux##1#1\@nil##2\@nnil{%
  \ifx\relax##2\@empty
    \expandafter\@cdr\expandafter\@empty\@cdr{}##1%
  \else##1\fi}}%
\expandafter\reserved@a
\expandafter{\detokenize{.tex}}
⟨/2ekernel | latexrelease⟩
```

(*End of definition for* `\@strip@tex@ext` *and* `\@strip@tex@ext@aux`.)

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\includeonly}{Spaces in file names}%
⟨latexrelease⟩
⟨latexrelease⟩\def\includeonly#1{%
⟨latexrelease⟩  \@partswtrue
⟨latexrelease⟩  \set@curr@file{\zap@space#1 \@empty}%
⟨latexrelease⟩  \let\@partlist\@curr@file
⟨latexrelease⟩  }
⟨latexrelease⟩
⟨latexrelease⟩\def\include#1{\relax
⟨latexrelease⟩  \ifnum\@auxout=\@partaux
⟨latexrelease⟩    \@latex@error{\string\include\space cannot be nested}\@eha
⟨latexrelease⟩  \else
⟨latexrelease⟩    \set@curr@file{#1 }%
⟨latexrelease⟩    \expandafter\@include\@curr@file
⟨latexrelease⟩  \fi}
⟨latexrelease⟩
⟨latexrelease⟩\let\@strip@tex@ext\@undefined
⟨latexrelease⟩\let\@strip@tex@ext@aux\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\includeonly}{Spaces in file names}%
⟨latexrelease⟩\def\includeonly#1{%
⟨latexrelease⟩  \@partswtrue
⟨latexrelease⟩  \edef\@partlist{\zap@space#1 \@empty}}
⟨latexrelease⟩
⟨latexrelease⟩\def\include#1{\relax
⟨latexrelease⟩  \ifnum\@auxout=\@partaux
⟨latexrelease⟩    \@latex@error{\string\include\space cannot be nested}\@eha
⟨latexrelease⟩  \else \@include#1 \fi}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@include

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                    {\@include}{Spaces in file names and hooks}%
```

```
\def\@include#1 {%
\ifx\@nodocument\relax
  \clearpage
  \if@filesw
    \immediate\write\@mainaux{\string\@input{#1.aux}}%
  \fi
  \@tempswatrue
  \if@partsw
    \@tempswafalse
    \edef\reserved@b{#1}%
    \@for\reserved@a:=\@partlist\do
      {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
  \fi
  \if@tempswa
    \let\@auxout\@partaux
    \if@filesw
      \immediate\openout\@partaux "#1.aux"
      \immediate\write\@partaux{\relax}%
    \fi
```

Now before going to the hooks we need to set `\CurrentFile`:

```
%-----------------------------------------
    \@filehook@set@CurrentFile
```

Execute the `before` hooks just after we switched the `.aux` file ...

```
    \UseHook{include/before}%
    \UseOneTimeHook{include/#1/before}%
%-----------------------------------------
    \@input@{#1.tex}%
%-----------------------------------------
```

... then `end` hooks ...

```
    \UseOneTimeHook{include/#1/end}%
    \UseHook{include/end}%
%-----------------------------------------
    \clearpage
%-----------------------------------------
```

... and after the `\clearpage` the `after` hooks followed by another `\clearpage` just in case new material got added (after all we need to be in well defined state after the `\include`).

```
    \UseOneTimeHook{include/#1/after}%
    \UseHook{include/after}%
    \clearpage
%-----------------------------------------
    \@writeckpt{#1}%
    \if@filesw
      \immediate\closeout\@partaux
```

```
    \fi
  \else
```

If the file is not included, reset `\deadcycles`, so that a long list of non-included files does not generate an 'Output loop' error.

```
    \deadcycles\z@
    \@nameuse{cp@#1}%
```

We also execute a hook in this case, first a general used for every include file that is exclude and then a specific one that contains the name of the include file.

```
%----------------------------------------
    \UseHook{include/excluded}%
    \UseOneTimeHook{include/#1/excluded}%
%----------------------------------------
  \fi
  \let\@auxout\@mainaux
```

```
\else
\@latex@warning{%
  \noexpand\include should only be used after \string\begin{document}}%
\@input@{#1}%
\fi}
```

Now declare the non-generic `include` hooks used above:

```
\NewHook{include/before}
\NewReversedHook{include/end}
\NewReversedHook{include/after}
\NewHook{include/excluded}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                 {\@include}{Spaces in file names and hooks}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\def\@include#1 {%
⟨latexrelease⟩\ifx\@nodocument\relax
⟨latexrelease⟩  \clearpage
⟨latexrelease⟩  \if@filesw
⟨latexrelease⟩    \immediate\write\@mainaux{\string\@input{#1.aux}}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \@tempswatrue
⟨latexrelease⟩  \if@partsw
⟨latexrelease⟩    \@tempswafalse
⟨latexrelease⟩    \edef\reserved@b{#1}%
⟨latexrelease⟩    \@for\reserved@a:=\@partlist\do
⟨latexrelease⟩      {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@tempswa
⟨latexrelease⟩    \let\@auxout\@partaux
⟨latexrelease⟩    \if@filesw
⟨latexrelease⟩      \immediate\openout\@partaux "#1.aux"
⟨latexrelease⟩      \immediate\write\@partaux{\relax}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩    \@filehook@set@CurrentFile
⟨latexrelease⟩    \UseHook{include/before}%
⟨latexrelease⟩    \UseOneTimeHook{include/#1/before}%
```

```
⟨latexrelease⟩    \@input@{#1.tex}%
⟨latexrelease⟩    \UseOneTimeHook{include/#1/end}%
⟨latexrelease⟩    \UseHook{include/end}%
⟨latexrelease⟩    \clearpage
⟨latexrelease⟩    \UseOneTimeHook{include/#1/after}%
⟨latexrelease⟩    \UseHook{include/after}%
⟨latexrelease⟩    \clearpage
⟨latexrelease⟩    \@writeckpt{#1}%
⟨latexrelease⟩    \if@filesw
⟨latexrelease⟩      \immediate\closeout\@partaux
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \deadcycles\z@
⟨latexrelease⟩    \@nameuse{cp@#1}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \let\@auxout\@mainaux
⟨latexrelease⟩\else
⟨latexrelease⟩\@latex@warning{%
⟨latexrelease⟩  \noexpand\include should only be used after \string\begin{document}}%
⟨latexrelease⟩\@input@{#1}%
⟨latexrelease⟩\fi}
⟨latexrelease⟩\NewHook{include/before}
⟨latexrelease⟩\NewReversedHook{include/end}
⟨latexrelease⟩\NewReversedHook{include/after}
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@include}{Spaces in file names and hooks}%
⟨latexrelease⟩\def\@include#1 {%
⟨latexrelease⟩  \clearpage
⟨latexrelease⟩  \if@filesw
⟨latexrelease⟩    \immediate\write\@mainaux{\string\@input{#1.aux}}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \@tempswatrue
⟨latexrelease⟩  \if@partsw
⟨latexrelease⟩    \@tempswafalse
⟨latexrelease⟩    \edef\reserved@b{#1}%
⟨latexrelease⟩    \@for\reserved@a:=\@partlist\do
⟨latexrelease⟩      {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@tempswa
⟨latexrelease⟩    \let\@auxout\@partaux
⟨latexrelease⟩    \if@filesw
⟨latexrelease⟩      \immediate\openout\@partaux #1.aux
⟨latexrelease⟩      \immediate\write\@partaux{\relax}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩    \@input@{#1.tex}%
⟨latexrelease⟩    \clearpage
⟨latexrelease⟩    \@writeckpt{#1}%
⟨latexrelease⟩    \if@filesw
⟨latexrelease⟩      \immediate\closeout\@partaux
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \deadcycles\z@
⟨latexrelease⟩    \@nameuse{cp@#1}%
⟨latexrelease⟩  \fi
```

```
⟨latexrelease⟩   \let\@auxout\@mainaux}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@include`.)

`\@writeckpt`

```
\def\@writeckpt#1{%
  \if@filesw
    \immediate\write\@partaux{\string\@setckpt{#1}\@charlb}%
    {\let\@elt\@wckptelt \cl@@ckpt}%
    \immediate\write\@partaux{\@charrb}%
  \fi}
```

(*End of definition for* `\@writeckpt`.)

`\@wckptelt`

```
\def\@wckptelt#1{%
  \immediate\write\@partaux{%
    \string\setcounter{#1}{\the\@nameuse{c@#1}}}}
```

(*End of definition for* `\@wckptelt`.)

`\@setckpt` RmS 93/08/31: introduced `\@setckpt`

```
\def\@setckpt#1{\global\@namedef{cp@#1}}
```

(*End of definition for* `\@setckpt`.)

`\@charlb` `\@charrb` The following defines `\@charlb` and `\@charrb` to be `{` and `}`, respectively with `\catcode` 11.

```
{\catcode`[=1 \catcode`]=2
\catcode`{=11 \catcode`}=11
\gdef\@charlb[{]
\gdef\@charrb[}]
]% }brace matching
```

(*End of definition for* `\@charlb` *and* `\@charrb`.)

1.1 Safe Input Macros

`\@curr@file` `\set@curr@file` File name handling is done by generating a csname from the provided file name (which means that UTF-8 octets gets turned into strings as this is what happens if they appear in a csname due to the code in `utf8.def`). By setting `\escapchar` to `-1` we ensure that we don't get a backslash in front. As a result we end up with all characters as catcode 12 (plus spaces). We then sometimes add quotes around the construct (removing any existing inner quotes). Sometimes we only remove the quotes if they have been supplied by the user. There is clearly some room for improvement.

A side effect of the new code is that we will see quotes around file name displays where there haven't been any before.

For compatibility with existing code using `{abc}.tex` or `{one.two}.png`, an initial brace group is discarded before expansion and `\string` is applied. The content of the brace group is discarded. This means that a leading space will be lost unless protected (by `{ }` or `" "` or `\space`) but filenames with a space are hopefully rare.

The definition below is from 2019 and only used during kernel bootstrapping, later on in `ltfilehook.dtx` it will get overwritten.

```
\def\set@curr@file#1{%
  \begingroup
    \escapechar\m@ne
    \xdef\@curr@file{%
      \expandafter\expandafter\expandafter\unquote@name
      \expandafter\expandafter\expandafter{%
      \expandafter\string
        \csname\@firstofone#1\@empty\endcsname}}%
  \endgroup
}
```

(*End of definition for* `\@curr@file` *and* `\set@curr@file`.)

`\quote@name`
`\quote@@name`
`\unquote@name`

Quoting spaces

```
a b c     -> "a b c"
"a b c"   -> "a b c"
a" "b" "c -> "a b c"
          -> ""
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\quote@name}{Quote file names}%
\def\quote@name#1{"\quote@@name#1\@gobble""}
\def\quote@@name#1"{#1\quote@@name}
```

and removing quotes ...

```
\def\unquote@name#1{\quote@@name#1\@gobble"}
```

(*End of definition for* `\quote@name`, `\quote@@name`, *and* `\unquote@name`.)

`\IfFileExists`

```
\DeclareRobustCommand\IfFileExists[1]{%
  \set@curr@file{#1}%
  \expandafter\IfFileExists@\expandafter{\@curr@file}}
```

(*End of definition for* `\IfFileExists`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\quote@name}{Quote file names}%
⟨latexrelease⟩
⟨latexrelease⟩\let\quote@name\@undefined
⟨latexrelease⟩\let\quote@@name\@undefined
⟨latexrelease⟩\let\unquote@name\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\long\def \IfFileExists#1#2#3{%
⟨latexrelease⟩  \openin\@inputcheck#1 %
⟨latexrelease⟩  \ifeof\@inputcheck
⟨latexrelease⟩    \ifx\input@path\@undefined
⟨latexrelease⟩      \def\reserved@a{#3}%
⟨latexrelease⟩    \else
```

```
⟨latexrelease⟩      \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \closein\@inputcheck
⟨latexrelease⟩    \edef\@filef@und{#1 }%
⟨latexrelease⟩    \def\reserved@a{#2}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \reserved@a}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\IfFileExists@` `\IfFileExists@@`

Argument `#1` is `\@curr@file` so catcode 12 string with no quotes.

The original definition picked up arguments `#2` and `#3` in a way that they couldn't contain unbalanced conditionals. A better implementation would have been not to pick up the arguments at all but instead use the usual `\@firstoftwo` and `\secondoftwo`. However, that changes how `#` is interpreted and so we can't do that nowadays without invalidating a lot of code. Therefore the somewhat curious construction near the end.

Earlier versions used `\openin` here, but this led to two code paths, one in expl3 and one here. To avoid that, and as the expl3 approach works by expansion, we use that here. As we need the file name to include the path, the actual expl3 function used is not the file existence test!

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2023/06/01}%
⟨latexrelease⟩                  {\IfFileExists@}{IfFileExists}
\long\def \IfFileExists@#1#2#3{%
  \edef\@filef@und{\IfFileExists@@{#1}}%
```

The expl3 function regards an empty argument as nothing at all, but the LaTeX 2_ε convention is that this is equal to the special `.tex` file.

```
  \ifx\@filef@und\@empty
    \if\relax\detokenize{#1}\relax
      \let\reserved@a\@firstoftwo
      \def\@filef@und{".tex" }%
    \else
      \let\reserved@a\@secondoftwo
    \fi
  \else
    \let\reserved@a\@firstoftwo
    \edef\@filef@und{"\@filef@und" }%
  \fi
```

This is just there so that any `#` inside `#2` or `#3` needs doubling (as that was the case in the past).

```
  \expandafter\def\expandafter\reserved@a
    \expandafter{\reserved@a{#2}{#3}}%
\reserved@a}
```

Pipes are not really files, but at the document level they are supported. To quickly trim of any leading spaces, we use a blank test and `\use:n` rather than `\tl_trim_spaces:n` for speed as we don't care about the end of the input.

```
\ExplSyntaxOn
\cs_new:Npn \IfFileExists@@ #1
```

```
  {
    \tl_if_blank:nF {#1}
      {
        \tl_if_head_eq_charcode:oNTF { \use:n #1 } |
          {#1}
          { \file_full_name:n {#1} }
      }
  }
\cs_generate_variant:Nn \tl_if_head_eq_charcode:nNTF { o }
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                 {\IfFileExists@}{IfFileExists}
⟨latexrelease⟩
⟨latexrelease⟩\long\def \IfFileExists@#1#2#3{%
⟨latexrelease⟩  \openin\@inputcheck"#1" %
⟨latexrelease⟩  \ifeof\@inputcheck
⟨latexrelease⟩    \ifx\input@path\@undefined
⟨latexrelease⟩      \let\reserved@a\@secondoftwo
⟨latexrelease⟩    \else
⟨latexrelease⟩      \def\reserved@a{\@iffileonpath{#1}}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \closein\@inputcheck
⟨latexrelease⟩    \edef\@filef@und{"#1" }%
⟨latexrelease⟩    \let\reserved@a\@firstoftwo
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \expandafter\def\expandafter\reserved@a
⟨latexrelease⟩    \expandafter{\reserved@a{#2}{#3}}%
⟨latexrelease⟩\reserved@a}
⟨latexrelease⟩\let\IfFileExists@@\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\IfFileExists@}{IfFileExists}
⟨latexrelease⟩
⟨latexrelease⟩\long\def \IfFileExists@#1#2#3{%
⟨latexrelease⟩  \openin\@inputcheck"#1" %
⟨latexrelease⟩  \ifeof\@inputcheck
⟨latexrelease⟩    \ifx\input@path\@undefined
⟨latexrelease⟩      \def\reserved@a{#3}%
⟨latexrelease⟩    \else
⟨latexrelease⟩      \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \closein\@inputcheck
⟨latexrelease⟩    \edef\@filef@und{"#1" }%
⟨latexrelease⟩    \def\reserved@a{#2}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \reserved@a}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\IfFileExists@}{IfFileExists}
```

```
⟨latexrelease⟩
⟨latexrelease⟩\let\IfFileExists@\@undefined
⟨latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\IfFileExists@` *and* `\IfFileExists@@`.)

`\@iffileonpath` If the file is not found by `\openin`, and `\input@path` is defined, look in all the directories specified in `\input@path`.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\@iffileonpath}{Quote file names}
\long\def\@iffileonpath#1{%
  \let\reserved@a\@secondoftwo
  \expandafter\@tfor\expandafter\reserved@b\expandafter
              :\expandafter=\input@path\do{%
    \openin\@inputcheck\expandafter\quote@name\expandafter{\reserved@b#1} %
    \ifeof\@inputcheck\else
      \edef\@filef@und{\expandafter\quote@name\expandafter{\reserved@b#1} }%
      \let\reserved@a\@firstoftwo%
      \closein\@inputcheck
      \@break@tfor
    \fi}%
  \reserved@a}
```

(*End of definition for* `\@iffileonpath`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\quote@name}{Quote file names}
⟨latexrelease⟩
⟨latexrelease⟩\long\def\@iffileonpath#1{%
⟨latexrelease⟩  \let\reserved@a\@secondoftwo
⟨latexrelease⟩  \expandafter\@tfor\expandafter\reserved@b\expandafter
⟨latexrelease⟩              :\expandafter=\input@path\do{%
⟨latexrelease⟩    \openin\@inputcheck\reserved@b#1 %
⟨latexrelease⟩    \ifeof\@inputcheck\else
⟨latexrelease⟩      \edef\@filef@und{\reserved@b#1 }%
⟨latexrelease⟩      \let\reserved@a\@firstoftwo%
⟨latexrelease⟩      \closein\@inputcheck
⟨latexrelease⟩      \@break@tfor
⟨latexrelease⟩    \fi}%
⟨latexrelease⟩  \reserved@a}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\InputIfFileExists` Now define `\InputIfFileExists` to input `#1` if it seems to exist. Immediately prior to the input, `#2` is executed. If the file `#1` does not exist, execute '`#3`'.

This here is a temporary definition for the kernel. The real one comes somewhat later in the file `ltfilehook.dtx`.

```
\DeclareRobustCommand \InputIfFileExists[2]{%
  \IfFileExists{#1}%
    {%
  \expandafter\@swaptwoargs\expandafter
      {\@filef@und}{#2\@addtofilelist{#1}\@@input}}}
```

(*End of definition for* `\InputIfFileExists`.)

`\@swaptwoargs` Swap two arguments and return them unbraced (like `\@firstoftwo` etc).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩  {\@swaptwoargs}{Don't lose the file name}%
\long\def\@swaptwoargs#1#2{#2#1}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\@swaptwoargs}{Don't lose the file name}%
⟨latexrelease⟩\let\@swaptwoargs\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@swaptwoargs`.)

`\input` Input a file: if the argument is given in braces use safe input macros, otherwise use TEX's primitive `\input` command (which is called `\@@input` in LATEX).

```
\def\input{\@ifnextchar\bgroup\@iinput\@@input}
```

(*End of definition for* `\input`.)

`\@iinput` Define `\@iinput` (i.e., `\input`) in terms of `\InputIfIfileExists`.

Changes to `\@iinput`: adapt to the changes to `\@missingfileerror`.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@iinput}{Change in file error handling}%
\def\@iinput#1{%
  \InputIfFileExists{#1}{}%
  {\filename@parse\@curr@file
   \edef\reserved@a{\noexpand\@missingfileerror
     {\filename@area\filename@base}%
     {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
```

This line now just sets `\@missingfile@`⟨*part*⟩:

```
   \reserved@a
```

Now here we have to use it. The file here is guaranteed to exist, because `\@missingfileerror` ensures so, but we have to use `\InputIfFileExists` because it executes the file hooks.

```
   \edef\reserved@a{\noexpand\@iinput{%
     \@missingfile@area\@missingfile@base.\@missingfile@ext}}%
   \reserved@a}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\@iinput}{Quote file names}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@iinput#1{%
⟨latexrelease⟩  \InputIfFileExists{#1}{}%
⟨latexrelease⟩  {\filename@parse\@curr@file
⟨latexrelease⟩   \edef\reserved@a{\noexpand\@missingfileerror
⟨latexrelease⟩     {\filename@area\filename@base}%
⟨latexrelease⟩     {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
⟨latexrelease⟩   \reserved@a}}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@iinput}{Quote file names}%
⟨latexrelease⟩\def\@iinput#1{%
⟨latexrelease⟩  \InputIfFileExists{#1}{}%
⟨latexrelease⟩  {\filename@parse{#1}%
⟨latexrelease⟩   \edef\reserved@a{\noexpand\@missingfileerror
⟨latexrelease⟩     {\filename@area\filename@base}%
⟨latexrelease⟩     {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
⟨latexrelease⟩   \reserved@a}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@iinput`.)

`\@input` Define `\@input` in terms of `\IfIfileExists`. So this is a 'safe input' command, but the files input are not listed by `\listfiles`.

We don't want `.aux`, `.toc` files etc be listed by `\listfiles`. However, something like `.bbl` probably should be listed and thus should be implemented not by `\@input`.

```
\def\@input#1{%
  \IfFileExists{#1}{\@@input\@filef@und}{\typeout{No file #1.}}}
```

(*End of definition for* `\@input`.)

`\@input@` Version of `\@input` that does add the file to `\@filelist`.

```
\def\@input@#1{\InputIfFileExists{#1}{}{\typeout{No file #1.}}}
```

(*End of definition for* `\@input@`.)

`\@missingfileerror` This 'error' command avoids TEX's primitive missing file loop.

Missing file error. Prompt for a new filename, offering a default extension.

Changes to `\@missingfileerror`: rather than trying to input the file by force, now `\@missingfileerror` just returns three `\@missingfile@`⟨*part*⟩ and the caller macro is responsible for doing the right thing with it.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩        {\@missingfileerror}{Do not load missing file immediately}%
\gdef\@missingfileerror#1#2{%
     \typeout{^^J! LaTeX Error: File ‘#1.#2’ not found.^^J^^J%
      Type X to quit or <RETURN> to proceed,^^J%
      or enter new name. (Default extension: #2)^^J}%
     \message{Enter file name: }%
      {\endlinechar\m@ne
```

```
        \global\read\m@ne to\@gtempa}%
    \ifx\@gtempa\@empty
```

If the user answers with ⟨*return*⟩, fallback to the `.tex` file (previously it did nothing).

```
      \let\@missingfile@area\@empty
      \let\@missingfile@base\@empty
      \def\@missingfile@ext{tex}%
    \else
```

Use `\batchmode\read-1 to` ⟨*tl*⟩ to end the TEX run, same as expl3 does (it was `\batchmode\@@end` before).

```
      \def\reserved@b{\batchmode\read-1 to \reserved@a}%
      \def\reserved@a{x}\ifx\reserved@a\@gtempa\reserved@b\fi
      \def\reserved@a{X}\ifx\reserved@a\@gtempa\reserved@b\fi
      \filename@parse\@gtempa
      \edef\filename@ext{%
        \ifx\filename@ext\relax#2\else\filename@ext\fi}%
     \edef\reserved@a{%
```

Only check `\IfFileExists` (it was `\InputIfFileExists`).

```
        \noexpand\IfFileExists
          {\filename@area\filename@base.\filename@ext}%
```

If the file exists, define `\@missingfile@`⟨*part*⟩.

```
          {\def\noexpand\@missingfile@area{\filename@area}%
           \def\noexpand\@missingfile@base{\filename@base}%
           \def\noexpand\@missingfile@ext {\filename@ext}}%
          {\noexpand\@missingfileerror
             {\filename@area\filename@base}{\filename@ext}}}%
      \reserved@a
    \fi
}
```

⟨/2ekernel | latexrelease⟩

⟨latexrelease⟩`\EndIncludeInRelease`

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩        {\@missingfileerror}{Do not load missing file immediately}%
⟨latexrelease⟩
⟨latexrelease⟩\gdef\@missingfileerror#1#2{%
⟨latexrelease⟩     \typeout{^^J! LaTeX Error: File ‘#1.#2’ not found.^^J^^J%
⟨latexrelease⟩      Type X to quit or <RETURN> to proceed,^^J%
⟨latexrelease⟩      or enter new name. (Default extension: #2)^^J}%
⟨latexrelease⟩     \message{Enter file name: }%
⟨latexrelease⟩      {\endlinechar\m@ne
⟨latexrelease⟩       \global\read\m@ne to\@gtempa}%
⟨latexrelease⟩    \ifx\@gtempa\@empty
⟨latexrelease⟩    \else
⟨latexrelease⟩      \def\reserved@a{x}\ifx\reserved@a\@gtempa\batchmode\@@end\fi
⟨latexrelease⟩      \def\reserved@a{X}\ifx\reserved@a\@gtempa\batchmode\@@end\fi
⟨latexrelease⟩      \filename@parse\@gtempa
⟨latexrelease⟩      \edef\filename@ext{%
⟨latexrelease⟩        \ifx\filename@ext\relax#2\else\filename@ext\fi}%
⟨latexrelease⟩     \edef\reserved@a{%
⟨latexrelease⟩       \noexpand\InputIfFileExists
⟨latexrelease⟩         {\filename@area\filename@base.\filename@ext}%
⟨latexrelease⟩         {}%
```

```
⟨latexrelease⟩          {\noexpand\@missingfileerror
⟨latexrelease⟩             {\filename@area\filename@base}{\filename@ext}}}%
⟨latexrelease⟩       \reserved@a
⟨latexrelease⟩     \fi}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@missingfileerror`.)

`\@obsoletefile` For compatibility with LaTeX 2.09 document styles, we distribute files called `article.sty`, `book.sty`, `report.sty`, `slides.sty` and `letter.sty`. These use the command `\@obsoletefile`, which produces a warning message.

```
\def\@obsoletefile#1#2{%
   \@latex@warning@no@line{inputting ‘#1’ instead of obsolete ‘#2’}}
\@onlypreamble\@obsoletefile
```

1.2 Listing files

`\@filelist` A list of files input so far. The initial value of `\@gobble` eats the comma before the first file name.

```
\let\@filelist\@gobble
```

`\@addtofilelist` Add to the list of files input so far. This ‘real’ definition is only used for ‘cfg’ files during initex. An initial definition of `\@gobble` has already been set.

```
%\def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}
```

`\if@listfiles@hashes` `\if@listfiles@sizes`

```
\ExplSyntaxOn
\keys_define:nn { __kernel / listfiles }
  {
    hashes .legacy_if_set:n = @listfiles@hashes ,
    sizes .legacy_if_set:n  = @listfiles@sizes
  }
\ExplSyntaxOff
```

`\listfiles` A preamble command to cause `\end{document}` to list files input from the main file.

```
\NewDocumentCommand\listfiles{O{}}{%
  \SetKeys[__kernel/listfiles]{#1}%
  \let\listfiles\relax
  \def\@listfiles##1##2##3##4##5##6##7##8##9\@@{%
     \def\reserved@d{\\}%
     \@tfor\reserved@c:=##1##2##3##4##5##6##7##8\do{%
       \ifx\reserved@c\reserved@d
         \edef\filename@area{ \filename@area}%
       \fi}}%

  \def\@dofilelist{%
     \typeout{^^J *File List*}%
     \@for\@currname:=\@filelist\do{%
       \filename@parse\@currname
       \edef\reserved@a{%
          \filename@base.%
```

```
          \ifx\filename@ext\relax tex\else\filename@ext\fi}%
        \expandafter\let\expandafter\reserved@b
                                \csname ver@\reserved@a\endcsname
```

Packages that `\relax` their `\ver@...` string to allow for multiple loading (e.g., fontenc) can use `\ver@@...` to store the version information instead.

```
        \ifx\reserved@b\relax
          \expandafter\let\expandafter\reserved@b
                            \csname ver@@\reserved@a\endcsname
        \fi
        \expandafter\expandafter\expandafter\@listfiles\expandafter
              \filename@area\filename@base\\\\\\\\\\\\\\\\\\\\@@
        \typeout{%
          \filename@area\reserved@a
          \ifx\reserved@b\relax\else\@spaces\reserved@b\fi
```

Now we add the additional information if requested.

```
          \ifnum0%
            \if@listfiles@hashes1\fi
            \if@listfiles@sizes1\fi
              >0 %
              ^^J\@spaces
              (%
                \if@listfiles@sizes
                  size \@dofilelist@size\@currname
                  \if@listfiles@hashes
                    , %
                  \fi
                \fi
                \if@listfiles@hashes
                  hash \@dofilelist@hash\@currname
                \fi
              )%
          \fi
      }}%
     \typeout{ ***********^^J}}}
```

```
\ExplSyntaxOn
\cs_new_eq:NN \@dofilelist@hash \file_mdfive_hash:n
\cs_new_eq:NN \@dofilelist@size \file_size:n
\ExplSyntaxOff
```

`\@dofilelist@hash`
`\@dofilelist@size`

The `\@filelist` will be de-activated if `\listfiles` does not appear in the preamble. `\begin{document}` contains code equivalent to the following:

```
\AtBeginDocument{%
  \ifx\@listfiles\@undefined
    \let\@filelist\relax
    \let\@addtofilelist\@gobble
  \fi}
```

```
\@onlypreamble\listfiles
```

\@dofilelist

```
\let\@dofilelist\relax
```

⟨/2ekernel⟩

(*End of definition for* `\@obsoletefile` *and others.*)

File 21
ltoutenc.dtx

1 Font encodings

This section of the kernel contains commands for declaring encoding-specific commands, such as accents. It also contains the code for some of the encoding files, including `omlenc.def`, `omsenc.def`, `t1enc.def` and `ot1enc.def` files, which define the `OML`, `OMS`, `T1` and `OT1` encodings, and the `fontenc` package for selecting encodings.

The `fontenc` package has options for encodings, of which the last option is the default encoding. For example, to use the `OT2`, `OT3` and `T1` encodings, with `T1` as the default, you say:

```
\usepackage[OT2,OT3,T1]{fontenc}
```

The standard kernel set-up loads font encoding files and selects an encoding as follows.

```
\input {omlenc.def}
\input  {t1enc.def}
\input {ot1enc.def}
\input {omsenc.def}
\fontencoding{OT1}
```

Note that the files in the standard `inputenc` package depend on this behaviour of the kernel.

The syntax for declaring encoding-specific commands is:

```
\DeclareTextCommand{⟨command⟩}{⟨encoding⟩}
                                [⟨number⟩][⟨default⟩]{⟨commands⟩}
```

This command is like `\newcommand`, except that it defines a command which is specific to one encoding. The resulting command is always robust, even if its definition is fragile. For example, the definition of `\l` in the `OT1` encoding is:

```
\DeclareTextCommand{\l}{OT1}{{\@xxxii l}}
```

`\DeclareTextCommand` takes the same optional arguments as `\newcommand`.

```
\ProvideTextCommand{⟨command⟩}{⟨encoding⟩}
                                [⟨number⟩][⟨default⟩]{⟨commands⟩}
```

This acts like `\DeclareTextCommand`, but does nothing if the command is already defined.

```
\DeclareTextSymbol{⟨command⟩}{⟨encoding⟩}{⟨slot⟩}
```

This command defines a text symbol, with a particular slot in that encoding. The commands:

```
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextCommand{\ss}{OT1}{\char25 }
```

have the same effect, but the `\DeclareTextSymbol` is faster.

```
\DeclareTextAccent{⟨command⟩}{⟨encoding⟩}{⟨slot⟩}
```

This command declares a text accent. The commands:

```
\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextCommand{\"}{OT1}{\add@accent {127}}
```

have the same effect.

\DeclareTextComposite{⟨*command*⟩}
{⟨*encoding*⟩}{⟨*argument*⟩}{⟨*slot*⟩}

This command declares a composite letter, for example in the T1 encoding \'{a} is slot 225, which is declared by:

```
\DeclareTextComposite{\'}{T1}{a}{225}
```

The *command* will normally have been declared with \DeclareTextAccent, or as a one-argument \DeclareTextCommand.

\DeclareTextComposite is the most common example of using the more general declaration \DeclareTextCompositeCommand, which can define a composite to be an arbitrary piece of text.

\DeclareTextCompositeCommand{⟨*command*⟩}
{⟨*encoding*⟩}{⟨*argument*⟩}{⟨*text*⟩}

For example, in the OT1 encoding Å has a hand-crafted definition this is declared as follows

```
\DeclareTextCompositeCommand{\r}{OT1}{A}
 {\leavevmode\setbox\z@\hbox{!}\dimen@\ht\z@\advance\dimen@-1ex%
  \rlap{\raise.67\dimen@\hbox{\char23}}A}
```

The *command* will normally have been declared with \DeclareTextAccent, or as a one-argument \DeclareTextCommand.

The commands defined using the above declarations can be used in two ways. Normally they are used by just calling the command in the appropriate encoding, for example \ss. However, sometimes you may wish to use a command in an encoding where it is not defined. If the command has no arguments, then you can use it in another encoding by calling \UseTextSymbol:

\UseTextSymbol{⟨*encoding*⟩}{⟨*command*⟩}

For example, \UseTextSymbol{OT1}{\ss} has the same effect as:

```
{\fontencoding{OT1}\selectfont\ss}
```

If the command has one argument then you can use it in another encoding by calling \UseTextAccent:

\UseTextAccent{⟨*encoding*⟩}{⟨*command*⟩}{⟨*text*⟩}

For example, if the current encoding is OT2 then \UseTextAccent{OT1}{\'}{a} has the same effect as:

```
{\fontencoding{OT1}\selectfont\'{\fontencoding{OT2}\selectfont a}}
```

You can also declare a default definition for a text command, which will be used if the current encoding has no appropriate definition. Such use will also set the definition for this command in the current encoding to equal this default definition; this makes subsequent uses of the command much faster.

```
\DeclareTextCommandDefault{⟨command⟩}{⟨definition⟩}
```

For example, the default definition of the command `\textonequarter` (which produces the fraction $\frac{1}{4}$) could be built using math mode:

```
\DeclareTextCommandDefault{\textonequarter}{\ensuremath {\frac14}}
```

There is a matching `\Provide` command which will not override an existing default definition:

```
\ProvideTextCommandDefault{⟨command⟩}{⟨definition⟩}
```

The most common use for these commands is to use symbols from other encodings, so there are some optimizations provided:

```
\DeclareTextSymbolDefault{⟨command⟩}{⟨encoding⟩}
\DeclareTextAccentDefault{⟨command⟩}{⟨encoding⟩}
```

are short for:

```
\DeclareTextCommandDefault{⟨command⟩}
                          {\UseTextSymbol{⟨encoding⟩}{⟨command⟩}}
\DeclareTextCommandDefault[1]{⟨command⟩}
                      {\UseTextAccent{⟨encoding⟩}{⟨command⟩}{#1}}
```

For example, to make `OT1` the default encoding for `\ss` and `\'` you say:

```
\DeclareTextSymbolDefault{\ss}{OT1}
\DeclareTextAccentDefault{\'}{OT1}
```

Note that you can use these commands on any zero- or one-argument commands declared with `\DeclareText*` or `\ProvideText*`, not just those defined using `\DeclareTextSymbol` or `\DeclareTextAccent`.

1.1 Removing encoding-specific commands

In some cases encoding definitions are given to provide some limited support since nothing better is available, for example, the definition for `\textdollar` in `OT1` is a hack since $ and £ actually share the same slot in this encoding. Thus if such a glyph becomes available in a different encoding (e.g., `TS1`) one would like to get rid of the flaky one and make the default definition point to the new encoding. In such a case defining

```
\DeclareTextSymbol{\textdollar}{TS1}{36}
\DeclareTextSymbolDefault{\textdollar}{TS1}
```

is not enough since if typesetting in `OT1` LaTeX will still find the encoding specific-definition for `OT1` and therefore ignore the new default. Therefore to ensure that in this case the `TS1` version is used we have to remove the `OT1` declaration:

```
\UndeclareTextCommand{\textdollar}{OT1}
```

Since the $ sign is a proper glyph in the T1 encoding there is no point removing its definition and forcing LaTeX to pick up the TS1 version if typesetting in this encoding. However, assume you want to use the variant dollar sign, i.e., $ for your dollars. In that case you have to get rid of the T1 declaration as well, e.g., the following would do that for you:

```
\UndeclareTextCommand{\textdollar}{OT1}
\UndeclareTextCommand{\textdollar} {T1}
\DeclareTextCommandDefault{\textdollar}
                          {\UseTextSymbol{TS1}\textdollaroldstyle}
```

1.2 The order of declarations

If an encoding-specific command is defined for more than one encoding, then it will execute fastest in the encoding in which it was defined last since its top-level definition will be set up to execute in that encoding without any overhead.

For this reason the file fonttext.ltx currently first loads the definitions for the T1 encoding and then those for the OT1 encoding so that typesetting in OT1 is optimized since that is (still) the default. However, when T1 is explicitly requested (via \usepackage[T1]{fontenc}) the top-level definitions are automatically changed to favour T1 since its declarations are reloaded in the process.

For the same reason default declarations should never come last since they are implemented as a special encoding themselves (with the name ?). Specifying them last would simply mean to make those encoding-specific commands equally inefficient in all encodings. Therefore the textcomp package, for example, first sets up all defaults to point to TS1 and then declares the commands in the TS1 encoding.

1.3 Docstrip modules

This .dtx file is be used to generate several related files containing font encoding definitions. The mutually exclusive docstrip options are listed here.

T1	generates t1enc.def for the Cork encoding.
TS1	generates ts1enc.def for the Text Companion encoding.
TS1sty	generates textcomp.sty, package that sets up use of the Text Companion encoding.
OT1	generates ot1enc.def for Knuth's CM encoding.
OMS	generates omsenc.def for Knuth's math symbol encoding.
OML	generates omlenc.def for Knuth's math letters encoding.
OT4	generates ot4enc.def for the Polish extension to the OT1 encoding, created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete.
TU	generates tuenc.def for Unicode font encoding.
package	generates fontenc.sty for selecting encodings.
2ekernel	for the kernel commands.

1.4 Definitions for the kernel

1.4.1 Declaration commands

This section contains definitions for commands such as accents which depend on the current encoding. These commands will usually be kept in `.def` files, for example `ot1enc.def` contains the definitions for the `OT1` encoding.

```
⟨*2ekernel⟩
\message{font encodings,}
```

Far too many macros in one block here!

\DeclareTextCommand
\ProvideTextCommand
\DeclareTextSymbol
\@dec@text@cmd
\chardef@text@cmd
\@changed@cmd
\@changed@x
\TextSymbolUnavailable
\@inmathwarn

If you say:

```
\DeclareTextCommand{\foo}{T1}...
```

then `\foo` is defined to be `\T1-cmd \foo \T1\foo`, where `\T1\foo` is *one* control sequence, not two! We then call `\newcommand` to define `\T1\foo`.

```
\def\DeclareTextCommand{%
   \@dec@text@cmd\newcommand}
\def\ProvideTextCommand{%
   \@dec@text@cmd\providecommand}
\def\@dec@text@cmd#1#2#3{%
   \expandafter\def\expandafter#2%
      \expandafter{%
         \csname#3-cmd\expandafter\endcsname
         \expandafter#2%
         \csname#3\string#2\endcsname
      }%
   \let\@ifdefinable\@rc@ifdefinable
   \expandafter#1\csname#3\string#2\endcsname}
```

This command was introduced to fix a major bug in `\@dec@text@cmd` without changing that command itself. This was thought to be necessary because it is defined in more than one package. (Perhaps the more serious bug is to put complex low-level commands like this in packages?)

The problem it solves is that whereas both `\newcommand` and `\providecommand` (used just above) both handle the resetting of `\@ifdefinable` (following its disabling in `\@dec@text@cmd`), the primitive `\chardef` neither needs the disabling, nor does the resetting.

```
\def\chardef@text@cmd{%
   \let\@ifdefinable\@@ifdefinable
   \chardef
  }
\def\DeclareTextSymbol#1#2#3{%
   \@dec@text@cmd\chardef@text@cmd#1{#2}#3\relax
  }
```

The declarations are only available before `\begin{document}`.

```
\@onlypreamble\DeclareTextCommand
\@onlypreamble\DeclareTextSymbol
```

The sneaky bit in all this is what `\T1-cmd \foo \T1\foo` does. There are five possibilities, depending on the current values of `\protect`, `\cf@encoding` and `\ifmmode`:

- If \protect is \@typeset@protect and \cf@encoding is T1, then we execute \T1\foo. This should be the normal behaviour, and is optimized for speed.
- If \protect is \@typeset@protect, \cf@encoding is (say) OT1, and \OT1\foo is defined, then we execute \OT1\foo.
- If \protect is \@typeset@protect, \cf@encoding is (say) OT1, we're in text mode, and \OT1\foo is undefined, then we define \OT1\foo to be the default value of \foo, and execute \OT1\foo.
- If \protect is \@typeset@protect, \cf@encoding is (say) OT1, we're in math mode, and \OT1\foo is undefined, then we execute the default value of \foo. (This is necessary so that things like $X_\copyright$ work properly.)
- If \protect is not \@typeset@protect then we execute \noexpand\foo. For example, if we are writing to a file, then this results in \foo being written. If we are in a \mark, then \foo will be put in the mark—since \foo is robust, it will then survive all the things which may happen to it whilst it's a \mark.

So after all that, we will either execute the appropriate definition of \foo for the current encoding, or we will execute \noexpand\foo.

The default value of \foo is \?\foo if it is defined, and an error message otherwise.

When the encoding is changed from T1 to OT1, \T1-cmd is defined to be \@changed@cmd and \OT1-cmd is defined to be \@current@cmd. This means that the test for what the current encoding is can be performed quickly.

```
\def\@current@cmd#1{%
   \ifx\protect\@typeset@protect
      \@inmathwarn#1%
   \else
      \noexpand#1\expandafter\@gobble
   \fi}
```

```
\def\@changed@cmd#1#2{%
   \ifx\protect\@typeset@protect
      \@inmathwarn#1%
      \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
         \expandafter\ifx\csname ?\string#1\endcsname\relax
            \expandafter\def\csname ?\string#1\endcsname{%
               \TextSymbolUnavailable#1%
            }%
         \fi
         \global\expandafter\let
               \csname\cf@encoding \string#1\expandafter\endcsname
               \csname ?\string#1\endcsname
      \fi
      \csname\cf@encoding\string#1%
         \expandafter\endcsname
   \else
      \noexpand#1%
   \fi}
```

```
\gdef\TextSymbolUnavailable#1{%
   \@latex@error{%
      Command \protect#1 unavailable in encoding \cf@encoding%
   }\@eha}
```

The command \@inmathwarn produces a warning message if we are currently in math mode. Note that since this command is used inside text commands, it can't call \relax before the \ifmmode. This means that it is possible for the warning to fail to be issued at the beginning of a row of an halign whose template enters math mode. This is probably a bad feature, but there's not much that can be done about it, since adding a \relax would break ligatures and kerning between text symbols.

A more efficient solution would be to make \@inmathwarn and \@inmatherr equal to \@empty and \relax by default, and to have \everymath reset them to their usual definitions. This is left for future investigation (for example it may break some third party code).

```
\def\@inmathwarn#1{%
   \ifmmode
      \@latex@warning{Command \protect#1 invalid in math mode}%
   \fi}
```

(*End of definition for* \DeclareTextCommand *and others.*)

\DeclareTextCommandDefault
\ProvideTextCommandDefault

These define commands with encoding ?.

Note that \DeclareTextCommandDefault can only be used in the preamble, but that the \Provide version is allowed in inputenc .def files, so is allowed anywhere.

```
\def\DeclareTextCommandDefault#1{%
   \DeclareTextCommand#1?}
```

```
\def\ProvideTextCommandDefault#1{%
   \ProvideTextCommand#1?}
```

```
\@onlypreamble\DeclareTextCommandDefault
%\@onlypreamble\ProvideTextCommandDefault
```

They require \?-cmd to be initialized as \@changed@cmd.

```
\expandafter\let\csname?-cmd\endcsname\@changed@cmd
```

(*End of definition for* \DeclareTextCommandDefault *and* \ProvideTextCommandDefault.)

\DeclareTextAccent

This is just a disguise for defining a TEX \accent command.

```
\def\DeclareTextAccent#1#2#3{%
   \DeclareTextCommand#1{#2}{\add@accent{#3}}}
```

```
\@onlypreamble\DeclareTextAccent
```

(*End of definition for* \DeclareTextAccent.)

\add@accent

To save space this code is shared between all text accents that are set using the \accent primitive. The argument is pre-set in a box so that any font loading that is needed is already done within the box. This is needed because font-loading involves grouping and that would prevent the accent mechanism from working so that the accent would not be positioned over the argument. Declarations that change the font should be allowed (only low-level ones are at present) inside the argument of an accent command, but not size changes, as they involve \setbox operations which also inhibit the mechanism of the \accent primitive.

Note that the whole process is within a group. For a detailed discussion of this reimplementation and its deficiencies, see pr/3160.

```
\def\add@accent#1#2{\hmode@bgroup
```

Turn off the group in \UseTextSymbol in case this is used inside the argument of \add@accent.

```
\let\hmode@start@before@group\@firstofone
\setbox\@tempboxa\hbox{#2%
```

When presetting the argument in a box we record its \spacefactor for later use after the accent got typeset. This way something like \`A gets the spacefactor of A (i.e., 999) rather than the default value of 1000.

```
    \global\mathchardef\accent@spacefactor\spacefactor}%
```

The accent primitive doesn't allow things \begingroup to interfere between accent and base character. Therefore we need to avoid that (they are some hidden inside \maybe@load@fontshape). As we don't have to load the fontshape in this case (as that already happened in the box above, if necessary), we simply disable that part of the code temporarily. We also ignore \ignorespaces which has the same issue and may show up as part of \normalfont if that is used.

```
\let\maybe@load@fontshape\relax
\let\ignorespaces\relax
\accent#1 #2\egroup\ifmmode\else\spacefactor\accent@spacefactor\fi}
```

Default definition for \accent@spacefactor prevents a horrible death of the above macro inside an unprotected \edef.

```
\let\accent@spacefactor\relax
```

(*End of definition for* \add@accent.)

\hmode@bgroup

```
\def\hmode@bgroup{\leavevmode\bgroup}
```

(*End of definition for* \hmode@bgroup.)

\DeclareTextCompositeCommand
\DeclareTextComposite
\@text@composite
\@text@composite@x
\@strip@args

Another amusing game to play with \expandafter, \csname, and \string. When you say \DeclareTextCompositeCommand{\foo}{T1}{a}{bar}, we look to see if the expansion of \T1\foo begins with \@text@composite, and if it doesn't, we redefine \T1\foo to be:

```
#1 -> \@text@composite \T1\foo #1\@empty \@text@composite {...}
```

where ... is the previous definition of \T1\foo. Finally, we define \\T1\foo-a to expand to bar.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/04/15}{\DeclareTextCompositeCommand}
⟨latexrelease⟩                              {test for undeclared accent}%
⟨*2ekernel | latexrelease⟩
\def\DeclareTextCompositeCommand#1#2#3#4{%
  \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
  \ifx\reserved@a\relax
   \DeclareTextCommand#1{#2}{%
     \@latex@error{\string#1 undeclared in encoding #2}\@eha}%
   \@latex@info{Composite with undeclared \string#1 in encoding #2}%
   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
  \fi
  \expandafter\expandafter\expandafter\ifx
  \expandafter\@car\reserved@a\relax\relax\@nil \@text@composite \else
      \edef\reserved@b##1{%
```

```
        \def\expandafter\noexpand
           \csname#2\string#1\endcsname####1{%
           \noexpand\@text@composite
              \expandafter\noexpand\csname#2\string#1\endcsname
              ####1\noexpand\@empty\noexpand\@text@composite
              {##1}}}%
      \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
   \fi
   \expandafter\def\csname\expandafter\string\csname
      #2\endcsname\string#1-\string#3\@empty\endcsname{#4}%
  }
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\DeclareTextCompositeCommand}
⟨latexrelease⟩                              {test for undeclared accent}%
⟨latexrelease⟩\def\DeclareTextCompositeCommand#1#2#3#4{%
⟨latexrelease⟩  \expandafter\let\expandafter\reserved@a
⟨latexrelease⟩                       \csname#2\string#1\endcsname
⟨latexrelease⟩  \expandafter\expandafter\expandafter\ifx
⟨latexrelease⟩  \expandafter\@car\reserved@a\relax\relax\@nil
⟨latexrelease⟩                                \@text@composite \else
⟨latexrelease⟩      \edef\reserved@b##1{%
⟨latexrelease⟩         \def\expandafter\noexpand
⟨latexrelease⟩           \csname#2\string#1\endcsname####1{%
⟨latexrelease⟩           \noexpand\@text@composite
⟨latexrelease⟩             \expandafter\noexpand\csname#2\string#1\endcsname
⟨latexrelease⟩             ####1\noexpand\@empty\noexpand\@text@composite
⟨latexrelease⟩             {##1}}}%
⟨latexrelease⟩      \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
⟨latexrelease⟩   \fi
⟨latexrelease⟩   \expandafter\def\csname\expandafter\string\csname
⟨latexrelease⟩      #2\endcsname\string#1-\string#3\@empty\endcsname{#4}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
\@onlypreamble\DeclareTextCompositeCommand
```

This all works because:

```
\@text@composite \T1\foo A\@empty \@text@composite {...}
```

expands to `\\T1\foo-A` if `\\T1\foo-A` has been defined, and `{...}` otherwise.

Note that `\@text@composite` grabs the first token of the argument and puts just that in the csname. This is so that `\'{\textit{e}}` will work—it checks whether `\\T1\'-\textit` is defined (which presumably it isn't) and so expands to `{\accent 1 \textit{e}}`.

This trick won't always work, for example `\'{{\itshape e}}` will expand to (with spaces added for clarity):

```
\csname \string \T1\' - \string {\itshape e} \@empty \endcsname
```

which will die pretty horribly. Unfortunately there's not much can be done about this if we're going to use `\csname` lookups as a fast way of accessing composites.

This has an unfortunate 'misfeature' though, which is that in the T1 encoding, `\'{aa}` produces á. This is not the expected behaviour, and should perhaps be fixed if the fix doesn't affect performance too badly.

Finally, it's worth noting that the `\@empty` is used in `\@text@composite` so that accents will work even when the argument is empty. If you say `\'{}` then this looks up `\\T1\'-\@empty`, which ought to be `\relax`, and so all is well. If we didn't include the `\@empty`, then `\'{}` would expand to:

```
\csname \string \T1\' - \string \endcsname
```

so the `\endcsname` would be `\string`'ed and the whole of the rest of the document would be put inside the `\csname`. This would not be good.

```
\def\@text@composite#1#2#3\@text@composite{%
   \expandafter\@text@composite@x
      \csname\string#1-\string#2\endcsname}
```

Originally the `\@text@composite@x` macro had two arguments and if `#1` was not `\relax` it was executed, otherwise `#2` was executed. All this happened within the `\ifx` code so that neither `#1` nor `#2` could have picked up any additional arguments form the input stream. This has now being changed using the typical `\@firstoftwo` / `\@secondoftwo` coding. This way the final expansion will happen without any `\else` or `\fi` intervening in the case that we need to get a further token from the input stream.

```
\def\@text@composite@x#1{%
   \ifx#1\relax
      \expandafter\@secondoftwo
   \else
      \expandafter\@firstoftwo
   \fi
   #1}
```

The command `\DeclareTextComposite` uses `\DeclareTextCompositeCommand` to declare a command which expands out to a single glyph.

```
\catcode\z@=11\relax
\def\DeclareTextComposite#1#2#3#4{%
   \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
   \bgroup
      \lccode\z@#4%
      \lowercase{%
   \egroup
      \reserved@a ^^@}}
\catcode\z@=15\relax
\@onlypreamble\DeclareTextComposite
```

(*End of definition for* `\DeclareTextCompositeCommand` *and others.*)

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                    {\UseTextAccent}{Make commands robust}%
```

`\UseTextAccent`
`\UseTextSymbol`
`\@use@text@encoding`

These fragile commands access glyphs from different encodings. They use grotty low-level calls to the font selection scheme for speed, and in order to make sure that `\UseTextSymbol` doesn't do anything which you're not allowed to do between an `\accent` and its glyph.

For a detailed discussion of this reimplementation and its deficiencies, see pr/3160.

```
\DeclareRobustCommand*\UseTextAccent[3]{%
  \hmode@start@before@group
   {%
```

Turn off the group in \UseTextSymbol in case this is used inside the arguments of \UseTextAccent.

```
    \let\hmode@start@before@group\@firstofone
    \let\@curr@enc\cf@encoding
    \@use@text@encoding{#1}%
    #2{\@use@text@encoding\@curr@enc#3}%
   }}
```

```
\DeclareRobustCommand*\UseTextSymbol[2]{%
       \hmode@start@before@group
       {%
          \def\@wrong@font@char{\MessageBreak
             for \noexpand\symbol`\string#2'}%
          \@use@text@encoding{#1}%
          #2%
       }%
    }
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\UseTextAccent}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\UseTextAccent
⟨latexrelease⟩\kernel@make@fragile\UseTextSymbol
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

Switch to a different text encoding without any grouping for use in \UseTextAccent or \UseTextSymbol (and for \oldstylenums).

```
\def\@use@text@encoding#1{%
   \edef\f@encoding{#1}%
   \xdef\font@name{%
      \csname\curr@fontshape/\f@size\endcsname}%
   \pickup@font
   \font@name
   \@@enc@update}
```

(*End of definition for* \UseTextAccent*,* \UseTextSymbol*, and* \@use@text@encoding*.*)

\hmode@start@before@group

The \hmode@start@before@group starts hmode and should be immediately followed by an explicit {...}. Its purpose is to ensure that hmode is started before this group is opened. Inside \add@accent and \UseTextAccent it is redefined to remove this group so that it doesn't conflict with the \accent primitive.

For a detailed discussion see pr/3160.

```
\let\hmode@start@before@group\leavevmode
```

(*End of definition for* \hmode@start@before@group*.*)

\DeclareTextSymbolDefault
\DeclareTextAccentDefault

Some syntactic sugar. Again, these should probably be optimized for speed.

```
\def\DeclareTextSymbolDefault#1#2{%
   \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}}
\def\DeclareTextAccentDefault#1#2{%
   \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}}
\@onlypreamble\DeclareTextSymbolDefault
\@onlypreamble\DeclareTextAccentDefault
```

(*End of definition for* `\DeclareTextSymbolDefault` *and* `\DeclareTextAccentDefault`.)

\UndeclareTextCommand

This command safely removes an encoding specific declaration for a given encoding. It is helpful if one intends to use the default definition always and therefore wants to get rid of a declaration for some specific encoding.

```
\def\UndeclareTextCommand#1#2{%
```

If there is no declaration for the current encoding do nothing. (This makes a hash table entry but without eTEX we can't do anything about that).

```
  \expandafter\ifx\csname#2\string#1\endcsname\relax
  \else
```

Else: throw away that declaration.

```
     \global\expandafter\let\csname#2\string#1\endcsname
            \@undefined
```

But this is unfortunately not enough, we have to take a look at the top-level definition of the encoding specific command which for a command `\foo` would look similar to `\T1-cmd \foo \T1\foo` (three tokens).

Of course, instead of `T1` one could see a different encoding name; which one depends the encoding for which `\foo` was declared last.

Now assume we have just removed the declaration for `\foo` in `T1` and the top-level of `\foo` expands to the above. Then we better change that pretty fast otherwise we do get an "undefined csname error" when we try to typeset `\foo` within `T1` instead of getting the default definition for `\foo`. And what is the best way to change that top-level definition? Well, the only "encoding" we know for sure will still be around is the default encoding denoted by `?`.

Thus in case the last token of the top-level expansion is now undefined we change the declaration to look like `\?-cmd \foo \?\foo` which is done by the following (readable?) code:

```
     \expandafter\expandafter\expandafter
     \ifx\expandafter\@thirdofthree#1\@undefined
       \expandafter\gdef\expandafter#1\expandafter
          {\csname ?-cmd\expandafter\endcsname\expandafter
           #1\csname?\string#1\endcsname}%
     \fi
 \fi
}
\@onlypreamble\UndeclareTextCommand
```

(*End of definition for* `\UndeclareTextCommand`.)

1.4.2 Hyphenation

\patterns
\@@patterns
\hyphenation
\@@hyphenation

We redefine \patterns and \hyphenation to allow the use of commands declared with \DeclareText* to be used inside them.

```
%\let\@@patterns\patterns
%\let\@@hyphenation\hyphenation
%\def\patterns{%
%   \bgroup
%      \let\protect\@empty
%      \let\@typeset@protect\@empty
%      \let\@changed@x\@changed@x@mouth
%   \afterassignment\egroup
%   \@@patterns
%}
%\def\hyphenation{%
%   \bgroup
%      \let\protect\@empty
%      \let\@typeset@protect\@empty
%      \let\@changed@x\@changed@x@mouth
%   \afterassignment\egroup
%   \@@hyphenation
%}
```

(*End of definition for* \patterns *and others.*)

1.4.3 Miscellania

\a

The \a command is used to access the accent commands even when they have been redefined (for example by the tabbing environment). Its internal name is \@tabacckludge.

The \string within the \csname guards against something like ' being active at the point of use.

```
\def\@tabacckludge#1{\expandafter\@changed@cmd
                                  \csname\string#1\endcsname\relax}
\let\a=\@tabacckludge
```

(*End of definition for* \a.)

1.4.4 Default encodings

We define the default encodings for most commands to be either OT1, OML or OMS. These defaults are in the kernel and therefore fonts with these encodings must be available unless these defaults are redefined elsewhere. Recall that the standard kernel loads the encoding files for these encodings, and also that for the T1 encoding.

The naming conventions in the kernel are not what we would use if we were starting from scratch... Those defined by DEK (like \ae and \ss) or by the TEX Users Group Technical Working Group on multi-lingual typesetting (like \th and \ng) have short names. Those which were added to the kernel in 1993 and early 1994 are named after their Adobe glyph names (like \guillemotleft and \quotedblbase). Unfortunately, this naming scheme won't work for all glyphs, since some names (like \space) are already used, and some (like \endash) are very likely to be defined by users. So we're now using the naming scheme of \text followed by the Adobe name, (like \textendash and \textsterling). Except that some glyphs don't have Adobe names, so we're using the names used by fontinst for those (like \textcompwordmark). Sigh.

Some accents from OT1:

```
\DeclareTextAccentDefault{\"}{OT1}
\DeclareTextAccentDefault{\'}{OT1}
\DeclareTextAccentDefault{\.}{OT1}
\DeclareTextAccentDefault{\=}{OT1}
\DeclareTextAccentDefault{\H}{OT1}
\DeclareTextAccentDefault{\^}{OT1}
\DeclareTextAccentDefault{\`}{OT1}
\DeclareTextAccentDefault{\b}{OT1}
\DeclareTextAccentDefault{\c}{OT1}
\DeclareTextAccentDefault{\d}{OT1}
\DeclareTextAccentDefault{\r}{OT1}
\DeclareTextAccentDefault{\u}{OT1}
\DeclareTextAccentDefault{\v}{OT1}
\DeclareTextAccentDefault{\~}{OT1}
```

Some symbols from OT1:

```
%\DeclareTextSymbolDefault{\AA}{OT1}
\DeclareTextSymbolDefault{\AE}{OT1}
\DeclareTextSymbolDefault{\L}{OT1}
\DeclareTextSymbolDefault{\OE}{OT1}
\DeclareTextSymbolDefault{\O}{OT1}
%\DeclareTextSymbolDefault{\aa}{OT1}
\DeclareTextSymbolDefault{\ae}{OT1}
\DeclareTextSymbolDefault{\i}{OT1}
\DeclareTextSymbolDefault{\j}{OT1}
```

```
\DeclareTextSymbolDefault{\ij}{OT1}
\DeclareTextSymbolDefault{\IJ}{OT1}
```

```
\DeclareTextSymbolDefault{\l}{OT1}
\DeclareTextSymbolDefault{\oe}{OT1}
\DeclareTextSymbolDefault{\o}{OT1}
\DeclareTextSymbolDefault{\ss}{OT1}
\DeclareTextSymbolDefault{\textdollar}{OT1}
\DeclareTextSymbolDefault{\textemdash}{OT1}
\DeclareTextSymbolDefault{\textendash}{OT1}
\DeclareTextSymbolDefault{\textexclamdown}{OT1}
%\DeclareTextSymbolDefault{\texthyphenchar}{OT1}
%\DeclareTextSymbolDefault{\texthyphen}{OT1}
\DeclareTextSymbolDefault{\textquestiondown}{OT1}
\DeclareTextSymbolDefault{\textquotedblleft}{OT1}
\DeclareTextSymbolDefault{\textquotedblright}{OT1}
\DeclareTextSymbolDefault{\textquoteleft}{OT1}
\DeclareTextSymbolDefault{\textquoteright}{OT1}
\DeclareTextSymbolDefault{\textsterling}{OT1}
```

Some symbols from OMS:

```
\DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
\DeclareTextSymbolDefault{\textbackslash}{OMS}
\DeclareTextSymbolDefault{\textbar}{OMS}
\DeclareTextSymbolDefault{\textbardbl}{OMS}
\DeclareTextSymbolDefault{\textbraceleft}{OMS}
\DeclareTextSymbolDefault{\textbraceright}{OMS}
\DeclareTextSymbolDefault{\textbullet}{OMS}
```

```
\DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
\DeclareTextSymbolDefault{\textdagger}{OMS}
\DeclareTextSymbolDefault{\textparagraph}{OMS}
\DeclareTextSymbolDefault{\textperiodcentered}{OMS}
\DeclareTextSymbolDefault{\textsection}{OMS}
\DeclareTextAccentDefault{\textcircled}{OMS}
```

Some symbols from OML:

```
\DeclareTextSymbolDefault{\textless}{OML}
\DeclareTextSymbolDefault{\textgreater}{OML}
\DeclareTextAccentDefault{\t}{OML}
```

Some defaults we can fake.

The interface for defining `\copyright` changed, it used to use `\expandafter` to add braces at the appropriate points.

```
\DeclareTextCommandDefault{\textcopyright}{\textcircled{c}}
% \expandafter\def\expandafter
%                 \copyright\expandafter{\expandafter{\copyright}}
```

```
\DeclareTextCommandDefault{\textasciicircum}{\^{}}
\DeclareTextCommandDefault{\textasciitilde}{\~{}}
\DeclareTextCommandDefault{\textunderscore}{%
  \leavevmode \kern.06em\vbox{\hrule\@width.3em}}
```

There is no good reason anymore to fake `\textcompwordmark`.

```
%\DeclareTextCommandDefault{\textcompwordmark}{\leavevmode\kern\z@}
\DeclareTextSymbolDefault{\textcompwordmark}{T1}
```

```
\DeclareTextCommandDefault{\textvisiblespace}{%
   \mbox{\kern.06em\vrule \@height.3ex}%
   \vbox{\hrule \@width.3em}%
   \hbox{\vrule \@height.3ex}}
```

Using `\fontdimen3` in the next definition is some sort of a kludge (since it is the interword stretch) but it makes the ellipsis come out right in mono-spaced fonts too (since there it is zero).

```
\DeclareTextCommandDefault{\textellipsis}{%
   .\kern\fontdimen3\font
   .\kern\fontdimen3\font
   .\kern\fontdimen3\font}
```

```
%\DeclareTextCommandDefault{\textregistered}{\textcircled{\scshape r}}
\DeclareTextCommandDefault{\textregistered}{\textcircled{%
     \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont R}}
\DeclareTextCommandDefault{\texttrademark}{\textsuperscript{TM}}
\DeclareTextCommandDefault{\SS}{SS}
```

```
\DeclareTextCommandDefault{\textordfeminine}{\textsuperscript{a}}
\DeclareTextCommandDefault{\textordmasculine}{\textsuperscript{o}}
```

1.4.5 Math material

Some commands can be used in both text and math mode:

```
\DeclareRobustCommand{\$}{\ifmmode\mathdollar\else\textdollar\fi}
```

We use `\protected` not `\DeclareRobustCommand` so that `\bigl\{` etc. works inside `\protected@edef`.

```
\protected\def\{{\ifmmode\lbrace\else\textbraceleft\fi}
\protected\def\}{\ifmmode\rbrace\else\textbraceright\fi}
```

```
\DeclareRobustCommand{\P}{\ifmmode\mathparagraph\else\textparagraph\fi}
\DeclareRobustCommand{\S}{\ifmmode\mathsection\else\textsection\fi}
\DeclareRobustCommand{\dag}{\ifmmode{\dagger}\else\textdagger\fi}
\DeclareRobustCommand{\ddag}{\ifmmode{\ddagger}\else\textdaggerdbl\fi}
```

For historical reasons `\copyright` needs `{}` around the definition in maths.

```
\DeclareRobustCommand{\_}{%
   \ifmmode\nfss@text{\textunderscore}\else\textunderscore\fi}
\DeclareRobustCommand{\copyright}{%
   \ifmmode{\nfss@text{\textcopyright}}\else\textcopyright\fi}
\DeclareRobustCommand{\pounds}{%
   \ifmmode\mathsterling\else\textsterling\fi}
```

```
\DeclareRobustCommand{\dots}{%
   \ifmmode\mathellipsis\else\textellipsis\fi}
```

```
\let\ldots\dots
```

Default definition of the commabelow accent.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/10/01}{\textcommabelow}{comma accent}%
⟨*2ekernel | latexrelease⟩
\DeclareTextCommandDefault\textcommabelow[1]
  {\hmode@bgroup\ooalign{\null#1\crcr\hidewidth\raise-.31ex
   \hbox{\check@mathfonts\fontsize\ssf@size\z@
   \math@fontsfalse\selectfont,}\hidewidth}\egroup}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\textcommabelow}{comma accent}%
⟨latexrelease⟩\let\textcommabelow\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-G\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-K\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-k\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-L\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-l\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-N\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-n\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩  \let\csname\string\T1\string\c-R\endcsname\@undefined
```

```
⟨latexrelease⟩\expandafter
⟨latexrelease⟩   \let\csname\string\T1\string\c-r\endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

Default definition of the commaabove accent(E.G.).

```
⟨latexrelease⟩\IncludeInRelease{2016/02/01}{\textcommaabove}{comma above}%
⟨*2ekernel | latexrelease⟩
\DeclareTextCommandDefault\textcommaabove[1]{%
  \hmode@bgroup
  \ooalign{%
    \hidewidth
    \raise.7ex\hbox{%
      \check@mathfonts\fontsize\ssf@size\z@\math@fontsfalse\selectfont‘%
    }%
   \hidewidth\crcr
   \null#1\crcr
  }%
  \egroup
}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\textcommaabove}{comma above}%
⟨latexrelease⟩\let\textcommaabove\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩   \let\csname\string\OT1\string\c-g\endcsname\@undefined
⟨latexrelease⟩\expandafter
⟨latexrelease⟩   \let\csname\string\T1\string\c-g\endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

1.5 Definitions for the OT1 encoding

The definitions for the ‘TEX text’ (OT1) encoding.

Declare the encoding.

```
⟨*OT1⟩
\DeclareFontEncoding{OT1}{}{}
```

Declare the accents.

```
\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextAccent{\’}{OT1}{19}
\DeclareTextAccent{\.}{OT1}{95}
\DeclareTextAccent{\=}{OT1}{22}
\DeclareTextAccent{\^}{OT1}{94}
\DeclareTextAccent{\‘}{OT1}{18}
\DeclareTextAccent{\~}{OT1}{126}
\DeclareTextAccent{\H}{OT1}{125}
\DeclareTextAccent{\u}{OT1}{21}
\DeclareTextAccent{\v}{OT1}{20}
\DeclareTextAccent{\r}{OT1}{23}
```

Some accents have to be built by hand: Note that `\ooalign` and `\o@lign` must be inside a group. In these definitions we no longer use the helper function `\sh@ft` from plain.tex since that now has two incompatible definitions.

```
\DeclareTextCommand{\b}{OT1}[1]
   {\hmode@bgroup\o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}%
```

```
     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
\DeclareTextCommand{\c}{OT1}[1]
   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
    \else{\ooalign{\unhbox\z@\crcr\hidewidth\char24\hidewidth}}\fi}
\DeclareTextCommand{\d}{OT1}[1]
   {\hmode@bgroup
    \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
```

Declare the text symbols.

```
\DeclareTextSymbol{\AE}{OT1}{29}
\DeclareTextSymbol{\OE}{OT1}{30}
\DeclareTextSymbol{\O}{OT1}{31}
\DeclareTextSymbol{\ae}{OT1}{26}
\DeclareTextSymbol{\i}{OT1}{16}
\DeclareTextSymbol{\j}{OT1}{17}
\DeclareTextSymbol{\oe}{OT1}{27}
\DeclareTextSymbol{\o}{OT1}{28}
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextSymbol{\textemdash}{OT1}{124}
\DeclareTextSymbol{\textendash}{OT1}{123}
```

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```
\DeclareTextCommand{\textnonbreakinghyphen}{OT1}{\mbox{-}\nobreak\hskip\z@}
\DeclareTextCommand{\textfiguredash}       {OT1}{\textendash}
\DeclareTextCommand{\texthorizontalbar}    {OT1}{\textemdash}
```

Using the ligatures helps with OT1 fonts that have `\textexclamdown` and `\textquestiondown` in unusual positions.

```
%\DeclareTextSymbol{\textexclamdown}{OT1}{60}
%\DeclareTextSymbol{\textquestiondown}{OT1}{62}
\DeclareTextCommand{\textexclamdown}{OT1}{!`}
\DeclareTextCommand{\textquestiondown}{OT1}{?`}
%\DeclareTextSymbol{\texthyphenchar}{OT1}{`\-}
%\DeclareTextSymbol{\texthyphen}{OT1}{`\-}
\DeclareTextSymbol{\textquotedblleft}{OT1}{92}
\DeclareTextSymbol{\textquotedblright}{OT1}{`\"}
\DeclareTextSymbol{\textquoteleft}{OT1}{`\`}
\DeclareTextSymbol{\textquoteright}{OT1}{`\'}
```

Some symbols which are faked from others:

```
% \DeclareTextCommand{\aa}{OT1}
%    {{\accent23a}}
\DeclareTextCommand{\L}{OT1}
   {\leavevmode\setbox\z@\hbox{L}\hb@xt@\wd\z@{\hss\@xxxii L}}
\DeclareTextCommand{\l}{OT1}
   {\hmode@bgroup\@xxxii l\egroup}
% \DeclareTextCommand{\AA}{OT1}
%    {\leavevmode\setbox\z@\hbox{h}\dimen@\ht\z@\advance\dimen@-1ex%
%     \rlap{\raise.67\dimen@\hbox{\char23}}A}
```

In the OT1 encoding Å has a hand-crafted definition, so we have here the first recorded explicit use of `\DeclareTextCompositeCommand`.

```
\DeclareTextCompositeCommand{\r}{OT1}{A}
   {\leavevmode\setbox\z@\hbox{!}\dimen@\ht\z@\advance\dimen@-1ex%
    \rlap{\raise.67\dimen@\hbox{\char23}}A}
```

The dutch language uses the letter 'ij'. It is available in T1 encoded fonts, but not in the OT1 encoded fonts. Therefore we fake it for the OT1 encoding.

```
\DeclareTextCommand{\ij}{OT1}{%
  \nobreak\hskip\z@skip i\kern-0.02em\nobreak\hskip\z@skip j}
\DeclareTextCommand{\IJ}{OT1}{%
  \nobreak\hskip\z@skip I\kern-0.02em\nobreak\hskip\z@skip J}
```

In the OT1 encoding, £ and $ share a slot.

```
\DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
   \ifdim \fontdimen\@ne\font >\z@
      \slshape
   \else
      \upshape
   \fi
   \char`\$\egroup}
```

```
\DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
   \ifdim \fontdimen\@ne\font >\z@
      \itshape
   \else
      \fontshape{ui}\selectfont
   \fi
   \char`\$\egroup}
```

Here we are adding some more composite commands to the OT1 encoding. This makes the use of certain accents with i compatible with their use with the T1 encoding; this enables them to become true LaTeX internal representations. However, it will make these accents work a little less fast since a check will always be made for the existence of a composite.

```
\DeclareTextComposite{\.}{OT1}{i}{`\i}
\DeclareTextComposite{\.}{OT1}{\i}{`\i}
\DeclareTextCompositeCommand{\`}{OT1}{i}{\@tabacckludge`\i}
\DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge'\i}
\DeclareTextCompositeCommand{\^}{OT1}{i}{\^\i}
\DeclareTextCompositeCommand{\"}{OT1}{i}{\"\i}
```

T1 encoding is given more extensive set of overloads for \c But here we just adjust \c{g}.

```
\ifx\textcommaabove\@undefined\else
\DeclareTextCompositeCommand{\c}{OT1}{g}{\textcommaabove{g}}
\fi
```

⟨/OT1⟩

1.6 Definitions for the T1 encoding

The definitions for the 'Extended TeX text' (T1) encoding.

Declare the encoding.

```
⟨*T1⟩
\DeclareFontEncoding{T1}{}{}
```

Declare the accents.

```
\DeclareTextAccent{\`}{T1}{0}
\DeclareTextAccent{\'}{T1}{1}
\DeclareTextAccent{\^}{T1}{2}
```

```
\DeclareTextAccent{\~}{T1}{3}
\DeclareTextAccent{\"}{T1}{4}
\DeclareTextAccent{\H}{T1}{5}
\DeclareTextAccent{\r}{T1}{6}
\DeclareTextAccent{\v}{T1}{7}
\DeclareTextAccent{\u}{T1}{8}
\DeclareTextAccent{\=}{T1}{9}
\DeclareTextAccent{\.}{T1}{10}
```

Some accents have to be built by hand. Note that `\ooalign` and `\o@lign` must be inside a group. In these definitions we no longer use the helper function `\sh@ft` from plain.tex since that now has two incompatible definitions.

```
\DeclareTextCommand{\b}{T1}[1]
   {\hmode@bgroup\o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}%
     \vbox to.2ex{\hbox{\char9}\vss}\hidewidth}\egroup}
\DeclareTextCommand{\c}{T1}[1]
   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
     \else{\ooalign{\unhbox\z@\crcr
        \hidewidth\char11\hidewidth}}\fi}
\DeclareTextCommand{\d}{T1}[1]
   {\hmode@bgroup
    \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
\DeclareTextCommand{\k}{T1}[1]
   {\hmode@bgroup\ooalign{\null#1\crcr\hidewidth\char12}\egroup}
\DeclareTextCommand{\textogonekcentered}{T1}[1]
   {\hmode@bgroup\ooalign{%
                 \null#1\crcr\hidewidth\char12\hidewidth}\egroup}
```

Some symbols are constructed.

Slot 24 contains a small circle intended for construction of these two glyphs.

```
\DeclareTextCommand{\textperthousand}{T1}
   {\%\char 24 }          % space or ‘relax as delimiter?
\DeclareTextCommand{\textpertenthousand}{T1}
   {\%\char 24\char 24 }  % space or ‘relax as delimiter?
```

For Maltese, `\Hwithstroke` and `\hwithstroke` are needed.

```
\DeclareTextCommand{\Hwithstroke}{T1}
   {%
    \hmode@bgroup
    \vphantom{H}%
    \sbox\z@{H}%
    \ooalign{%
      H\cr
      \hidewidth
      \vrule
        height \dimexpr 0.7\ht\z@+0.1ex\relax
        depth  -0.7\ht\z@
        width  0.8\wd\z@
      \hidewidth\cr
    }%
    \egroup
   }
\DeclareTextCommand{\hwithstroke}{T1}
   {%
```

```
    \hmode@bgroup
    \vphantom{h}%
    \sbox\z@{h}%
    \ooalign{%
      h\cr
      \kern0.075\wd\z@
      \vrule
        height \dimexpr 0.7\ht\z@+0.1ex\relax
        depth  -0.7\ht\z@
        width  0.4\wd\z@
      \hidewidth\cr
    }%
    \egroup
  }
```

Declare the text symbols.

```
%\DeclareTextSymbol{\AA}{T1}{197}
\DeclareTextSymbol{\AE}{T1}{198}
\DeclareTextSymbol{\DH}{T1}{208}
\DeclareTextSymbol{\DJ}{T1}{208}
\DeclareTextSymbol{\L}{T1}{138}
\DeclareTextSymbol{\NG}{T1}{141}
\DeclareTextSymbol{\OE}{T1}{215}
\DeclareTextSymbol{\O}{T1}{216}
\DeclareTextSymbol{\SS}{T1}{223}
\DeclareTextSymbol{\TH}{T1}{222}
%\DeclareTextSymbol{\aa}{T1}{229}
\DeclareTextSymbol{\ae}{T1}{230}
\DeclareTextSymbol{\dh}{T1}{240}
\DeclareTextSymbol{\dj}{T1}{158}

\DeclareTextSymbol{\guillemetleft}{T1}{19}
\DeclareTextSymbol{\guillemetright}{T1}{20}
% old Adobe names
\DeclareTextSymbol{\guillemotleft}{T1}{19}
\DeclareTextSymbol{\guillemotright}{T1}{20}

\DeclareTextSymbol{\guilsinglleft}{T1}{14}
\DeclareTextSymbol{\guilsinglright}{T1}{15}
\DeclareTextSymbol{\i}{T1}{25}
\DeclareTextSymbol{\j}{T1}{26}
\DeclareTextSymbol{\ij}{T1}{188}
\DeclareTextSymbol{\IJ}{T1}{156}
\DeclareTextSymbol{\l}{T1}{170}
\DeclareTextSymbol{\ng}{T1}{173}
\DeclareTextSymbol{\oe}{T1}{247}
\DeclareTextSymbol{\o}{T1}{248}
\DeclareTextSymbol{\quotedblbase}{T1}{18}
\DeclareTextSymbol{\quotesinglbase}{T1}{13}
\DeclareTextSymbol{\ss}{T1}{255}
\DeclareTextSymbol{\textasciicircum}{T1}{`\^}
\DeclareTextSymbol{\textasciitilde}{T1}{`\~}
\DeclareTextSymbol{\textbackslash}{T1}{`\\}
\DeclareTextSymbol{\textbar}{T1}{`\|}
\DeclareTextSymbol{\textbraceleft}{T1}{`\{}
```

```
\DeclareTextSymbol{\textbraceright}{T1}{`\}}
\DeclareTextSymbol{\textcompwordmark}{T1}{23}
\DeclareTextSymbol{\textdollar}{T1}{`\$}
\DeclareTextSymbol{\textemdash}{T1}{22}
\DeclareTextSymbol{\textendash}{T1}{21}
```

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```
\DeclareTextCommand{\textnonbreakinghyphen}{T1}{\mbox{-}\nobreak\hskip\z@}
\DeclareTextCommand{\textfiguredash}       {T1}{\textendash}
\DeclareTextCommand{\texthorizontalbar}    {T1}{\textemdash}
```

```
\DeclareTextSymbol{\textexclamdown}{T1}{189}
\DeclareTextSymbol{\textgreater}{T1}{`\>}
%\DeclareTextSymbol{\texthyphenchar}{T1}{127}
%\DeclareTextSymbol{\texthyphen}{T1}{`\-}
\DeclareTextSymbol{\textless}{T1}{`\<}
\DeclareTextSymbol{\textquestiondown}{T1}{190}
\DeclareTextSymbol{\textquotedblleft}{T1}{16}
\DeclareTextSymbol{\textquotedblright}{T1}{17}
\DeclareTextSymbol{\textquotedbl}{T1}{`\"}
\DeclareTextSymbol{\textquoteleft}{T1}{`\`}
\DeclareTextSymbol{\textquoteright}{T1}{`\'}
\DeclareTextSymbol{\textsection}{T1}{159}
\DeclareTextSymbol{\textsterling}{T1}{191}
\DeclareTextSymbol{\textunderscore}{T1}{95}
\DeclareTextSymbol{\textvisiblespace}{T1}{32}
\DeclareTextSymbol{\th}{T1}{254}
```

Declare the composites.

```
\DeclareTextComposite{\.}{T1}{i}{`\i}
\DeclareTextComposite{\.}{T1}{\i}{`\i}
```

"80 = 128

```
\DeclareTextComposite{\u}{T1}{A}{128}
\DeclareTextComposite{\k}{T1}{A}{129}
\DeclareTextComposite{\'}{T1}{C}{130}
\DeclareTextComposite{\v}{T1}{C}{131}
\DeclareTextComposite{\v}{T1}{D}{132}
\DeclareTextComposite{\v}{T1}{E}{133}
\DeclareTextComposite{\k}{T1}{E}{134}
\DeclareTextComposite{\u}{T1}{G}{135}
```

"88 = 136

```
\DeclareTextComposite{\'}{T1}{L}{136}
\DeclareTextComposite{\v}{T1}{L}{137}
\DeclareTextComposite{\'}{T1}{N}{139}
\DeclareTextComposite{\v}{T1}{N}{140}
\DeclareTextComposite{\H}{T1}{O}{142}
\DeclareTextComposite{\'}{T1}{R}{143}
```

"90 = 144

```
\DeclareTextComposite{\v}{T1}{R}{144}
\DeclareTextComposite{\'}{T1}{S}{145}
\DeclareTextComposite{\v}{T1}{S}{146}
\DeclareTextComposite{\c}{T1}{S}{147}
```

```
\DeclareTextComposite{\v}{T1}{T}{148}
\DeclareTextComposite{\c}{T1}{T}{149}
\DeclareTextComposite{\H}{T1}{U}{150}
\DeclareTextComposite{\r}{T1}{U}{151}
```

"98 = 152

```
\DeclareTextComposite{\"}{T1}{Y}{152}
\DeclareTextComposite{\'}{T1}{Z}{153}
\DeclareTextComposite{\v}{T1}{Z}{154}
\DeclareTextComposite{\.}{T1}{Z}{155}
\DeclareTextComposite{\.}{T1}{I}{157}
```

"A0 = 160

```
\DeclareTextComposite{\u}{T1}{a}{160}
\DeclareTextComposite{\k}{T1}{a}{161}
\DeclareTextComposite{\'}{T1}{c}{162}
\DeclareTextComposite{\v}{T1}{c}{163}
\DeclareTextComposite{\v}{T1}{d}{164}
\DeclareTextComposite{\v}{T1}{e}{165}
\DeclareTextComposite{\k}{T1}{e}{166}
\DeclareTextComposite{\u}{T1}{g}{167}
```

"A8 = 168

```
\DeclareTextComposite{\'}{T1}{l}{168}
\DeclareTextComposite{\v}{T1}{l}{169}
\DeclareTextComposite{\'}{T1}{n}{171}
\DeclareTextComposite{\v}{T1}{n}{172}
\DeclareTextComposite{\H}{T1}{o}{174}
\DeclareTextComposite{\'}{T1}{r}{175}
```

"B0 = 176

```
\DeclareTextComposite{\v}{T1}{r}{176}
\DeclareTextComposite{\'}{T1}{s}{177}
\DeclareTextComposite{\v}{T1}{s}{178}
\DeclareTextComposite{\c}{T1}{s}{179}
\DeclareTextComposite{\v}{T1}{t}{180}
\DeclareTextComposite{\c}{T1}{t}{181}
\DeclareTextComposite{\H}{T1}{u}{182}
\DeclareTextComposite{\r}{T1}{u}{183}
```

"B8 = 184

```
\DeclareTextComposite{\"}{T1}{y}{184}
\DeclareTextComposite{\'}{T1}{z}{185}
\DeclareTextComposite{\v}{T1}{z}{186}
\DeclareTextComposite{\.}{T1}{z}{187}
```

"C0 = 192

```
\DeclareTextComposite{\`}{T1}{A}{192}
\DeclareTextComposite{\'}{T1}{A}{193}
\DeclareTextComposite{\^}{T1}{A}{194}
\DeclareTextComposite{\~}{T1}{A}{195}
\DeclareTextComposite{\"}{T1}{A}{196}
\DeclareTextComposite{\r}{T1}{A}{197}
\DeclareTextComposite{\c}{T1}{C}{199}
```

"C8 = 200

```
\DeclareTextComposite{\`}{T1}{E}{200}
\DeclareTextComposite{\'}{T1}{E}{201}
\DeclareTextComposite{\^}{T1}{E}{202}
\DeclareTextComposite{\"}{T1}{E}{203}
\DeclareTextComposite{\`}{T1}{I}{204}
\DeclareTextComposite{\'}{T1}{I}{205}
\DeclareTextComposite{\^}{T1}{I}{206}
\DeclareTextComposite{\"}{T1}{I}{207}
```

"D0 = 208

```
\DeclareTextComposite{\~}{T1}{N}{209}
\DeclareTextComposite{\`}{T1}{O}{210}
\DeclareTextComposite{\'}{T1}{O}{211}
\DeclareTextComposite{\^}{T1}{O}{212}
\DeclareTextComposite{\~}{T1}{O}{213}
\DeclareTextComposite{\"}{T1}{O}{214}
```

"D8 = 216

```
\DeclareTextComposite{\`}{T1}{U}{217}
\DeclareTextComposite{\'}{T1}{U}{218}
\DeclareTextComposite{\^}{T1}{U}{219}
\DeclareTextComposite{\"}{T1}{U}{220}
\DeclareTextComposite{\'}{T1}{Y}{221}
```

"E0 = 224

```
\DeclareTextComposite{\`}{T1}{a}{224}
\DeclareTextComposite{\'}{T1}{a}{225}
\DeclareTextComposite{\^}{T1}{a}{226}
\DeclareTextComposite{\~}{T1}{a}{227}
\DeclareTextComposite{\"}{T1}{a}{228}
\DeclareTextComposite{\r}{T1}{a}{229}
\DeclareTextComposite{\c}{T1}{c}{231}
```

"E8 = 232

```
\DeclareTextComposite{\`}{T1}{e}{232}
\DeclareTextComposite{\'}{T1}{e}{233}
\DeclareTextComposite{\^}{T1}{e}{234}
\DeclareTextComposite{\"}{T1}{e}{235}
\DeclareTextComposite{\`}{T1}{i}{236}
\DeclareTextComposite{\`}{T1}{\i}{236}
\DeclareTextComposite{\'}{T1}{i}{237}
\DeclareTextComposite{\'}{T1}{\i}{237}
\DeclareTextComposite{\^}{T1}{i}{238}
\DeclareTextComposite{\^}{T1}{\i}{238}
\DeclareTextComposite{\"}{T1}{i}{239}
\DeclareTextComposite{\"}{T1}{\i}{239}
```

"F0 = 240

```
\DeclareTextComposite{\~}{T1}{n}{241}
\DeclareTextComposite{\`}{T1}{o}{242}
\DeclareTextComposite{\'}{T1}{o}{243}
\DeclareTextComposite{\^}{T1}{o}{244}
\DeclareTextComposite{\~}{T1}{o}{245}
\DeclareTextComposite{\"}{T1}{o}{246}
```

"F8 = 248

```
\DeclareTextComposite{\`}{T1}{u}{249}
\DeclareTextComposite{\'}{T1}{u}{250}
\DeclareTextComposite{\^}{T1}{u}{251}
\DeclareTextComposite{\"}{T1}{u}{252}
\DeclareTextComposite{\'}{T1}{y}{253}
```

```
\DeclareTextCompositeCommand{\k}{T1}{o}{\textogonekcentered{o}}
\DeclareTextCompositeCommand{\k}{T1}{O}{\textogonekcentered{O}}
```

```
\ifx\textcommaabove\@undefined\else
\DeclareTextCompositeCommand{\c}{T1}{g}{\textcommaabove{g}}
\fi
\ifx\textcommabelow\@undefined\else
\DeclareTextCompositeCommand{\c}{T1}{G}{\textcommabelow{G}}
\DeclareTextCompositeCommand{\c}{T1}{K}{\textcommabelow{K}}
\DeclareTextCompositeCommand{\c}{T1}{k}{\textcommabelow{k}}
\DeclareTextCompositeCommand{\c}{T1}{L}{\textcommabelow{L}}
\DeclareTextCompositeCommand{\c}{T1}{l}{\textcommabelow{l}}
\DeclareTextCompositeCommand{\c}{T1}{N}{\textcommabelow{N}}
\DeclareTextCompositeCommand{\c}{T1}{n}{\textcommabelow{n}}
\DeclareTextCompositeCommand{\c}{T1}{R}{\textcommabelow{R}}
\DeclareTextCompositeCommand{\c}{T1}{r}{\textcommabelow{r}}
\fi
```

⟨/T1⟩

1.7 Definitions for the OMS encoding

The definitions for the 'TEX math symbol' (OMS) encoding. Even though this is meant to be a math font, it includes some of the standard LATEX text symbols.

Declare the encoding.

⟨*OMS⟩

```
\DeclareFontEncoding{OMS}{}{}
```

Declare the symbols. Note that slot 13 has in places been named `\Orb`: please root out and destroy this impolity wherever you find it!

```
\DeclareTextSymbol{\textasteriskcentered}{OMS}{3}   % "03
\DeclareTextSymbol{\textbackslash}{OMS}{110}        % "6E
\DeclareTextSymbol{\textbar}{OMS}{106}              % "6A
\DeclareTextSymbol{\textbardbl}{OMS}{107}           % "6B
\DeclareTextSymbol{\textbraceleft}{OMS}{102}        % "66
\DeclareTextSymbol{\textbraceright}{OMS}{103}       % "67
\DeclareTextSymbol{\textbullet}{OMS}{15}            % "0F
\DeclareTextSymbol{\textdaggerdbl}{OMS}{122}        % "7A
\DeclareTextSymbol{\textdagger}{OMS}{121}           % "79
\DeclareTextSymbol{\textparagraph}{OMS}{123}        % "7B
\DeclareTextSymbol{\textperiodcentered}{OMS}{1}     % "01
\DeclareTextSymbol{\textsection}{OMS}{120}          % "78
\DeclareTextSymbol{\textbigcircle}{OMS}{13}         % "0D
\DeclareTextCommand{\textcircled}{OMS}[1]{\hmode@bgroup
   \ooalign{%
      \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
      \char 13 % "0D
```

```
  }%
 \egroup}
⟨/OMS⟩
```

1.8 Definitions for the OML encoding

The definitions for the 'TEX math italic' (OML) encoding. Even though this is meant to be a math font, it includes some of the standard LATEX text symbols.

Declare the encoding.

```
⟨*OML⟩
\DeclareFontEncoding{OML}{}{}
```

Declare the symbols.

```
\DeclareTextSymbol{\textless}{OML}{`\<}
\DeclareTextSymbol{\textgreater}{OML}{`\>}
\DeclareTextAccent{\t}{OML}{127}  % "7F
⟨/OML⟩
```

1.9 Definitions for the OT4 encoding

These definitions are for the Polish extension to the 'TEX text' (OT1) encoding. This encoding was created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete. In positions 0–127 it is identical to OT1 but it contains some additional characters in the upper half. The LATEX support was developed by Mariusz Olko.

The PL fonts that use it are available as follows:
Metafont sources `ftp://ftp.gust.org.pl/TeX/language/polish/pl-mf.zip`;
Font files `ftp://ftp.gust.org.pl/TeX/language/polish/pl-tfm.zip`.

Declare the encoding.

```
⟨*OT4⟩
\DeclareFontEncoding{OT4}{}{}
\DeclareFontSubstitution{OT4}{cmr}{m}{n}
```

Declare the accents.

```
\DeclareTextAccent{\"}{OT4}{127}
\DeclareTextAccent{\'}{OT4}{19}
\DeclareTextAccent{\.}{OT4}{95}
\DeclareTextAccent{\=}{OT4}{22}
\DeclareTextAccent{\^}{OT4}{94}
\DeclareTextAccent{\`}{OT4}{18}
\DeclareTextAccent{\~}{OT4}{126}
\DeclareTextAccent{\H}{OT4}{125}
\DeclareTextAccent{\u}{OT4}{21}
\DeclareTextAccent{\v}{OT4}{20}
\DeclareTextAccent{\r}{OT4}{23}
```

The ogonek accent is available only under a e A & E. But we have to provide some definition for `\k`. Some other accents have to be built by hand as in OT1:

```
\DeclareTextCommand{\k}{OT4}[1]{%
    \TextSymbolUnavailable{\k{#1}}#1}
```

In these definitions we no longer use the helper function `\sh@ft` from plain.tex since that now has two incompatible definitions.

```
\DeclareTextCommand{\b}{OT4}[1]
   {\hmode@bgroup\o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}%
     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
\DeclareTextCommand{\c}{OT4}[1]
   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
    \else{\ooalign{\unhbox\z@\crcr\hidewidth\char24\hidewidth}}\fi}
\DeclareTextCommand{\d}{OT4}[1]
   {\hmode@bgroup
    \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
```

Declare the text symbols.

```
\DeclareTextSymbol{\AE}{OT4}{29}
\DeclareTextSymbol{\OE}{OT4}{30}
\DeclareTextSymbol{\O}{OT4}{31}
\DeclareTextSymbol{\L}{OT4}{138}
\DeclareTextSymbol{\ae}{OT4}{26}

\DeclareTextSymbol{\guillemetleft}{OT4}{174}
\DeclareTextSymbol{\guillemetright}{OT4}{175}
% old Adobe names
\DeclareTextSymbol{\guillemotleft}{OT4}{174}
\DeclareTextSymbol{\guillemotright}{OT4}{175}

\DeclareTextSymbol{\i}{OT4}{16}
\DeclareTextSymbol{\j}{OT4}{17}
\DeclareTextSymbol{\l}{OT4}{170}
\DeclareTextSymbol{\o}{OT4}{28}
\DeclareTextSymbol{\oe}{OT4}{27}
\DeclareTextSymbol{\quotedblbase}{OT4}{255}
\DeclareTextSymbol{\ss}{OT4}{25}
\DeclareTextSymbol{\textemdash}{OT4}{124}
\DeclareTextSymbol{\textendash}{OT4}{123}
\DeclareTextSymbol{\textexclamdown}{OT4}{60}
%\DeclareTextSymbol{\texthyphenchar}{OT4}{`\-}
%\DeclareTextSymbol{\texthyphen}{OT4}{`\-}
\DeclareTextSymbol{\textquestiondown}{OT4}{62}
\DeclareTextSymbol{\textquotedblleft}{OT4}{92}
\DeclareTextSymbol{\textquotedblright}{OT4}{`\"}
\DeclareTextSymbol{\textquoteleft}{OT4}{`\`}
\DeclareTextSymbol{\textquoteright}{OT4}{`\'}
```

Definition for Å as in OT1:

```
\DeclareTextCompositeCommand{\r}{OT4}{A}
   {\leavevmode\setbox\z@\hbox{!}\dimen@\ht\z@\advance\dimen@-1ex%
    \rlap{\raise.67\dimen@\hbox{\char23}}A}
```

In the OT4 encoding, £ and $ share a slot.

```
\DeclareTextCommand{\textdollar}{OT4}{\hmode@bgroup
   \ifdim \fontdimen\@ne\font >\z@
      \slshape
   \else
      \upshape
   \fi
   \char`\$\egroup}
```

```
\DeclareTextCommand{\textsterling}{OT4}{\hmode@bgroup
   \ifdim \fontdimen\@ne\font >\z@
      \itshape
   \else
      \fontshape{ui}\selectfont
   \fi
   \char`\$\egroup}
```

Declare the composites.

```
\DeclareTextComposite{\k}{OT4}{A}{129}
\DeclareTextComposite{\'}{OT4}{C}{130}
\DeclareTextComposite{\k}{OT4}{E}{134}
\DeclareTextComposite{\'}{OT4}{N}{139}
\DeclareTextComposite{\'}{OT4}{S}{145}
\DeclareTextComposite{\'}{OT4}{Z}{153}
\DeclareTextComposite{\.}{OT4}{Z}{155}
\DeclareTextComposite{\k}{OT4}{a}{161}
\DeclareTextComposite{\'}{OT4}{c}{162}
\DeclareTextComposite{\k}{OT4}{e}{166}
\DeclareTextComposite{\'}{OT4}{n}{171}
\DeclareTextComposite{\'}{OT4}{s}{177}
\DeclareTextComposite{\'}{OT4}{z}{185}
\DeclareTextComposite{\.}{OT4}{z}{187}
\DeclareTextComposite{\'}{OT4}{O}{211}
\DeclareTextComposite{\'}{OT4}{o}{243}
⟨/OT4⟩
```

1.10 Definitions for the TS1 encoding

```
⟨*TS1⟩
\DeclareFontEncoding{TS1}{}{}
\DeclareFontSubstitution{TS1}{cmr}{m}{n}
```

Some accents have to be built by hand. Note that `\ooalign` and `\o@lign` must be inside a group.

```
\DeclareTextCommand{\capitalcedilla}{TS1}[1]
   {\hmode@bgroup
    \ooalign{\null#1\crcr\hidewidth\char11\hidewidth}\egroup}
\DeclareTextCommand{\capitalogonek}{TS1}[1]
   {\hmode@bgroup
    \ooalign{\null#1\crcr\hidewidth\char12\hidewidth}\egroup}
```

Accents for capital letters.

These commands can be used by the end user either directly or through definitions of the type
`\DeclareTextCompositeCommand{\'}{T1}{X}{\capitalacute X}`

None of the latter definitions are provided by default, since they are probably rarely used.

"00 = 0

```
\DeclareTextAccent{\capitalgrave}{TS1}{0}
\DeclareTextAccent{\capitalacute}{TS1}{1}
\DeclareTextAccent{\capitalcircumflex}{TS1}{2}
\DeclareTextAccent{\capitaltilde}{TS1}{3}
\DeclareTextAccent{\capitaldieresis}{TS1}{4}
\DeclareTextAccent{\capitalhungarumlaut}{TS1}{5}
```

```
\DeclareTextAccent{\capitalring}{TS1}{6}
\DeclareTextAccent{\capitalcaron}{TS1}{7}
```

"08 = 8

```
\DeclareTextAccent{\capitalbreve}{TS1}{8}
\DeclareTextAccent{\capitalmacron}{TS1}{9}
\DeclareTextAccent{\capitaldotaccent}{TS1}{10}
```

Tie accents.

The tie accent was borrowed from the `cmmi` font. The tc fonts now provide four tie accents, the first two are done in the classical way with asymmetric glyphs hanging out of their boxes; the new ties are centered in their boxes like all other accents. They need a name: please tell us if you know what to call them.

" =

```
\DeclareTextAccent{\t}{TS1}{26}
\DeclareTextAccent{\capitaltie}{TS1}{27}
\DeclareTextAccent{\newtie}{TS1}{28}
\DeclareTextAccent{\capitalnewtie}{TS1}{29}
```

Compound word marks.

The text companion fonts contain two compound word marks of different heights, one has `cap_height`, the other `asc_height`.

```
\DeclareTextSymbol{\textcapitalcompwordmark}{TS1}{23}
\DeclareTextSymbol{\textascendercompwordmark}{TS1}{31}
```

The text companion symbols.

```
\DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
```

"10 = 16

```
\DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
\DeclareTextSymbol{\texttwelveudash}{TS1}{21}
\DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}
```

"18 = 24

```
\DeclareTextSymbol{\textleftarrow}{TS1}{24}
\DeclareTextSymbol{\textrightarrow}{TS1}{25}
```

"20 = 32

```
\DeclareTextSymbol{\textblank}{TS1}{32}
\DeclareTextSymbol{\textdollar}{TS1}{36}
\DeclareTextSymbol{\textquotesingle}{TS1}{39}
```

"28 = 40

The symbol `\textasteriskcentered` "$*$" is supposed to be always available in `TS1` and that is important as it is used in footnote symbols. However, in a few fonts it is missing even though they are otherwise fairly complete. We therefore use a rather elaborate method and check if the slot has a glyph and if not produce a poor man's version by using a normal "*" slightly enlarged and somewhat lowered. The main application for this symbol is in footnote symbols and there it should produce a comparable size and show a similar placement.

```
%\DeclareTextSymbol{\textasteriskcentered}{TS1}{42} % that's wanted
\DeclareTextCommand \textasteriskcentered{TS1}{%    % and that's needed
  \iffontchar\font 42 \char42 \else
   \begingroup\fontencoding{T1}%
       \fontsize
```

```
        {\the\dimexpr1.3\dimexpr\f@size pt\relax}%
        {\f@baselineskip}%
      \selectfont
      \raisebox{-0.7ex}[\dimexpr\height-0.7ex][0pt]{*}%
    \endgroup
    \fi
}
```

Note that '054 is a comma and '056 is a full stop: these make numbers using oldstyle digits easier to input.

```
\DeclareTextSymbol{\textdblhyphen}{TS1}{45}
\DeclareTextSymbol{\textfractionsolidus}{TS1}{47}
```

Oldstyle digits.

"30 = 48

```
\DeclareTextSymbol{\textzerooldstyle}{TS1}{48}
\DeclareTextSymbol{\textoneoldstyle}{TS1}{49}
\DeclareTextSymbol{\texttwooldstyle}{TS1}{50}
\DeclareTextSymbol{\textthreeoldstyle}{TS1}{51}
\DeclareTextSymbol{\textfouroldstyle}{TS1}{52}
\DeclareTextSymbol{\textfiveoldstyle}{TS1}{53}
\DeclareTextSymbol{\textsixoldstyle}{TS1}{54}
\DeclareTextSymbol{\textsevenoldstyle}{TS1}{55}
```

"38 = 56

```
\DeclareTextSymbol{\texteightoldstyle}{TS1}{56}
\DeclareTextSymbol{\textnineoldstyle}{TS1}{57}
```

More text companion symbols.

```
\DeclareTextSymbol{\textlangle}{TS1}{60}
\DeclareTextSymbol{\textminus}{TS1}{61}
\DeclareTextSymbol{\textrangle}{TS1}{62}
```

"48 = 72

```
\DeclareTextSymbol{\textmho}{TS1}{77}
```

The big circle is here to define the command `\textcircled`. Formerly it was taken from the `cmsy` font.

```
\DeclareTextSymbol{\textbigcircle}{TS1}{79}
\DeclareTextCommand{\textcircled}{TS1}[1]{\hmode@bgroup
   \ooalign{%
      \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
      \char 79   % '117 = "4F
   }%
 \egroup}
```

More text companion symbols.

"50 = 80

```
\DeclareTextSymbol{\textohm}{TS1}{87}
```

"58 = 88

```
\DeclareTextSymbol{\textlbrackdbl}{TS1}{91}
\DeclareTextSymbol{\textrbrackdbl}{TS1}{93}
\DeclareTextSymbol{\textuparrow}{TS1}{94}
\DeclareTextSymbol{\textdownarrow}{TS1}{95}
```

"60 = 96

```
\DeclareTextSymbol{\textasciigrave}{TS1}{96}
\DeclareTextSymbol{\textborn}{TS1}{98}
\DeclareTextSymbol{\textdivorced}{TS1}{99}
\DeclareTextSymbol{\textdied}{TS1}{100}
```

"68 = 104

```
\DeclareTextSymbol{\textleaf}{TS1}{108}
\DeclareTextSymbol{\textmarried}{TS1}{109}
\DeclareTextSymbol{\textmusicalnote}{TS1}{110}
```

"78 = 120

```
\DeclareTextSymbol{\texttildelow}{TS1}{126}
```

This glyph, `\textdblhyphenchar` is hanging, like the hyphenchar of the ec fonts.

```
\DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
```

"80 = 128

```
\DeclareTextSymbol{\textasciibreve}{TS1}{128}
\DeclareTextSymbol{\textasciicaron}{TS1}{129}
```

This next glyph is *not* the same as `\textquotedbl`.

```
\DeclareTextSymbol{\textacutedbl}{TS1}{130}
\DeclareTextSymbol{\textgravedbl}{TS1}{131}
\DeclareTextSymbol{\textdagger}{TS1}{132}
\DeclareTextSymbol{\textdaggerdbl}{TS1}{133}
\DeclareTextSymbol{\textbardbl}{TS1}{134}
\DeclareTextSymbol{\textperthousand}{TS1}{135}
```

"88 = 136

```
\DeclareTextSymbol{\textbullet}{TS1}{136}
\DeclareTextSymbol{\textcelsius}{TS1}{137}
\DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
\DeclareTextSymbol{\textcentoldstyle}{TS1}{139}
\DeclareTextSymbol{\textflorin}{TS1}{140}
\DeclareTextSymbol{\textcolonmonetary}{TS1}{141}
\DeclareTextSymbol{\textwon}{TS1}{142}
\DeclareTextSymbol{\textnaira}{TS1}{143}
```

"90 = 144

```
\DeclareTextSymbol{\textguarani}{TS1}{144}
\DeclareTextSymbol{\textpeso}{TS1}{145}
\DeclareTextSymbol{\textlira}{TS1}{146}
\DeclareTextSymbol{\textrecipe}{TS1}{147}
\DeclareTextSymbol{\textinterrobang}{TS1}{148}
\DeclareTextSymbol{\textinterrobangdown}{TS1}{149}
\DeclareTextSymbol{\textdong}{TS1}{150}
\DeclareTextSymbol{\texttrademark}{TS1}{151}
```

"98 = 152

```
\DeclareTextSymbol{\textpertenthousand}{TS1}{152}
\DeclareTextSymbol{\textpilcrow}{TS1}{153}
\DeclareTextSymbol{\textbaht}{TS1}{154}
\DeclareTextSymbol{\textnumero}{TS1}{155}
```

This next name may change. For the following sign we know only a german name, which is abzüglich. The meaning is something like "commercial minus". An ASCII ersatz is ./. (dot slash dot). The temporary English name is `\textdiscount`.

```
\DeclareTextSymbol{\textdiscount}{TS1}{156}
\DeclareTextSymbol{\textestimated}{TS1}{157}
\DeclareTextSymbol{\textopenbullet}{TS1}{158}
\DeclareTextSymbol{\textservicemark}{TS1}{159}
```

"A0 = 160

```
\DeclareTextSymbol{\textlquill}{TS1}{160}
\DeclareTextSymbol{\textrquill}{TS1}{161}
\DeclareTextSymbol{\textcent}{TS1}{162}
\DeclareTextSymbol{\textsterling}{TS1}{163}
\DeclareTextSymbol{\textcurrency}{TS1}{164}
\DeclareTextSymbol{\textyen}{TS1}{165}
\DeclareTextSymbol{\textbrokenbar}{TS1}{166}
\DeclareTextSymbol{\textsection}{TS1}{167}
```

"A8 = 168

```
\DeclareTextSymbol{\textasciidieresis}{TS1}{168}
\DeclareTextSymbol{\textcopyright}{TS1}{169}
\DeclareTextSymbol{\textordfeminine}{TS1}{170}
\DeclareTextSymbol{\textcopyleft}{TS1}{171}
\DeclareTextSymbol{\textlnot}{TS1}{172}
```

The meaning of the circled-P is "sound recording copyright".

```
\DeclareTextSymbol{\textcircledP}{TS1}{173}
\DeclareTextSymbol{\textregistered}{TS1}{174}
\DeclareTextSymbol{\textasciimacron}{TS1}{175}
```

"B0 = 176

```
\DeclareTextSymbol{\textdegree}{TS1}{176}
\DeclareTextSymbol{\textpm}{TS1}{177}
\DeclareTextSymbol{\texttwosuperior}{TS1}{178}
\DeclareTextSymbol{\textthreesuperior}{TS1}{179}
\DeclareTextSymbol{\textasciiacute}{TS1}{180}
\DeclareTextSymbol{\textmu}{TS1}{181} % micro sign
\DeclareTextSymbol{\textparagraph}{TS1}{182}
\DeclareTextSymbol{\textperiodcentered}{TS1}{183}
```

"B8 = 184

```
\DeclareTextSymbol{\textreferencemark}{TS1}{184}
\DeclareTextSymbol{\textonesuperior}{TS1}{185}
\DeclareTextSymbol{\textordmasculine}{TS1}{186}
\DeclareTextSymbol{\textsurd}{TS1}{187}
\DeclareTextSymbol{\textonequarter}{TS1}{188}
\DeclareTextSymbol{\textonehalf}{TS1}{189}
\DeclareTextSymbol{\textthreequarters}{TS1}{190}
\DeclareTextSymbol{\texteuro}{TS1}{191}
```

"E0 = 208

```
\DeclareTextSymbol{\texttimes}{TS1}{214}
```

"F0 = 240

```
\DeclareTextSymbol{\textdiv}{TS1}{246}
```

⟨/TS1⟩

1.11 Definitions for the TU encoding

The TU encoding was originally introduced in the contributed package `fontspec` as a Unicode encoding for XeTeX and LuaTeX.

Normally for these engines, the input consists of Unicode characters encoded in UTF-8. There is therefore little need to use the traditional (ASCII) encoding-specific commands

However, sometimes (e.g. for backwards compatibility) it can be useful to access these Unicode characters via such ASCII-based markup. The commands provided here Cover the characters in the T1 and TS1 encodings, but specified in Unicode position. Almost all the command names have been mechanically extracted form the `inputenc` UTF-8 support, which is essentially doing a reverse mapping from UTF-8 data to LaTeX LICR commands.

A few additional names for character which were supported in the original `fontspec` version of this file have also been added, even though they are not currently in the default `inputenc` UTF-8 declarations.

```
⟨*TU⟩
```

In the base interface the Unicode encoding is always known as TU But we parameterize the encoding name to allow for modelling differences in Unicode support by different fonts.

```
\providecommand\UnicodeEncodingName{TU}
```

As the Unicode encoding, TU, is only currently available with XeTeX or LuaTeX, we detect these engines first, and make adjustments for the differing font loading syntax. For other engines, we issue a warning then abort this file, switching back to T1 encoding.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname XeTeXrevision\endcsname\relax
 \begingroup\expandafter\expandafter\expandafter\endgroup
 \expandafter\ifx\csname directlua\endcsname\relax
```

Not LuaTeX or XeTeX, abort with a warning.

```
   \PackageWarningNoLine{fontenc}
     {\UnicodeEncodingName\space
      encoding is only available with XeTeX and LuaTeX.\MessageBreak
      Defaulting to T1 encoding}
     \def\encodingdefault{T1}
   \expandafter\expandafter\expandafter\endinput
 \else
```

LuaTeX. For LuaTeX 1.10+, define a Lua function to disable any handing by the font code. Otherwise we reload the font without TeX ligatures.

```
   \def\UnicodeFontTeXLigatures{+tlig;}
   \ifnum\luatexversion<110
     \def\reserved@a#1{%
       \def\@remove@tlig@##1{\@remove@tlig@##1\@nil#1\@nil\relax}
       \def\@remove@tlig@##1#1{\@remove@tlig@@##1}}
     \edef\reserved@b{\detokenize{+tlig;}}
     \expandafter\reserved@a\expandafter{\reserved@b}
     \def\@remove@tlig@@#1\@nil#2\relax{#1}
```

```
      \def\remove@tlig#1{%
        \begingroup
        \font\remove@tlig
        \expandafter\@remove@tlig\expandafter{\fontname\font}%
        \remove@tlig
        \char#1\relax
        \endgroup
      }
    \else
      \newprotectedluacmd\@remove@tlig@@@
```

Now we can define the function. Mostly we just have to insert a protected glyph node, which is a glyph node with subtype 256. But we have to keep track of the current mode to avoid inserting the glyph into a vlist.

```
      \now@and@everyjob{\directlua{
        local rawchar_func = token.create'@remove@tlig@@@'.index
        local forcehmode = tex.forcehmode
        local put_next = token.put_next
        local glyph_id = node.id'glyph'
        local rawchar_token = token.new(rawchar_func, token.command_id'lua_call')
        lua.get_functions_table()[rawchar_func] = function()
          local mode = tex.nest.top.mode
          if mode == 1 or mode == -1 then
            put_next(rawchar_token)
            return forcehmode(true)
          end
          local n = node.new(glyph_id, 256)
          n.font = font.current()
          n.char = token.scan_int()
          return node.write(n)
        end
      }}
```

Now `\remove@tlig` can be implemented almost as in XeTeX.

```
      \def\remove@tlig#1{\@remove@tlig@@@#1\relax}
    \fi
  \fi
\else
```

XeTeX

```
  \def\UnicodeFontTeXLigatures{mapping=tex-text;}
  \def\remove@tlig#1{\XeTeXglyph\numexpr\XeTeXcharglyph#1\relax}
\fi
\def\UnicodeFontFile#1#2{"[#1]:#2"}
\def\UnicodeFontName#1#2{"#1:#2"}
```

Declare the encoding

```
\DeclareFontEncoding\UnicodeEncodingName{}{}
```

Declare accent command to use a postpended combining character rather than the TeX `\accent` primitive

```
\def\add@unicode@accent#1#2{%
  \if\relax\detokenize{#2}\relax^^a0\else#2\fi
  \char#1\relax}
```

In its original implementation \DeclareUnicodeAccent was given 3 arguments (with second the "Unicode encoding" a.k.a., \UnicodeEncodingName) while in other places, e.g., \DeclareUnicodeComposite, we always made encoding implicit. So we now change it here to implicit too so that the interfaces become a bit more consistent. To avoid making that a breaking change (even though it only affects two packages on CTAN) we test for #2 being \UnicodeEncodingName. This would not catch if somebody used \DeclareUnicodeAccent{\=}{TU-sub}{"0304} but that fortunately hasn't happened. With the implicit argument you would need to change \UnicodeEncodingName instead, as you have to do anyway for the other interface commands.

```
\def\DeclareUnicodeAccent#1#2{%
  \edef\reserved@a{#2}%
  \edef\reserved@b{\UnicodeEncodingName}%
  \ifx\reserved@a\reserved@b
    \def\reserved@a{\DeclareUnicodeAccent@{#1}}%
  \else
    \def\reserved@a{\DeclareUnicodeAccent@{#1}\UnicodeEncodingName}%
  \fi
  \reserved@a{#2}%
}
\def\DeclareUnicodeAccent@#1#2#3{%
  \DeclareTextCommand{#1}{#2}{\add@unicode@accent{#3}}%
}
```

Wrapper around \DeclareTextCompositeCommand that uses the declared composite if it exists in the current font or falls back to the default definition for the TU accent if not.

```
{
\catcode\z@=11\relax
\gdef\DeclareUnicodeComposite#1#2#3{%
   \def\reserved@a##1##2{%
     \DeclareTextCompositeCommand#1\UnicodeEncodingName{#2}{%
   \iffontchar\font#3 ##2%
      \else ##1\fi}}%
    \expandafter\expandafter\expandafter\extract@default@composite
    \csname\UnicodeEncodingName\string#1\endcsname{#2}\@nil
   \bgroup
      \lccode\z@#3 %
      \lowercase{\egroup
      \expandafter\reserved@a\expandafter{\reserved@b}{^^@}}}%
}
```

```
\def\extract@default@composite#1{%
 \ifx\@text@composite#1%
   \expandafter\extract@default@composite@a
 \else
   \expandafter\extract@default@composite@b\expandafter#1%
 \fi}
```

```
\def\extract@default@composite@a#1\@text@composite#2\@nil{%
  \def\reserved@b{#2}}
\def\extract@default@composite@b#1#2\@nil{%
  \def\reserved@b{#1#2}}
```

Next two commands are simply syntactic sugar to go with the other \DeclareUnicode... declarations.

```
\def\DeclareUnicodeSymbol#1{\DeclareTextSymbol{#1}{\UnicodeEncodingName}}
\def\DeclareUnicodeCommand#1{\DeclareTextCommand{#1}{\UnicodeEncodingName}}

\DeclareUnicodeCommand\textquotesingle {\remove@tlig{"0027}}
\DeclareUnicodeCommand\textasciigrave  {\remove@tlig{"0060}}
\DeclareUnicodeCommand\textquotedbl    {\remove@tlig{"0022}}

\DeclareUnicodeSymbol{\textdollar}          {"0024}
\DeclareUnicodeSymbol{\textless}            {"003C}
\DeclareUnicodeSymbol{\textgreater}         {"003E}
\DeclareUnicodeSymbol{\textbackslash}       {"005C}
\DeclareUnicodeSymbol{\textasciicircum}     {"005E}
\DeclareUnicodeSymbol{\textunderscore}      {"005F}
\DeclareUnicodeSymbol{\textbraceleft}       {"007B}
\DeclareUnicodeSymbol{\textbar}             {"007C}
\DeclareUnicodeSymbol{\textbraceright}      {"007D}
\DeclareUnicodeSymbol{\textasciitilde}      {"007E}
\DeclareUnicodeSymbol{\textexclamdown}      {"00A1}
\DeclareUnicodeSymbol{\textcent}            {"00A2}
\DeclareUnicodeSymbol{\textsterling}        {"00A3}
\DeclareUnicodeSymbol{\textcurrency}        {"00A4}
\DeclareUnicodeSymbol{\textyen}             {"00A5}
\DeclareUnicodeSymbol{\textbrokenbar}       {"00A6}
\DeclareUnicodeSymbol{\textsection}         {"00A7}
\DeclareUnicodeSymbol{\textasciidieresis}   {"00A8}
\DeclareUnicodeSymbol{\textcopyright}       {"00A9}
\DeclareUnicodeSymbol{\textordfeminine}     {"00AA}

\DeclareUnicodeSymbol{\guillemetleft}       {"00AB}
% old Adobe name
\DeclareUnicodeSymbol{\guillemotleft}       {"00AB}

\DeclareUnicodeSymbol{\textlnot}            {"00AC}
\DeclareUnicodeSymbol{\textregistered}      {"00AE}
\DeclareUnicodeSymbol{\textasciimacron}     {"00AF}
\DeclareUnicodeSymbol{\textdegree}          {"00B0}
\DeclareUnicodeSymbol{\textpm}              {"00B1}
\DeclareUnicodeSymbol{\texttwosuperior}     {"00B2}
\DeclareUnicodeSymbol{\textthreesuperior}   {"00B3}
\DeclareUnicodeSymbol{\textasciiacute}      {"00B4}
\DeclareUnicodeSymbol{\textmu}              {"00B5}
\DeclareUnicodeSymbol{\textparagraph}       {"00B6}
\DeclareUnicodeSymbol{\textperiodcentered}  {"00B7}
\DeclareUnicodeSymbol{\textonesuperior}     {"00B9}
\DeclareUnicodeSymbol{\textordmasculine}    {"00BA}

\DeclareUnicodeSymbol{\guillemetright}      {"00BB}
% old Adobe name
\DeclareUnicodeSymbol{\guillemotright}      {"00BB}

\DeclareUnicodeSymbol{\textonequarter}      {"00BC}
\DeclareUnicodeSymbol{\textonehalf}         {"00BD}
\DeclareUnicodeSymbol{\textthreequarters}   {"00BE}
\DeclareUnicodeSymbol{\textquestiondown}    {"00BF}
\DeclareUnicodeSymbol{\AE}                  {"00C6}
\DeclareUnicodeSymbol{\DH}                  {"00D0}
\DeclareUnicodeSymbol{\texttimes}           {"00D7}
```

```
\DeclareUnicodeSymbol{\O}                    {"00D8}
\DeclareUnicodeSymbol{\TH}                   {"00DE}
\DeclareUnicodeSymbol{\ss}                   {"00DF}
\DeclareUnicodeSymbol{\ae}                   {"00E6}
\DeclareUnicodeSymbol{\dh}                   {"00F0}
\DeclareUnicodeSymbol{\textdiv}              {"00F7}
\DeclareUnicodeSymbol{\o}                    {"00F8}
\DeclareUnicodeSymbol{\th}                   {"00FE}
\DeclareUnicodeSymbol{\DJ}                   {"0110}
\DeclareUnicodeSymbol{\dj}                   {"0111}
\DeclareUnicodeSymbol{\i}                    {"0131}
\DeclareUnicodeSymbol{\IJ}                   {"0132}
\DeclareUnicodeSymbol{\ij}                   {"0133}
\DeclareUnicodeSymbol{\L}                    {"0141}
\DeclareUnicodeSymbol{\l}                    {"0142}
\DeclareUnicodeSymbol{\NG}                   {"014A}
\DeclareUnicodeSymbol{\ng}                   {"014B}
\DeclareUnicodeSymbol{\OE}                   {"0152}
\DeclareUnicodeSymbol{\oe}                   {"0153}
\DeclareUnicodeSymbol{\textflorin}           {"0192}
\DeclareUnicodeSymbol{\j}                    {"0237}
\DeclareUnicodeSymbol{\textasciicaron}       {"02C7}
\DeclareUnicodeSymbol{\textasciibreve}       {"02D8}
\DeclareUnicodeSymbol{\textacutedbl}         {"02DD}
\DeclareUnicodeSymbol{\textgravedbl}         {"02F5}
\DeclareUnicodeSymbol{\texttildelow}         {"02F7}
\DeclareUnicodeSymbol{\textbaht}             {"0E3F}
\DeclareUnicodeSymbol{\SS}                   {"1E9E}
\DeclareUnicodeSymbol{\textcompwordmark}     {"200C}
```

```
%\DeclareUnicodeSymbol{\textnonbreakinghyphen} {"2011}
%\DeclareUnicodeSymbol{\textfiguredash}        {"2012}
\DeclareUnicodeSymbol{\textendash}           {"2013}
\DeclareUnicodeSymbol{\textemdash}           {"2014}
%\DeclareUnicodeSymbol{\texthorizontalbar}   {"2015}
```

Unfortunately some fonts do not implement `"2011`, `"2012` and/or `"2015` (including the LaTeX default fonts for Unicode engines) so we provide some approximations if the glyph is missing, like we do for `OT1` and `T1`.

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```
\DeclareUnicodeCommand{\textnonbreakinghyphen}
     {\iffontchar\font "2011 \char "2011 \else \mbox{-}\nobreak\hskip\z@ \fi}
\DeclareUnicodeCommand{\textfiguredash}
     {\iffontchar\font "2012 \char "2012 \else \char "2013 \fi}
\DeclareUnicodeCommand{\texthorizontalbar}
     {\iffontchar\font "2015 \char "2015 \else \char "2014 \fi}
```

```
\DeclareUnicodeSymbol{\textbardbl}           {"2016}
\DeclareUnicodeSymbol{\textquoteleft}        {"2018}
\DeclareUnicodeSymbol{\textquoteright}       {"2019}
\DeclareUnicodeSymbol{\quotesinglbase}       {"201A}
\DeclareUnicodeSymbol{\textquotedblleft}     {"201C}
\DeclareUnicodeSymbol{\textquotedblright}    {"201D}
\DeclareUnicodeSymbol{\quotedblbase}         {"201E}
```

```
\DeclareUnicodeSymbol{\textdagger}          {"2020}
\DeclareUnicodeSymbol{\textdaggerdbl}       {"2021}
\DeclareUnicodeSymbol{\textbullet}          {"2022}
\DeclareUnicodeSymbol{\textellipsis}        {"2026}
\DeclareUnicodeSymbol{\textperthousand}     {"2030}
\DeclareUnicodeSymbol{\textpertenthousand}  {"2031}
\DeclareUnicodeSymbol{\guilsinglleft}       {"2039}
\DeclareUnicodeSymbol{\guilsinglright}      {"203A}
\DeclareUnicodeSymbol{\textreferencemark}   {"203B}
\DeclareUnicodeSymbol{\textinterrobang}     {"203D}
\DeclareUnicodeSymbol{\textfractionsolidus} {"2044}
\DeclareUnicodeSymbol{\textlquill}          {"2045}
\DeclareUnicodeSymbol{\textrquill}          {"2046}
\DeclareUnicodeSymbol{\textdiscount}        {"2052}
\DeclareUnicodeSymbol{\textcolonmonetary}   {"20A1}
\DeclareUnicodeSymbol{\textlira}            {"20A4}
\DeclareUnicodeSymbol{\textnaira}           {"20A6}
\DeclareUnicodeSymbol{\textwon}             {"20A9}
\DeclareUnicodeSymbol{\textdong}            {"20AB}
\DeclareUnicodeSymbol{\texteuro}            {"20AC}
\DeclareUnicodeSymbol{\textpeso}            {"20B1}
\DeclareUnicodeSymbol{\textcelsius}         {"2103}
\DeclareUnicodeSymbol{\textnumero}          {"2116}
\DeclareUnicodeSymbol{\textcircledP}        {"2117}
\DeclareUnicodeSymbol{\textrecipe}          {"211E}
\DeclareUnicodeSymbol{\textservicemark}     {"2120}
\DeclareUnicodeSymbol{\texttrademark}       {"2122}
\DeclareUnicodeSymbol{\textohm}             {"2126}
\DeclareUnicodeSymbol{\textmho}             {"2127}
\DeclareUnicodeSymbol{\textestimated}       {"212E}
\DeclareUnicodeSymbol{\textleftarrow}       {"2190}
\DeclareUnicodeSymbol{\textuparrow}         {"2191}
\DeclareUnicodeSymbol{\textrightarrow}      {"2192}
\DeclareUnicodeSymbol{\textdownarrow}       {"2193}
\DeclareUnicodeSymbol{\textminus}           {"2212}

\DeclareUnicodeSymbol{\Hwithstroke}         {"0126}
\DeclareUnicodeSymbol{\hwithstroke}         {"0127}
```

Not all fonts have U+2217 but using U+002A requires some adjustment.

```
\DeclareUnicodeCommand{\textasteriskcentered}{%
  \iffontchar\font"2217 \char"2217 \else
    \begingroup
      \fontsize
       {\the\dimexpr1.3\dimexpr\f@size pt\relax}%
       {\f@baselineskip}%
      \selectfont
      \raisebox{-0.7ex}[\dimexpr\height-0.7ex][0pt]{*}%
    \endgroup
  \fi
}
```

```
\DeclareUnicodeSymbol{\textsurd}            {"221A}
\DeclareUnicodeSymbol{\textlangle}          {"2329}
```

```
\DeclareUnicodeSymbol{\textrangle}          {"232A}
\DeclareUnicodeSymbol{\textblank}           {"2422}
\DeclareUnicodeSymbol{\textvisiblespace}    {"2423}
\DeclareUnicodeSymbol{\textopenbullet}      {"25E6}
\DeclareUnicodeSymbol{\textbigcircle}       {"25EF}
\DeclareUnicodeSymbol{\textmusicalnote}     {"266A}
\DeclareUnicodeSymbol{\textmarried}         {"26AD}
\DeclareUnicodeSymbol{\textdivorced}        {"26AE}
\DeclareUnicodeSymbol{\textinterrobangdown} {"2E18}
```

Accents must be declared before the composites that use them.

```
\DeclareUnicodeAccent{\`}{"0300}
\DeclareUnicodeAccent{\'}{"0301}
\DeclareUnicodeAccent{\^}{"0302}
\DeclareUnicodeAccent{\~}{"0303}
\DeclareUnicodeAccent{\=}{"0304}
\DeclareUnicodeAccent{\u}{"0306}
\DeclareUnicodeAccent{\.}{"0307}
\DeclareUnicodeAccent{\"}{"0308}
\DeclareUnicodeAccent{\r}{"030A}
\DeclareUnicodeAccent{\H}{"030B}
\DeclareUnicodeAccent{\v}{"030C}
\DeclareUnicodeAccent{\b}{"0332}
\DeclareUnicodeAccent{\d}{"0323}
\DeclareUnicodeAccent{\c}{"0327}
\DeclareUnicodeAccent{\k}{"0328}
```

The odd one out:

```
\DeclareUnicodeCommand\textcommabelow[1]
  {\hmode@bgroup\ooalign{\null#1\crcr\hidewidth\raise-.31ex
   \hbox{\check@mathfonts\fontsize\ssf@size\z@
   \math@fontsfalse\selectfont,}\hidewidth}\egroup}
```

```
\DeclareUnicodeComposite{\^}              {}{"005E}
\DeclareUnicodeComposite{\~}              {}{"007E}
```

```
\DeclareUnicodeComposite{\`}             {A}{"00C0}
\DeclareUnicodeComposite{\'}             {A}{"00C1}
\DeclareUnicodeComposite{\^}             {A}{"00C2}
\DeclareUnicodeComposite{\~}             {A}{"00C3}
\DeclareUnicodeComposite{\"}             {A}{"00C4}
\DeclareUnicodeComposite{\r}             {A}{"00C5}
\DeclareUnicodeComposite{\c}             {C}{"00C7}
\DeclareUnicodeComposite{\`}             {E}{"00C8}
\DeclareUnicodeComposite{\'}             {E}{"00C9}
\DeclareUnicodeComposite{\^}             {E}{"00CA}
\DeclareUnicodeComposite{\"}             {E}{"00CB}
\DeclareUnicodeComposite{\`}             {I}{"00CC}
\DeclareUnicodeComposite{\'}             {I}{"00CD}
\DeclareUnicodeComposite{\^}             {I}{"00CE}
\DeclareUnicodeComposite{\"}             {I}{"00CF}
\DeclareUnicodeComposite{\~}             {N}{"00D1}
\DeclareUnicodeComposite{\`}             {O}{"00D2}
\DeclareUnicodeComposite{\'}             {O}{"00D3}
\DeclareUnicodeComposite{\^}             {O}{"00D4}
\DeclareUnicodeComposite{\~}             {O}{"00D5}
```

```
\DeclareUnicodeComposite{\"}            {O}{"00D6}
\DeclareUnicodeComposite{\`}            {U}{"00D9}
\DeclareUnicodeComposite{\'}            {U}{"00DA}
\DeclareUnicodeComposite{\^}            {U}{"00DB}
\DeclareUnicodeComposite{\"}            {U}{"00DC}
\DeclareUnicodeComposite{\'}            {Y}{"00DD}
\DeclareUnicodeComposite{\`}            {a}{"00E0}
\DeclareUnicodeComposite{\'}            {a}{"00E1}
\DeclareUnicodeComposite{\^}            {a}{"00E2}
\DeclareUnicodeComposite{\~}            {a}{"00E3}
\DeclareUnicodeComposite{\"}            {a}{"00E4}
\DeclareUnicodeComposite{\r}            {a}{"00E5}
\DeclareUnicodeComposite{\c}            {c}{"00E7}
\DeclareUnicodeComposite{\`}            {e}{"00E8}
\DeclareUnicodeComposite{\'}            {e}{"00E9}
\DeclareUnicodeComposite{\^}            {e}{"00EA}
\DeclareUnicodeComposite{\"}            {e}{"00EB}
\DeclareUnicodeComposite{\`}            \i {"00EC}
\DeclareUnicodeComposite{\`}            {i}{"00EC}
\DeclareUnicodeComposite{\'}            \i {"00ED}
\DeclareUnicodeComposite{\'}            {i}{"00ED}
\DeclareUnicodeComposite{\^}            \i {"00EE}
\DeclareUnicodeComposite{\^}            {i}{"00EE}
\DeclareUnicodeComposite{\"}            \i {"00EF}
\DeclareUnicodeComposite{\"}            {i}{"00EF}
\DeclareUnicodeComposite{\~}            {n}{"00F1}
\DeclareUnicodeComposite{\`}            {o}{"00F2}
\DeclareUnicodeComposite{\'}            {o}{"00F3}
\DeclareUnicodeComposite{\^}            {o}{"00F4}
\DeclareUnicodeComposite{\~}            {o}{"00F5}
\DeclareUnicodeComposite{\"}            {o}{"00F6}
\DeclareUnicodeComposite{\`}            {u}{"00F9}
\DeclareUnicodeComposite{\'}            {u}{"00FA}
\DeclareUnicodeComposite{\^}            {u}{"00FB}
\DeclareUnicodeComposite{\"}            {u}{"00FC}
\DeclareUnicodeComposite{\'}            {y}{"00FD}
\DeclareUnicodeComposite{\"}            {y}{"00FF}
\DeclareUnicodeComposite{\=}            {A}{"0100}
\DeclareUnicodeComposite{\=}            {a}{"0101}
\DeclareUnicodeComposite{\u}            {A}{"0102}
\DeclareUnicodeComposite{\u}            {a}{"0103}
\DeclareUnicodeComposite{\k}            {A}{"0104}
\DeclareUnicodeComposite{\k}            {a}{"0105}
\DeclareUnicodeComposite{\'}            {C}{"0106}
\DeclareUnicodeComposite{\'}            {c}{"0107}
\DeclareUnicodeComposite{\^}            {C}{"0108}
\DeclareUnicodeComposite{\^}            {c}{"0109}
\DeclareUnicodeComposite{\.}            {C}{"010A}
\DeclareUnicodeComposite{\.}            {c}{"010B}
\DeclareUnicodeComposite{\v}            {C}{"010C}
\DeclareUnicodeComposite{\v}            {c}{"010D}
\DeclareUnicodeComposite{\v}            {D}{"010E}
\DeclareUnicodeComposite{\v}            {d}{"010F}
\DeclareUnicodeComposite{\=}            {E}{"0112}
```

```
\DeclareUnicodeComposite{\=}            {e}{"0113}
\DeclareUnicodeComposite{\u}            {E}{"0114}
\DeclareUnicodeComposite{\u}            {e}{"0115}
\DeclareUnicodeComposite{\.}            {E}{"0116}
\DeclareUnicodeComposite{\.}            {e}{"0117}
\DeclareUnicodeComposite{\k}            {E}{"0118}
\DeclareUnicodeComposite{\k}            {e}{"0119}
\DeclareUnicodeComposite{\v}            {E}{"011A}
\DeclareUnicodeComposite{\v}            {e}{"011B}
\DeclareUnicodeComposite{\^}            {G}{"011C}
\DeclareUnicodeComposite{\^}            {g}{"011D}
\DeclareUnicodeComposite{\u}            {G}{"011E}
\DeclareUnicodeComposite{\u}            {g}{"011F}
\DeclareUnicodeComposite{\.}            {G}{"0120}
\DeclareUnicodeComposite{\.}            {g}{"0121}
\DeclareUnicodeComposite{\c}            {G}{"0122}
\DeclareUnicodeComposite{\c}            {g}{"0123}
\DeclareUnicodeComposite{\^}            {H}{"0124}
\DeclareUnicodeComposite{\^}            {h}{"0125}
\DeclareUnicodeComposite{\~}            {I}{"0128}
\DeclareUnicodeComposite{\~}            \i {"0129}
\DeclareUnicodeComposite{\~}            {i}{"0129}
\DeclareUnicodeComposite{\=}            {I}{"012A}
\DeclareUnicodeComposite{\=}            \i {"012B}
\DeclareUnicodeComposite{\=}            {i}{"012B}
\DeclareUnicodeComposite{\u}            {I}{"012C}
\DeclareUnicodeComposite{\u}            \i {"012D}
\DeclareUnicodeComposite{\u}            {i}{"012D}
\DeclareUnicodeComposite{\k}            {I}{"012E}
\DeclareUnicodeComposite{\k}            \i {"012F}
\DeclareUnicodeComposite{\k}            {i}{"012F}
\DeclareUnicodeComposite{\.}            {I}{"0130}
\DeclareUnicodeComposite{\^}            {J}{"0134}
\DeclareUnicodeComposite{\^}            \j {"0135}
\DeclareUnicodeComposite{\^}            {j}{"0135}
\DeclareUnicodeComposite{\c}            {K}{"0136}
\DeclareUnicodeComposite{\c}            {k}{"0137}
\DeclareUnicodeComposite{\'}            {L}{"0139}
\DeclareUnicodeComposite{\'}            {l}{"013A}
\DeclareUnicodeComposite{\c}            {L}{"013B}
\DeclareUnicodeComposite{\c}            {l}{"013C}
\DeclareUnicodeComposite{\v}            {L}{"013D}
\DeclareUnicodeComposite{\v}            {l}{"013E}
\DeclareUnicodeComposite{\'}            {N}{"0143}
\DeclareUnicodeComposite{\'}            {n}{"0144}
\DeclareUnicodeComposite{\c}            {N}{"0145}
\DeclareUnicodeComposite{\c}            {n}{"0146}
\DeclareUnicodeComposite{\v}            {N}{"0147}
\DeclareUnicodeComposite{\v}            {n}{"0148}
\DeclareUnicodeComposite{\=}            {O}{"014C}
\DeclareUnicodeComposite{\=}            {o}{"014D}
\DeclareUnicodeComposite{\u}            {O}{"014E}
\DeclareUnicodeComposite{\u}            {o}{"014F}
\DeclareUnicodeComposite{\H}            {O}{"0150}
```

```
\DeclareUnicodeComposite{\H}            {o}{"0151}
\DeclareUnicodeComposite{\'}            {R}{"0154}
\DeclareUnicodeComposite{\'}            {r}{"0155}
\DeclareUnicodeComposite{\c}            {R}{"0156}
\DeclareUnicodeComposite{\c}            {r}{"0157}
\DeclareUnicodeComposite{\v}            {R}{"0158}
\DeclareUnicodeComposite{\v}            {r}{"0159}
\DeclareUnicodeComposite{\'}            {S}{"015A}
\DeclareUnicodeComposite{\'}            {s}{"015B}
\DeclareUnicodeComposite{\^}            {S}{"015C}
\DeclareUnicodeComposite{\^}            {s}{"015D}
\DeclareUnicodeComposite{\c}            {S}{"015E}
\DeclareUnicodeComposite{\c}            {s}{"015F}
\DeclareUnicodeComposite{\v}            {S}{"0160}
\DeclareUnicodeComposite{\v}            {s}{"0161}
\DeclareUnicodeComposite{\c}            {T}{"0162}
\DeclareUnicodeComposite{\c}            {t}{"0163}
\DeclareUnicodeComposite{\v}            {T}{"0164}
\DeclareUnicodeComposite{\v}            {t}{"0165}
\DeclareUnicodeComposite{\~}            {U}{"0168}
\DeclareUnicodeComposite{\~}            {u}{"0169}
\DeclareUnicodeComposite{\=}            {U}{"016A}
\DeclareUnicodeComposite{\=}            {u}{"016B}
\DeclareUnicodeComposite{\u}            {U}{"016C}
\DeclareUnicodeComposite{\u}            {u}{"016D}
\DeclareUnicodeComposite{\r}            {U}{"016E}
\DeclareUnicodeComposite{\r}            {u}{"016F}
\DeclareUnicodeComposite{\H}            {U}{"0170}
\DeclareUnicodeComposite{\H}            {u}{"0171}
\DeclareUnicodeComposite{\k}            {U}{"0172}
\DeclareUnicodeComposite{\k}            {u}{"0173}
\DeclareUnicodeComposite{\^}            {W}{"0174}
\DeclareUnicodeComposite{\^}            {w}{"0175}
\DeclareUnicodeComposite{\^}            {Y}{"0176}
\DeclareUnicodeComposite{\^}            {y}{"0177}
\DeclareUnicodeComposite{\"}            {Y}{"0178}
\DeclareUnicodeComposite{\'}            {Z}{"0179}
\DeclareUnicodeComposite{\'}            {z}{"017A}
\DeclareUnicodeComposite{\.}            {Z}{"017B}
\DeclareUnicodeComposite{\.}            {z}{"017C}
\DeclareUnicodeComposite{\v}            {Z}{"017D}
\DeclareUnicodeComposite{\v}            {z}{"017E}
\DeclareUnicodeComposite{\v}            {A}{"01CD}
\DeclareUnicodeComposite{\v}            {a}{"01CE}
\DeclareUnicodeComposite{\v}            {I}{"01CF}
\DeclareUnicodeComposite{\v}            \i {"01D0}
\DeclareUnicodeComposite{\v}            {i}{"01D0}
\DeclareUnicodeComposite{\v}            {O}{"01D1}
\DeclareUnicodeComposite{\v}            {o}{"01D2}
\DeclareUnicodeComposite{\v}            {U}{"01D3}
\DeclareUnicodeComposite{\v}            {u}{"01D4}

\DeclareUnicodeComposite{\'}            \AE{"01FC}
\DeclareUnicodeComposite{\'}            {Æ}{"01FC}
```

```
\DeclareUnicodeComposite{\'}            \ae{"01FD}
\DeclareUnicodeComposite{\'}            {æ}{"01FD}
\DeclareUnicodeComposite{\=}            \AE{"01E2}
\DeclareUnicodeComposite{\=}            {Æ}{"01E2}
\DeclareUnicodeComposite{\=}            \ae{"01E3}
\DeclareUnicodeComposite{\=}            {æ}{"01E3}
\DeclareUnicodeComposite{\v}            {G}{"01E6}
\DeclareUnicodeComposite{\v}            {g}{"01E7}
\DeclareUnicodeComposite{\v}            {K}{"01E8}
\DeclareUnicodeComposite{\v}            {k}{"01E9}
\DeclareUnicodeComposite{\k}            {O}{"01EA}
\DeclareUnicodeComposite{\k}            {o}{"01EB}
\DeclareUnicodeComposite{\v}            \j {"01F0}
\DeclareUnicodeComposite{\v}            {j}{"01F0}
\DeclareUnicodeComposite{\'}            {G}{"01F4}
\DeclareUnicodeComposite{\'}            {g}{"01F5}
\DeclareUnicodeComposite{\textcommabelow}{S}{"0218}
\DeclareUnicodeComposite{\textcommabelow}{s}{"0219}
\DeclareUnicodeComposite{\textcommabelow}{T}{"021A}
\DeclareUnicodeComposite{\textcommabelow}{t}{"021B}
\DeclareUnicodeComposite{\=}            {Y}{"0232}
\DeclareUnicodeComposite{\=}            {y}{"0233}
\DeclareUnicodeComposite{\.}            {B}{"1E02}
\DeclareUnicodeComposite{\.}            {b}{"1E03}
\DeclareUnicodeComposite{\d}            {B}{"1E04}
\DeclareUnicodeComposite{\d}            {b}{"1E05}
\DeclareUnicodeComposite{\d}            {D}{"1E0C}
\DeclareUnicodeComposite{\d}            {d}{"1E0D}
\DeclareUnicodeComposite{\=}            {G}{"1E20}
\DeclareUnicodeComposite{\=}            {g}{"1E21}
\DeclareUnicodeComposite{\d}            {H}{"1E24}
\DeclareUnicodeComposite{\d}            {h}{"1E25}
\DeclareUnicodeComposite{\d}            {K}{"1E32}
\DeclareUnicodeComposite{\d}            {k}{"1E33}
\DeclareUnicodeComposite{\d}            {L}{"1E36}
\DeclareUnicodeComposite{\d}            {l}{"1E37}
\DeclareUnicodeComposite{\d}            {M}{"1E42}
\DeclareUnicodeComposite{\d}            {m}{"1E43}
\DeclareUnicodeComposite{\d}            {N}{"1E46}
\DeclareUnicodeComposite{\d}            {n}{"1E47}
\DeclareUnicodeComposite{\d}            {R}{"1E5A}
\DeclareUnicodeComposite{\d}            {r}{"1E5B}
\DeclareUnicodeComposite{\d}            {S}{"1E62}
\DeclareUnicodeComposite{\d}            {s}{"1E63}
\DeclareUnicodeComposite{\d}            {T}{"1E6C}
\DeclareUnicodeComposite{\d}            {t}{"1E6D}
\DeclareUnicodeComposite{\d}            {V}{"1E7E}
\DeclareUnicodeComposite{\d}            {v}{"1E7F}
\DeclareUnicodeComposite{\d}            {W}{"1E88}
\DeclareUnicodeComposite{\d}            {w}{"1E89}
\DeclareUnicodeComposite{\d}            {Z}{"1E92}
\DeclareUnicodeComposite{\d}            {z}{"1E93}
\DeclareUnicodeComposite{\d}            {A}{"1EA0}
\DeclareUnicodeComposite{\d}            {a}{"1EA1}
```

```
\DeclareUnicodeComposite{\d}            {E}{"1EB8}
\DeclareUnicodeComposite{\d}            {e}{"1EB9}
\DeclareUnicodeComposite{\d}            {I}{"1ECA}
\DeclareUnicodeComposite{\d}            {i}{"1ECB}
\DeclareUnicodeComposite{\d}            {O}{"1ECC}
\DeclareUnicodeComposite{\d}            {o}{"1ECD}
\DeclareUnicodeComposite{\d}            {U}{"1EE4}
\DeclareUnicodeComposite{\d}            {u}{"1EE5}
\DeclareUnicodeComposite{\d}            {Y}{"1EF4}
\DeclareUnicodeComposite{\d}            {y}{"1EF5}
```

⟨/TU⟩

2 Package files

This file now also contains some packages that provide access to the more specialised encodings.

2.1 The fontenc package

This package allows authors to specify which encodings they will use. For each encoding `FOO`, the package looks to see if the encoding `FOO` has already been declared. If it has not, the file `fooenc.def` is loaded. The default encoding is set to be `FOO`.

In addition the package at the moment contains extra code to extend the `\@uclclist` (list of upper/lower case pairs) for encodings that involve cyrillic characters. THIS IS A TEMPORARY SOLUTION and will not stay this way forever (or so we hope) but right now we are missing a proper interface for this and didn't wanted to rush it.

⟨∗package⟩

Here we define a macro that extends the `\@uclclist` if needed and afterwards turns itself in a noop.

```
\def\update@uclc@with@cyrillic{%
 \expandafter\def\expandafter\@uclclist\expandafter
  {\@uclclist
  \cyra\CYRA\cyrabhch\CYRABHCH\cyrabhchdsc\CYRABHCHDSC\cyrabhdze
  \CYRABHDZE\cyrabhha\CYRABHHA\cyrae\CYRAE\cyrb\CYRB\cyrbyus
  \CYRBYUS\cyrc\CYRC\cyrch\CYRCH\cyrchldsc\CYRCHLDSC\cyrchrdsc
  \CYRCHRDSC\cyrchvcrs\CYRCHVCRS\cyrd\CYRD\cyrdelta\CYRDELTA
  \cyrdje\CYRDJE\cyrdze\CYRDZE\cyrdzhe\CYRDZHE\cyre\CYRE\cyreps
  \CYREPS\cyrerev\CYREREV\cyrery\CYRERY\cyrf\CYRF\cyrfita
  \CYRFITA\cyrg\CYRG\cyrgdsc\CYRGDSC\cyrgdschcrs\CYRGDSCHCRS
  \cyrghcrs\CYRGHCRS\cyrghk\CYRGHK\cyrgup\CYRGUP\cyrh\CYRH
  \cyrhdsc\CYRHDSC\cyrhhcrs\CYRHHCRS\cyrhhk\CYRHHK\cyrhrdsn
  \CYRHRDSN\cyri\CYRI\cyrie\CYRIE\cyrii\CYRII\cyrishrt\CYRISHRT
  \cyrishrtdsc\CYRISHRTDSC\cyrizh\CYRIZH\cyrje\CYRJE\cyrk\CYRK
  \cyrkbeak\CYRKBEAK\cyrkdsc\CYRKDSC\cyrkhcrs\CYRKHCRS\cyrkhk
  \CYRKHK\cyrkvcrs\CYRKVCRS\cyrl\CYRL\cyrldsc\CYRLDSC\cyrlhk
  \CYRLHK\cyrlje\CYRLJE\cyrm\CYRM\cyrmdsc\CYRMDSC\cyrmhk\CYRMHK
  \cyrn\CYRN\cyrndsc\CYRNDSC\cyrng\CYRNG\cyrnhk\CYRNHK\cyrnje
  \CYRNJE\cyrnlhk\CYRNLHK\cyro\CYRO\cyrotld\CYROTLD\cyrp\CYRP
  \cyrphk\CYRPHK\cyrq\CYRQ\cyrr\CYRR\cyrrdsc\CYRRDSC\cyrrhk
  \CYRRHK\cyrrtick\CYRRTICK\cyrs\CYRS\cyrsacrs\CYRSACRS
  \cyrschwa\CYRSCHWA\cyrsdsc\CYRSDSC\cyrsemisftsn\CYRSEMISFTSN
```

```
  \cyrsftsn\CYRSFTSN\cyrsh\CYRSH\cyrshch\CYRSHCH\cyrshha\CYRSHHA
  \cyrt\CYRT\cyrtdsc\CYRTDSC\cyrtetse\CYRTETSE\cyrtshe\CYRTSHE
  \cyru\CYRU\cyrushrt\CYRUSHRT\cyrv\CYRV\cyrw\CYRW\cyry\CYRY
  \cyrya\CYRYA\cyryat\CYRYAT\cyryhcrs\CYRYHCRS\cyryi\CYRYI\cyryo
  \CYRYO\cyryu\CYRYU\cyrz\CYRZ\cyrzdsc\CYRZDSC\cyrzh\CYRZH
  \cyrzhdsc\CYRZHDSC}%
 \let\update@uclc@with@cyrillic\relax
}
```

Here we process each option:

```
\DeclareOption*{%
  \let\encodingdefault\CurrentOption
```

From 2020/02/02 release onward we only load the encoding files if they haven't be loaded already. To check this we look at whether `\T@`*encoding* is already defined. If not, we load it later (indicated by setting the switch `@tempswa` to true) and we always load if we are using an older format (or rather in a rollback situation).

```
  \@tempswafalse
  \@ifl@t@r\fmtversion{2020/02/02}%
      {\expandafter\ifx\csname T@\CurrentOption\endcsname\relax
        \@tempswatrue\fi}%
      {\@tempswatrue}%
```

Load if necessary:

```
  \if@tempswa

    \edef\reserved@f{%
      \lowercase{\def\noexpand\reserved@f{\CurrentOption enc.def}}}%
    \reserved@f
    \InputIfFileExists\reserved@f
         {}{\PackageError{fontenc}%
          {Encoding file `\reserved@f' not found.%
           \MessageBreak
           You might have misspelled the name of the encoding
           \MessageBreak
           or a required support package (e.g., cyrillic) is
           \MessageBreak
           missing in your installation}%
          {Necessary code for this encoding was not
           loaded.\MessageBreak
           Thus calling the encoding later on will
           produce further error messages.}}%
   \let\reserved@f\relax
```

In case the current encoding is one of a list of known cyrillic ones we extend the `\@uclclist`:

```
   \expandafter\in@\expandafter{\CurrentOption}%
                                {T2A,T2B,T2C,X2,LCY,OT2}%
   \ifin@
```

But only if it hasn't already been extended. This might happen if there are several calls to fontenc loading one of the above encodings. If we don't do this check the `\@uclclist` gets unnecessarily big, slowing down the processing at runtime.

```
      \expandafter\in@\expandafter\cyra\expandafter
                                {\@uclclist}%
      \ifin@
```

```
      \else
        \update@uclc@with@cyrillic
      \fi
    \fi
  \fi
}
\ProcessOptions*
```

We select the new font encoding default (i.e., the last encoding specified in the option list). But this encoding may not work with the current `\f@shape`: e.g., `LY1` is not defined for `cmr` and therefore packages switching to `LY1` usually also change `\rmdefault`. But that only applies at `\begin{document}` so we get a spurious warning if we use what LaTeX previously used:

```
%\fontencoding\encodingdefault\selectfont
```

So instead we do this here:

```
\usefont\encodingdefault\familydefault\seriesdefault\shapedefault
```

To save some space we get rid of the macro extending the `\@uclclist` (might have happened already).

```
\let\update@uclc@with@cyrillic\relax
```

Finally we pretend that the fontenc package wasn't read in. This allows for using it several times, e.g., in a class file and in the preamble (at the cost of not getting any version info). That kind of hackery shows that using a general purpose package just for loading an encoding is not the right kind of interface for setting up encodings — it will get replaced at some point in the future.

```
\let\@elt\relax
\xdef\@fontenc@load@list{\@fontenc@load@list
  \@elt{\csname opt@fontenc.sty\endcsname}}

\global\expandafter\let\csname ver@@fontenc.sty\expandafter\endcsname
                        \csname ver@fontenc.sty\endcsname
\global\expandafter\let\csname ver@fontenc.sty\endcsname\relax
\global\expandafter\let\csname opt@fontenc.sty\endcsname\relax
\global\let\@ifl@ter@@\@ifl@ter
\def\@ifl@ter#1#2#3#4#5{\global\let\@ifl@ter\@ifl@ter@@}
⟨/package⟩
```

File 22
ltcounts.dtx

1 Counters and Lengths

Commands for defining and using counters. This file defines:

`\newcounter` To define a new counter.

`\setcounter` To set the value of counters.

`\addtocounter` Increase the counter `#1` by the number `#2`.

`\stepcounter` Increase a counter by one.

`\refstepcounter` Increase a counter by one, also setting the value used by `\label`.

`\value` For accessing the value of the counter as a TEX number (as opposed to `\the`⟨*counter*⟩ which expands to the *printed* representation of ⟨*counter*⟩)

`\arabic` `\arabic{`⟨*counter*⟩`}`: 1, 2, 3, ...

`\roman` `\roman{`⟨*counter*⟩`}`: i, ii, iii, ...

`\Roman` `\Roman{`⟨*counter*⟩`}`: I, II, III, ...

`\alph` `\alph{`⟨*counter*⟩`}`: a, b, c, ...

`\Alph` `\Alph{`⟨*counter*⟩`}`: A, B, C, ...

`\fnsymbol` `\fnsymbol{`⟨*counter*⟩`}`: ∗, †, ‡, ...

`\counterwithin` `\counterwithin[`⟨*format*⟩`]{`⟨*counter*⟩`}{`⟨*within-counter*⟩`}`: Resets ⟨*counter*⟩ whenever ⟨*within-counter*⟩ is stepped. Also redefines `\the`⟨*counter*⟩ command to produce `\the`⟨*within-counter*⟩`.`⟨*format*⟩`{`⟨*counter*⟩`}` with `\arabic` as the default for ⟨*format*⟩. Star form omits redefining the print representation.

`\counterwithout` `\counterwithout[`⟨*format*⟩`]{`⟨*counter*⟩`}{`⟨*within-counter*⟩`}`: Removes ⟨*counter*⟩ from the reset list of ⟨*within-counter*⟩. Also redefines `\the`⟨*counter*⟩ command to produce ⟨*format*⟩`{`⟨*counter*⟩`}` with `\arabic` as the default for ⟨*format*⟩. Star form omits redefining the print representation.

```
⟨*2ekernel⟩
```

1.1 Environment Counter Macros

An environment foo has an associated counter defined by the following control sequences:

`\c@foo` Contains the counter's numerical value. It is defined by `\newcount\foocounter`.

`\thefoo` Macro that expands to the printed value of `\foocounter`. For example, if sections are numbered within chapters, and section headings look like
Section II-3. The Nature of Counters
then `\thesection` might be defined by:

```
\def\thesection
    {\@Roman{\c@chapter}-\@arabic{\c@section}}
```

`\p@foo` Macro that expands to a printed 'reference prefix' of counter foo. Any `\ref` to a value created by counter foo will produce the expansion of `\p@foo\thefoo` when the `\label` command is executed. See file `ltxref.dtx` for an extension of this mechanism.

`\cl@foo` List of counters to be reset when foo stepped. Has format `\@elt{countera}\@elt{counterb}\@elt{counterc}`.

NOTE:
\thefoo and \p@foo *must* be defined in such a way that \edef\bar{\thefoo} or \edef\bar{\p@foo} defines \bar so that it will evaluate to the counter value at the time of the \edef, even after \foocounter and any other counters have been changed. This will happen if you use the standard commands \@arabic, \@Roman, etc.

The following commands are used to define and modify counters.

\refstepcounter{⟨*foo*⟩}
Same as \stepcounter, but it also defines \@currentlabel, \@currentHref and \@currentcounter and so that a subsequent \label{⟨*bar*⟩} command causes \ref{⟨*bar*⟩} to generate the current value of counter ⟨*foo*⟩.

\@definecounter{⟨*foo*⟩}
Initializes counter {⟨*foo*⟩} (with empty reset list), defines \p@foo and \thefoo to be null and \theHfoo to be \number\value{foo}. Also adds ⟨*foo*⟩ to \cl@@ckpt – the reset list of a dummy counter @ckpt used for taking checkpoints for the \include system.

\@addtoreset{⟨*foo*⟩}{⟨*bar*⟩} : Adds counter ⟨*foo*⟩ to the list of counters \cl@bar to be reset when counter ⟨*bar*⟩ is stepped.

\@removefromreset{⟨*foo*⟩}{⟨*bar*⟩} : Removes counter ⟨*foo*⟩ to the list of counters \cl@bar to be reset when counter ⟨*bar*⟩ is stepped.

\setcounter

\setcounter{⟨*foo*⟩}{⟨*val*⟩} : Globally sets \foocounter equal to ⟨*val*⟩.

```
\def\setcounter#1#2{%
  \@ifundefined{c@#1}%
    {\@nocounterr{#1}}%
    {\global\csname c@#1\endcsname#2\relax}}
```

(*End of definition for* \setcounter.)

\addtocounter

\addtocounter{⟨*foo*⟩}{⟨*val*⟩} Globally increments \foocounter by ⟨*val*⟩.

```
\def\addtocounter#1#2{%
  \@ifundefined{c@#1}%
    {\@nocounterr{#1}}%
    {\global\advance\csname c@#1\endcsname #2\relax}}
```

(*End of definition for* \addtocounter.)

\newcounter

\newcounter{⟨*newctr*⟩}[⟨*oldctr*⟩] Defines ⟨*newctr*⟩ to be a counter, which is reset when counter ⟨*oldctr*⟩ is stepped. If ⟨*newctr*⟩ already defined produces '`c@newctr already defined`' error.

```
\def\newcounter#1{%
  \expandafter\@ifdefinable \csname c@#1\endcsname
    {\@definecounter{#1}}%
  \@ifnextchar[{\@newctr{#1}}{}}
```

(*End of definition for* \newcounter.)

\value

\value{⟨*ctr*⟩} produces the value of counter ⟨*ctr*⟩, for use with a \setcounter or \addtocounter command.

```
\def\value#1{\csname c@#1\endcsname}
```

(*End of definition for* \value.)

\@newctr

```
\def\@newctr#1[#2]{%
  \@ifundefined{c@#2}{\@nocounterr{#2}}{\@addtoreset{#1}{#2}}}
```

(*End of definition for* `\@newctr`.)

`\stepcounter` `\stepcounter`foo Globally increments counter `\c@FOO` and resets all subsidiary counters.

```
\def\stepcounter#1{%
  \addtocounter{#1}\@ne
  \begingroup
    \let\@elt\@stpelt
    \csname cl@#1\endcsname
  \endgroup}
```

(*End of definition for* `\stepcounter`.)

`\@stpelt` Rather than resetting the "within" counter to zero we set it to −1 and then run `\stepcounter` that moves it to 0 and also initiates resetting the next level down.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@stpelt}
⟨latexrelease⟩                              {Reset nested counters}%
⟨*2ekernel | latexrelease⟩
\def\@stpelt#1{\global\csname c@#1\endcsname \m@ne\stepcounter{#1}}%
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@stpelt}
⟨latexrelease⟩                              {Reset nested counters}%%
⟨latexrelease⟩\def\@stpelt#1{\global\csname c@#1\endcsname \z@}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@stpelt`.)

`\cl@@ckpt`

```
\def\cl@@ckpt{\@elt{page}}
```

(*End of definition for* `\cl@@ckpt`.)

`\@definecounter`

```
%  \changes{v1.1p}{2024/10/26}{Fully expand counter name in theHfoo commands (gh/1508)}
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2024/11/01}{\@definecounter}
⟨latexrelease⟩                              {provide theHfoo commands}%
⟨*2ekernel | latexrelease⟩
\def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
     \setcounter{#1}\z@
     \global\expandafter\let\csname cl@#1\endcsname\@empty
     \@addtoreset{#1}{@ckpt}%
     \global\expandafter\let\csname p@#1\endcsname\@empty
     \expandafter\xdef\csname theH#1\endcsname{%
       \noexpand\the\noexpand\value{#1}}%
```

If `\the#1` is undefined or `\relax` we define it with the standard definition for counters, otherwise we warn. This will catch, for example, that somebody defines a counter named "index" conflicting with the `theindex` environment.

```
     \expandafter
     \ifx\csname the#1\endcsname\relax
       \expandafter
       \gdef\csname the#1\expandafter\endcsname\expandafter
```

```
              {\expandafter\@arabic\csname c@#1\endcsname}%
     \else
       \@latex@warning{Command `\string\the#1' already
                        defined -- not changed}%
     \fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@definecounter}
⟨latexrelease⟩                              {provide theHfoo commands}%%
⟨latexrelease⟩\def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
⟨latexrelease⟩     \setcounter{#1}\z@
⟨latexrelease⟩     \global\expandafter\let\csname cl@#1\endcsname\@empty
⟨latexrelease⟩     \@addtoreset{#1}{@ckpt}%
⟨latexrelease⟩     \global\expandafter\let\csname p@#1\endcsname\@empty
⟨latexrelease⟩     \expandafter
⟨latexrelease⟩     \ifx\csname the#1\endcsname\relax
⟨latexrelease⟩       \expandafter
⟨latexrelease⟩     \gdef\csname the#1\expandafter\endcsname\expandafter
⟨latexrelease⟩         {\expandafter\@arabic\csname c@#1\endcsname}%
⟨latexrelease⟩     \else
⟨latexrelease⟩       \@latex@warning{Command `\string\the#1' already
⟨latexrelease⟩                       defined -- not changed}%
⟨latexrelease⟩     \fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@definecounter`.)

`\@addtoreset` If a counter is reset when a parent counter changes it no longer has an unique value across the document. As `\theH<counter>` should be unique this representation is changed to include also the representation of the parent. This is not 100% guaranteed to work but has been used this way by hyperref for many years without causing problems.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2024/11/01}{\@addtoreset}
⟨latexrelease⟩                              {provide theHfoo commands}%
⟨*2ekernel | latexrelease⟩
\def\@addtoreset#1#2{\expandafter\@cons\csname cl@#2\endcsname {{#1}}%
  \expandafter\xdef\csname theH#1\endcsname{%
    \expandafter\noexpand\csname theH#2\endcsname.%
    \noexpand\the\noexpand\value{#1}}%
}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@addtoreset}
⟨latexrelease⟩                              {provide theHfoo commands}%%
⟨latexrelease⟩\def\@addtoreset#1#2{\expandafter\@cons\csname cl@#2\endcsname {{#1}}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@addtoreset`.)

```
⟨/2ekernel⟩
```

`\@removefromreset`

```
⟨latexrelease⟩\IncludeInRelease{2018-04-01}
```

```
⟨latexrelease⟩                  {\@removefromreset}{Add interfaces}%
⟨*2ekernel | latexrelease⟩
\def\@removefromreset#1#2{%
```

Even through this is internal and the programmer should know what he/she is doing we test here if counter #2 is defined. If not, the execution would run into a tight loop.

```
  \@ifundefined{c@#2}\relax
  {\begingroup
     \expandafter\let\csname c@#1\endcsname\@removefromreset
     \def\@elt##1{%
       \expandafter\ifx\csname c@##1\endcsname\@removefromreset
       \else
         \noexpand\@elt{##1}%
       \fi}%
     \expandafter\xdef\csname cl@#2\endcsname
      {\csname cl@#2\endcsname}%
    \endgroup}}
```

(*End of definition for* `\@removefromreset`.)

`\@ifbothcounters` Test if arg #1 and #2 are counters and if so execute #3.

```
\def\@ifbothcounters#1#2#3{%
  \@ifundefined{c@#1}{\@nocounterr{#1}}%
    {% else counter is defined
     \@ifundefined{c@#2}{\@nocounterr{#2}}%
       {% else both counter and within  are defined
        #3}}}
```

(*End of definition for* `\@ifbothcounters`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}
⟨latexrelease⟩                  {\@removefromreset}{Add interfaces}%
⟨latexrelease⟩\let \@removefromreset \undefined
⟨latexrelease⟩\let \@ifbothcounters  \undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\counterwithout` `\counterwithin` New implementation using xparse and supporting an optional format argument.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                  {\counterwithout}{counter without/within}%
\NewDocumentCommand \counterwithout {sO{\arabic}mm}{%
  \@ifbothcounters{#3}{#4}{%
    \@removefromreset{#3}{#4}%
    \IfBooleanF #1%
        {\expandafter
         \gdef\csname the#3\endcsname {#2{#3}}}%
  }%
}
```

```
\NewDocumentCommand \counterwithin {sO{\arabic}mm}{%
  \@ifbothcounters{#3}{#4}{%
    \@addtoreset{#3}{#4}%
    \IfBooleanF #1%
       {\expandafter
         \gdef\csname the#3\expandafter\endcsname
         \expandafter
             {\csname the#4\endcsname .#2{#3}}}%
  }%
}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2018-04-01}
⟨latexrelease⟩                 {\counterwithout}{counter without/within}%
⟨latexrelease⟩
⟨latexrelease⟩\def\counterwithout {\@ifstar\counterwithout@s\counterwithout@x}
⟨latexrelease⟩\def\counterwithout@s#1#2{%
⟨latexrelease⟩  \@ifbothcounters{#1}{#2}{\@removefromreset{#1}{#2}}}
⟨latexrelease⟩\def\counterwithout@x#1#2{%
⟨latexrelease⟩  \@ifbothcounters{#1}{#2}%
⟨latexrelease⟩      {\@removefromreset{#1}{#2}%
⟨latexrelease⟩       \expandafter
⟨latexrelease⟩       \gdef\csname the#1\expandafter\endcsname\expandafter
⟨latexrelease⟩            {\expandafter
⟨latexrelease⟩             \@arabic\csname c@#1\endcsname}}}
⟨latexrelease⟩
⟨latexrelease⟩\def\counterwithin{\@ifstar\counterwithin@s\counterwithin@x}
⟨latexrelease⟩\def\counterwithin@s#1#2{%
⟨latexrelease⟩  \@ifbothcounters{#1}{#2}{\@addtoreset{#1}{#2}}}
```

```
⟨latexrelease⟩\def\counterwithin@x#1#2{%
⟨latexrelease⟩  \@ifbothcounters{#1}{#2}%
⟨latexrelease⟩      {\@addtoreset{#1}{#2}%
⟨latexrelease⟩       \expandafter
⟨latexrelease⟩       \gdef\csname the#1\expandafter\endcsname\expandafter
⟨latexrelease⟩            {\csname the#2\expandafter\endcsname\expandafter
⟨latexrelease⟩             .\expandafter
⟨latexrelease⟩             \@arabic\csname c@#1\endcsname}}}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}
⟨latexrelease⟩                 {\counterwithout}{counter without/within}%
⟨latexrelease⟩\let \counterwithout   \undefined
⟨latexrelease⟩\let \counterwithout@s \undefined
⟨latexrelease⟩\let \counterwithout@x \undefined
⟨latexrelease⟩\let \counterwithin    \undefined
⟨latexrelease⟩\let \counterwithin@s  \undefined
⟨latexrelease⟩\let \counterwithin@x  \undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\counterwithout` *and* `\counterwithin`.)

Numbering commands for definitions of `\theCOUNTER` and `\list` arguments. All commands can now be used in text and math mode.

`\arabic` Representation of⟨*counter*⟩ as arabic numerals. Changed 29 Apr 86 to make it print the obvious thing it COUNTER not positive.

```
\def\arabic#1{\expandafter\@arabic\csname c@#1\endcsname}
```

(*End of definition for* `\arabic`.)

`\roman` Representation of ⟨*counter*⟩ as lower-case Roman numerals.

```
\def\roman#1{\expandafter\@roman\csname c@#1\endcsname}
```

(*End of definition for* `\roman`.)

`\Roman` Representation of ⟨*counter*⟩ as upper-case Roman numerals.

```
\def\Roman#1{\expandafter\@Roman\csname c@#1\endcsname}
```

(*End of definition for* `\Roman`.)

`\alph` Representation of ⟨*counter*⟩ as a lower-case letter: 1 = a, 2 = b, etc.

```
\def\alph#1{\expandafter\@alph\csname c@#1\endcsname}
```

(*End of definition for* `\alph`.)

`\Alph` Representation of ⟨*counter*⟩ as an upper-case letter: 1 = A, 2 = B, etc.

```
\def\Alph#1{\expandafter\@Alph\csname c@#1\endcsname}
```

(*End of definition for* `\Alph`.)

`\fnsymbol` Representation of ⟨*COUNTER*⟩ as a footnote symbol: 1 = ∗, 2 = †, etc.

```
\def\fnsymbol#1{\expandafter\@fnsymbol\csname c@#1\endcsname}
```

(*End of definition for* `\fnsymbol`.)

`\@arabic` `\@arabic\FOOcounter` Representation of `\FOOcounter` as arabic numerals.

```
\def\@arabic#1{\number #1}  %% changed 29 Apr 86
```

(*End of definition for* `\@arabic`.)

`\@roman` `\@roman\FOOcounter` Representation of `\FOOcounter` as lower-case Roman numerals.

```
\def\@roman#1{\romannumeral #1}
```

(*End of definition for* `\@roman`.)

`\@Roman` `\@Roman\FOOcounter` Representation of `\FOOcounter` as upper-case Roman numerals.

```
\def\@Roman#1{\expandafter\@slowromancap\romannumeral #1@}
```

(*End of definition for* `\@Roman`.)

`\@slowromancap` Fully expandable macro to change a roman number to uppercase.

```
\def\@slowromancap#1{\ifx @#1% then terminate
     \else
       \if i#1I\else\if v#1V\else\if x#1X\else\if l#1L\else\if
       c#1C\else\if d#1D\else \if m#1M\else#1\fi\fi\fi\fi\fi\fi\fi
       \expandafter\@slowromancap
     \fi
}
```

(*End of definition for* `\@slowromancap`.)

\@alph \@alph\FOOcounter Representation of \FOOcounter as a lower-case letter: 1 = a, 2 = b, etc.

```
\def\@alph#1{%
  \ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or
   k\or l\or m\or n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or x\or
    y\or z\else\@ctrerr\fi}
```

(*End of definition for* \@alph.)

\@Alph \@Alph\FOOcounter Representation of \FOOcounter as an upper-case letter: 1 = A, 2 = B, etc.

```
\def\@Alph#1{%
  \ifcase#1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or
   K\or L\or M\or N\or O\or P\or Q\or R\or S\or T\or U\or V\or W\or X\or
    Y\or Z\else\@ctrerr\fi}
```

(*End of definition for* \@Alph.)

\@fnsymbol Typesetting old fashioned footnote symbols. This can be done both in text or math mode now.

This macro is another example of an ever recurring problem in TEX: Determining if something is text-mode or math-mode. It is imperative for the decision between text and math to be delayed until the actual typesetting is done as the code in question may go through an \edef or \write where an \ifmmode test would be executed prematurely. Hence in the implementation below, \@fnsymbol is not robust in itself but the parts doing the actual typesetting are.

In the case of \@fnsymbol we make use of the robust command \TextOrMath which takes two arguments and typesets the first if in text-mode and the second if in math-mode. Note that in order for this command to make the correct decision, it must insert a \relax token if run under regular TEX, which ruins any kerning between the preceding characters and whatever awaits typesetting. If you use eTEX as engine for LATEX (as recommended) this unfortunate side effect is not present.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@fnsymbol}{Use \TextOrMath}%
⟨*2ekernel | latexrelease⟩
\def\@fnsymbol#1{%
   \ifcase#1\or \TextOrMath\textasteriskcentered *\or
   \TextOrMath \textdagger \dagger\or
   \TextOrMath \textdaggerdbl \ddagger \or
   \TextOrMath \textsection  \mathsection\or
   \TextOrMath \textparagraph \mathparagraph\or
   \TextOrMath \textbardbl \|\or
   \TextOrMath {\textasteriskcentered\textasteriskcentered}{**}\or
   \TextOrMath {\textdagger\textdagger}{\dagger\dagger}\or
   \TextOrMath {\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger}\else
   \@ctrerr \fi
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@fnsymbol}{Use \TextOrMath}%
⟨latexrelease⟩\def\@fnsymbol#1{\ensuremath{%
⟨latexrelease⟩   \ifcase#1\or *\or \dagger\or \ddagger\or \mathsection\or
⟨latexrelease⟩      \mathparagraph\or \|\or **\or \dagger\dagger
```

```
⟨latexrelease⟩      \or \ddagger\ddagger \else\@ctrerr\fi}}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@fnsymbol`.)

`\TextOrMath` When using regular TEX, we make this command robust so that it always selects the correct branch in an `\ifmmode` switch with the usual disadvantage of ruining kerning. For the application we use it for here that shouldn't matter. The alternative would be to mimic `\IeC` from inputenc but then it will have the disadvantage of choosing the wrong branch if appearing at the beginning of an alignment cell. However, users of eTEX will be pleasantly surprised to get the best of both worlds and no bad side effects.

First some code for checking if we are running eTEX but making sure not to permanently turn `\protected` into `\relax`.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\TextOrMath}{\TextOrMath}%
⟨*2ekernel | latexrelease⟩
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname protected\endcsname\relax
```

In case of ordinary TEX we define `\TextOrMath` as a robust command but make sure it always grabs its arguments. If we didn't do this it might very well gobble spaces in the input stream.

```
\DeclareRobustCommand\TextOrMath{%
  \ifmmode  \expandafter\@secondoftwo
  \else     \expandafter\@firstoftwo  \fi}
\protected@edef\TextOrMath#1#2{\TextOrMath{#1}{#2}}
\else
```

For eTEX the situation is similar. The robust macro is a hidden one so that we again avoid problems of gobbling spaces in the input.

```
\protected\expandafter\def\csname TextOrMath\space\endcsname{%
  \ifmmode  \expandafter\@secondoftwo
  \else     \expandafter\@firstoftwo  \fi}
\edef\TextOrMath#1#2{%
  \expandafter\noexpand\csname TextOrMath\space\endcsname
  {#1}{#2}}
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\TextOrMath}{\TextOrMath}%
⟨latexrelease⟩\let\TextOrMath\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\TextOrMath`.)

```
⟨/2ekernel⟩
```

File 23
ltlength.dtx

1 Lengths

\newlength Declare #1 to be a new length command.
\setlength Set the length command, #1, to the value #2.
\addtolength Increase the value of the length command, #1, by the value #2.
\settowidth Set the length, #1 to the width of a box containing #2.
\settoheight Set the length, #1 to the height of a box containing #2.
\settodepth Set the length, #1 to the depth of a box containing #2.

```
⟨*2ekernel⟩
\message{lengths,}
```

\newlength

```
\def\newlength#1{\@ifdefinable#1{\newskip#1}}
```

(*End of definition for* \newlength.)

\setlength

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\setlength}{Using \setlength with \dimen0}%
⟨*2ekernel | latexrelease⟩
\def\setlength#1#2{#1 #2\relax}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\setlength}{Using \setlength with \dimen0}%
⟨latexrelease⟩\def\setlength#1#2{#1#2\relax}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \setlength.)

\addtolength \relax added 24 Mar 86

```
\def\addtolength#1#2{\advance#1 #2\relax}
```

(*End of definition for* \addtolength.)

\settoheight The obvious analogs of \settowidth.
\settodepth
\settowidth
\@settodim

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/11/01}%
⟨latexrelease⟩                  {\@settodim}{suspend tagging}%
\def\@settodim#1#2#3{\setbox\@tempboxa\hbox
  {{\SuspendTagging{\@settodim}#3\ResumeTagging{\@settodim}}}#2#1\@tempboxa
```

Clear the memory afterwards (which might be a lot).

```
        \setbox\@tempboxa\box\voidb@x}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@settodim}{suspend tagging}%
⟨latexrelease⟩\def\@settodim#1#2#3{\setbox\@tempboxa\hbox{{#3}}#2#1\@tempboxa
⟨latexrelease⟩       \setbox\@tempboxa\box\voidb@x}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\DeclareRobustCommand\settoheight{\@settodim\ht}
\DeclareRobustCommand\settodepth {\@settodim\dp}
\DeclareRobustCommand\settowidth {\@settodim\wd}
```

(*End of definition for* `\settoheight` *and others.*)

`\@settopoint` This macro takes the contents of the skip register that is supplied as its argument and removes the fractional part to make it a whole number of points. This can be used in class files to avoid values like `345.4666666pt` when calculating a dimension.

```
\def\@settopoint#1{\divide#1\p@\multiply#1\p@}
⟨/2ekernel⟩
```

(*End of definition for* `\@settopoint`.)

File 24
ltfssbas.dtx

This file contains the main implementation of the 'low level' font selection commands. See other parts of the LaTeX distribution, or *The LaTeX Companion* for higher level documentation of the LaTeX 'New' Font Selection Scheme.

> **Warning:** The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

1 Preliminary macros

We define a number of macros that will be used later.

`\@nomath` `\@nomath` is used by most macros that will have no effect in math mode. It issues a warning message.

```
⟨*2ekernel⟩
\def\@nomath#1{\relax\ifmmode
   \@font@warning{Command \noexpand#1invalid in math mode}\fi}
```

(*End of definition for* `\@nomath`.)

`\no@alphabet@error` The macro `\no@alphabet@error` is called whenever the user requests a math *alphabet* that is not available in the current *version*. In math mode an error message is produced otherwise the command keeps silent. The argument is the name of the control sequence that identifies the math *alphabet*. The `\relax` at the beginning is necessary to prevent TeX from scanning too far in certain situations.

```
\gdef\no@alphabet@error#1{\relax \ifmmode
    \@latex@error{Math\space alphabet\space identifier\space
          \noexpand#1is\space undefined\space in\space math\space
           version\space `\math@version'}%
        {Your\space requested\space math\space alphabet\space
         is\space undefined\space in\space the\space current\space
          math\space version.^^JCheck\space the\space spelling\space
          or\space use\space the\space \noexpand\SetMathAlphabet\space
          command.}
     \fi}
```

(*End of definition for* `\no@alphabet@error`.)

`\new@mathgroup` `\mathgroup` We also give a new name to `\newfam` and `\fam` to avoid verbal confusion (see the introduction).[31]

```
%\def\new@mathgroup{\alloc@8\mathgroup\chardef\sixt@@n}
\let\mathgroup\fam
%\let\newfam\new@mathgroup
\@onlypreamble\new@mathgroup
```

(*End of definition for* `\new@mathgroup` *and* `\mathgroup`.)

[31] For the same reason it seems advisable to `\let\fam` and `\newfam` equal to `\relax`, but this is commented out to retain compatibility to existing style files.

2 Macros for setting up the tables

\DeclareFontShape

The macro \DeclareFontShape takes 6 arguments:

```
\def\DeclareFontShape{\begingroup
```

First we restore the catcodes of all characters used in the syntax.

```
  \nfss@catcodes
```

We use \expandafter \endgroup to restore catcode in case something goes wrong with the argument parsing (suggested by Tim Van Zandt)

```
  \expandafter\endgroup
  \DeclareFontShape@}
```

(*End of definition for* \DeclareFontShape.)

\DeclareFontShape@

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                    {\DeclareFontShape@}{Maybe drop one m}%
\def\DeclareFontShape@#1#2#3#4#5#6{%
  \expandafter\ifx\csname #1+#2\endcsname\relax
    \@latex@error{Font family ‘#1+#2’ unknown}\@eha
  \else
```

If the series value is incorrectly specified with an extra “m”, e.g., “mc” instead of just “c”, drop the surplus “m” but keep the “m” if it is by its own. In that case also issue a warning that the declaration needs correction.

For this we compare the given value #3 with one where we may have dropped an “m”. If nothing has changes, fine. Otherwise there was a wrong value which is now corrected in \reserved@b so we use that and also issue a warning.

```
    \edef\reserved@a{#3}%
    \series@maybe@drop@one@m\reserved@a\reserved@b
    \ifx\reserved@a\reserved@b\else
      \@latex@note{Font shape #1/#2/#3/#4 has incorrect series
        value ‘#3’.\MessageBreak It should not contain an ‘m’!
        Please correct it.\MessageBreak Found}%
    \fi
    \expandafter
      \xdef\csname#1/#2/\reserved@b/#4\endcsname
                {\expandafter\noexpand\csname #5\endcsname}%
%
```

Most of the time #6 is empty so using \let to \@empty saves on space compared to using \def. That’s really one of the old space saving techniques and probably not necessary these days.

```
    \def\reserved@a{#6}%
    \global
    \expandafter\let\csname#5\expandafter\endcsname
       \ifx\reserved@a\@empty
         \@empty
       \else
         \reserved@a
       \fi
  \fi
 }
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\DeclareFontShape@}{Maybe drop one m}%
⟨latexrelease⟩
⟨latexrelease⟩\def\DeclareFontShape@#1#2#3#4#5#6{%
⟨latexrelease⟩   \expandafter\ifx\csname #1+#2\endcsname\relax
⟨latexrelease⟩     \@latex@error{Font family ‘#1+#2’ unknown}\@eha
⟨latexrelease⟩   \else
⟨latexrelease⟩     \expandafter
⟨latexrelease⟩       \xdef\csname#1/#2/#3/#4\endcsname{\expandafter\noexpand
⟨latexrelease⟩                                 \csname #5\endcsname}%
⟨latexrelease⟩     \def\reserved@a{#6}%
⟨latexrelease⟩     \global
⟨latexrelease⟩     \expandafter\let\csname#5\expandafter\endcsname
⟨latexrelease⟩        \ifx\reserved@a\@empty
⟨latexrelease⟩          \@empty
⟨latexrelease⟩        \else
⟨latexrelease⟩          \reserved@a
⟨latexrelease⟩        \fi
⟨latexrelease⟩   \fi
⟨latexrelease⟩  }
⟨latexrelease⟩\EndIncludeInRelease
⟨∗2ekernel⟩
```

(*End of definition for* `\DeclareFontShape@`.)

`\DeclareFixedFont` Define a direct font switch that avoids all overhead.

```
\def\DeclareFixedFont#1#2#3#4#5#6{%
   \begingroup
      \math@fontsfalse
      \every@math@size{}%
      \fontsize{#6}\z@
      \usefont{#2}{#3}{#4}{#5}%
      \global\expandafter\let\expandafter#1\the\font
   \endgroup
  }
```

(*End of definition for* `\DeclareFixedFont`.)

`\do@subst@correction`

```
\def\do@subst@correction{%
       \xdef\subst@correction{%
          \font@name
          \global\expandafter\font
            \csname \curr@fontshape/\f@size\endcsname
            \noexpand\fontname\font
           \relax}%
```

Calling `\subst@correction` after the current group means calling it after we have loaded the substitution font which is done inside a group.

```
       \aftergroup\subst@correction
}
```

(*End of definition for* `\do@subst@correction`.)

\DeclareFontFamily

```
\def\DeclareFontFamily#1#2#3{%
```

If we want fast checking for the encoding scheme we can just check for \T@.. being defined.

```
% \@tempswafalse
% \def\reserved@b{#1}%
% \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
%      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
% \cdp@list
% \if@tempswa
 \@ifundefined{T@#1}%
    {%
     \@latex@error{Encoding scheme  `#1' unknown}\@eha
    }%
    {%
```

Now we have to define the macro \⟨#1⟩+⟨#2⟩ to contain #3. But since most of the time #3 will be empty we use \let in a tricky way rather than a simple \def since this will save internal memory. We store the argument #3 in a temporary macro \reserved@a.

```
      \def\reserved@a{#3}%
```

We compare \reserved@a with \@empty If these two are the same we \let the 'extra' macro equal to \@empty which is not the same a doing a \let to \reserved@a — the latter would blow one extra memory location rather then reusing the one from \@empty.

```
      \global
      \expandafter\let\csname #1+#2\expandafter\endcsname
             \ifx \reserved@a\@empty
               \@empty
             \else \reserved@a
             \fi
    }%
}
```

(*End of definition for* \DeclareFontFamily.)

\cdp@list We initialize the code page list to be empty.

```
\let\cdp@list\@empty
\@onlypreamble\cdp@list
```

(*End of definition for* \cdp@list.)

\cdp@elt

```
\let\cdp@elt\relax
\@onlypreamble\cdp@elt
```

(*End of definition for* \cdp@elt.)

\DeclareFontEncoding

```
\def\DeclareFontEncoding{%
```

First we start with ignoring all blanks and newlines since every surplus space in the second or third argument will come out in a weird place in the document.

```
  \begingroup
  \nfss@catcodes
  \expandafter\endgroup
  \DeclareFontEncoding@}
\@onlypreamble\DeclareFontEncoding
```

```
\def\DeclareFontEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
     \def\cdp@elt{\noexpand\cdp@elt}%
     \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                    {\default@family}{\default@series}%
                    {\default@shape}}%
```

To support encoding dependent commands (like accents) we initialise the command `\⟨encoding⟩-cmd` to be `\@changed@cmd`. (See `ltoutenc.dtx` for details.)

```
     \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
  \else
     \@font@info{Redeclaring font encoding #1}%
  \fi
```

```
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@M#3}%
```

Keep a record of the last encoding being declared:

```
  \xdef\LastDeclaredEncoding{#1}%
  }
\@onlypreamble\DeclareFontEncoding@
```

(*End of definition for* `\DeclareFontEncoding`.)

`\LastDeclaredEncoding` The last encoding being declared by `\DeclareFontEncoding`.

```
\def\LastDeclaredEncoding{}
```

(*End of definition for* `\LastDeclaredEncoding`.)

`\DeclareFontSubstitution`

```
\def\DeclareFontSubstitution#1#2#3#4{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
    \@latex@error{Encoding scheme  `#1' unknown}\@eha
  \else
    \begingroup
```

We loop through the `\cdp@list` and rebuild it anew in `\toks@` thereby replacing the defaults for the encoding in question with the new defaults. It is important to store the encoding to test against expanded in `\reserved@a` since it might just be `\LastDeclaredEncoding` that is passed as `#1`.

```
       \edef\reserved@a{#1}%
       \toks@{}%
       \def\cdp@elt##1##2##3##4{%
          \def\reserved@b{##1}%
          \ifx\reserved@a\reserved@b
```

Here we use the new defaults but we use `##1` (i.e., the encoding name already stored previously) since we know that it is expanded.

```
            \addto@hook\toks@{\cdp@elt{##1}{#2}{#3}{#4}}%
          \else
```

If `\reserved@a` and `\reserved@b` differ then we simply copy from the old list to the new.

```
            \addto@hook\toks@{\cdp@elt{##1}{##2}{##3}{##4}}%
          \fi}%
        \cdp@list
        \xdef\cdp@list{\the\toks@}%
    \endgroup
    \global
    \@namedef{D@#1}{%
           \def\default@family{#2}%
           \def\default@series{#3}%
           \def\default@shape{#4}%
           }%
  \fi
 }
\@onlypreamble\DeclareFontSubstitution
```

(*End of definition for* `\DeclareFontSubstitution`.)

`\DeclareFontEncodingDefaults`

```
\def\DeclareFontEncodingDefaults#1#2{%
  \ifx\relax#1\else
    \ifx\default@T\@empty\else
      \@font@info{Overwriting encoding scheme text defaults}%
    \fi
    \gdef\default@T{#1}%
  \fi
  \ifx\relax#2\else
    \ifx\default@M\@empty\else
      \@font@info{Overwriting encoding scheme math defaults}%
    \fi
    \gdef\default@M{#2}%
  \fi
}
\@onlypreamble\DeclareFontEncodingDefaults
```

(*End of definition for* `\DeclareFontEncodingDefaults`.)

`\default@T`
`\default@M`

```
\let\default@T\@empty
\let\default@M\@empty
```

(*End of definition for* `\default@T` *and* `\default@M`.)

`\DeclarePreloadSizes`

```
\def\DeclarePreloadSizes#1#2#3#4#5{%
 \@ifundefined{T@#1}%
   {\@latex@error{Encoding scheme  `#1' unknown}\@eha}%
   {%
```

Don't know at the moment what this group here does!

```
    \begingroup
```

We define a macro \reserved@f[32] that grabs the next *size* and loads the corresponding font. This is done by delimiting \reserved@f's only argument by the token , (comma).

```
      \def\reserved@f##1,{%
```

The end of the list will be detected when there are no more elements, i.e. when \reserved@f's argument is empty. The trick used here is explained in Appendix D of the TEXbook: if the argument is empty the \if will select the first clause and \let \reserved@f equal to \relax. (We use the > character here since it cannot appear in font file names.)

```
          \if>##1>%
            \let\reserved@f\relax
          \else
```

Otherwise, we define \font@name appropriately and call \pickup@font to do the work. Note that the requested \curr@fontshape combination must have been defined, or you will get an error. The definition of \font@name is carried out globally to be consistent with the rest of the code in this file.

```
            \xdef\font@name{\csname#1/#2/#3/#4/##1\endcsname}%
            \pickup@font
```

Now we forget the name of the font just loaded. More precisely, we set the corresponding control sequence to \relax. This means that later on, when the font is first used, the macro \define@newfont is called again to execute the 'extra' macro for this font.

```
            \global\expandafter\let\font@name\relax
          \fi
```

Finally we call \reserved@f again to process the next *size*. If \reserved@f was \let equal to \relax this will end the macro.

```
          \reserved@f}%
```

We finish with reinserting the list of sizes after the \reserved@f macro and appending an empty element so that the end of the list is recognized properly.

```
      \reserved@f#5,,%
    \endgroup
    }%
}
\@onlypreamble\DeclarePreloadSizes
```

(*End of definition for* \DeclarePreloadSizes.)

\ifmath@fonts We need a switch to decide if we have to switch math fonts. For this purpose we provide \ifmath@fonts that can be set to true or false by the \S@... macros depending on if math fonts are provided for this size or not. The default is of course to switch all fonts.

```
\newif\ifmath@fonts \math@fontstrue
```

(*End of definition for* \ifmath@fonts.)

[32] We cannot use \@tempa since it is needed in \pickup@font.

\DeclareMathSizes
\DeclareMathSizes*

\DeclareMathSizes takes the text size, math text size, math script size, and math scriptscript size as arguments and defines the right \S@... macro.

```
\def\DeclareMathSizes{%
  \@ifstar{\@DeclareMathSizes\math@fontsfalse}%
          {\@DeclareMathSizes{}}}
\@onlypreamble\DeclareMathSizes
```

(*End of definition for* \DeclareMathSizes *and* \DeclareMathSizes*.)

\@DeclareMathSizes

This modification by Michael J. Downes on comp.text.tex on 2002/10/17 allows the user to have settings such as
\DeclareMathSizes{9.5dd}{9.5dd}{7.4dd}{6.6dd}.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@DeclareMathSizes}%
⟨latexrelease⟩                 {Arbitrary units in \DeclareMathSizes}%
⟨*2ekernel | latexrelease⟩
\def\@DeclareMathSizes #1#2#3#4#5{%
  \@defaultunits\dimen@ #2pt\relax\@nnil
  \if $#3$%
    \expandafter\let\csname S@\strip@pt\dimen@\endcsname\math@fontsfalse
  \else
    \@defaultunits\dimen@ii #3pt\relax\@nnil
    \@defaultunits\@tempdima #4pt\relax\@nnil
    \@defaultunits\@tempdimb #5pt\relax\@nnil
    \toks@{#1}%
    \expandafter\xdef\csname S@\strip@pt\dimen@\endcsname{%
      \gdef\noexpand\tf@size{\strip@pt\dimen@ii}%
      \gdef\noexpand\sf@size{\strip@pt\@tempdima}%
      \gdef\noexpand\ssf@size{\strip@pt\@tempdimb}%
      \the\toks@
    }%
  \fi
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@DeclareMathSizes}%
⟨latexrelease⟩                 {Arbitrary units in \DeclareMathSizes}%
⟨latexrelease⟩\def\@DeclareMathSizes#1#2#3#4#5{%
⟨latexrelease⟩    \@defaultunits\dimen@#2pt\relax\@nnil
⟨latexrelease⟩    \if$#3$%
⟨latexrelease⟩      \expandafter \let
⟨latexrelease⟩        \csname S@\strip@pt\dimen@\endcsname
⟨latexrelease⟩        \math@fontsfalse
⟨latexrelease⟩    \else
⟨latexrelease⟩      \expandafter \gdef
⟨latexrelease⟩      \csname S@\strip@pt\dimen@\endcsname
⟨latexrelease⟩            {\gdef\tf@size{#3}\gdef\sf@size{#4}%
⟨latexrelease⟩                              \gdef\ssf@size{#5}%
⟨latexrelease⟩             #1%
⟨latexrelease⟩                              }%
⟨latexrelease⟩    \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\@DeclareMathSizes
```

(*End of definition for* `\@DeclareMathSizes`.)

3 Selecting a new font

3.1 Macros for the user

`\fontencoding` `\f@encoding`

As we said in the introduction a font is described by four parameters. We first define macros to specify the wanted *family*, *series*, or *shape*. These are simply recorded in internal macros `\f@family`, `\f@series`, and `\f@shape`, resp. We use `\edef`'s so that the arguments can also be macros.

```
\DeclareRobustCommand\fontencoding[1]{%
    \expandafter\ifx\csname T@#1\endcsname\relax
      \@latex@error{Encoding scheme `#1' unknown}\@eha
    \else
      \edef\f@encoding{#1}%
      \ifx\cf@encoding\f@encoding
```

If the new encoding is the same as the old encoding we have nothing to do. However, in case we had a sequence of several encoding changes without a `\selectfont` in-between we can save processing by making sure that `\enc@update` is `\relax`.

```
        \let\enc@update\relax
      \else
```

If current and new encoding differ we define the macro `\enc@update` to contain all updates necessary at `\selectfont` time.

```
        \let\enc@update\@@enc@update
      \fi
    \fi
}
```

(*End of definition for* `\fontencoding` *and* `\f@encoding`.)

`\@@enc@update`

```
\def\@@enc@update{%
```

When `\@@enc@update` is executed `\f@encoding` holds the encoding name for the new encoding and `\cf@encoding` the name of the last active encoding.

We start by setting the init command for encoding dependent macros to `\@changed@cmd`.

```
          \expandafter
          \let
            \csname\cf@encoding -cmd\endcsname
            \@changed@cmd
```

Then we turn the one for the new encoding to `\@current@cmd` (see `ltoutenc.dtx` for further explanations).

```
          \expandafter
          \let
            \csname\f@encoding-cmd\endcsname
            \@current@cmd
```

We execute the default settings `\default@T`, followed by the one for the new encoding.

```
        \default@T
        \csname T@\f@encoding\endcsname
```

Finally we change the default substitution values, disable `\enc@update` and make `\f@encoding` officially the current encoding.

```
        \csname D@\f@encoding\endcsname
        \let\enc@update\relax
        \let\cf@encoding\f@encoding
}
```

(*End of definition for* `\@@enc@update`.)

`\enc@update` The default action in `\selectfont` is to do nothing.

```
\let\enc@update\relax
```

(*End of definition for* `\enc@update`.)

`\fontfamily` `\f@family` `\fontseries` `\f@series` `\fontshape` `\f@shape`

```
\DeclareRobustCommand\fontfamily[1]{\edef\f@family{#1}}
```

There are now defined later (and differently).

```
%\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
%\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
```

(*End of definition for* `\fontfamily` *and others.*)

`\usefont` Some handy abbreviation if you want to get some particular font in the current size. If also the size should change one has to issue a `\fontsize` command first.

`\fontencoding` needs to do some setup work so we call that, but instead of calling `\fontfamily`, `\fontseries` and `\fontshape` it earlier versions of this code did, we now set `\f@family`, etc. directly. If we would call `\fontseries` or `\fontshape` as it was done in the past, they would now interact with the existing series and shape which is not desired if we intend to use an explicit font shape!

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                  {\usefont}{Force font face}%
\DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
   \edef\f@family{#2}%
   \set@target@series{#3}%
   \edef\f@shape{#4}%
```

Any earlier `\fontseries`, etc. should be canceled and we should switch unconditionally to the requested font face so we drop any code that may have been stored in `\delayed@f@adjustment`.

```
   \let\delayed@f@adjustment\@empty
   \selectfont
   \ignorespaces}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {\usefont}{Drop m in usefont}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
⟨latexrelease⟩   \edef\f@family{#2}%
⟨latexrelease⟩   \set@target@series{#3}%
⟨latexrelease⟩   \edef\f@shape{#4}\selectfont
```

```
⟨latexrelease⟩    \ignorespaces}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\usefont}{Drop m in usefont}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
⟨latexrelease⟩    \edef\f@family{#2}%
⟨latexrelease⟩    \edef\f@series{#3}%
⟨latexrelease⟩    \edef\f@shape{#4}\selectfont
⟨latexrelease⟩    \ignorespaces}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\usefont`.)

`\linespread` The command `\linespread` changes the current `\baselinestretch` by calling `\set@fontsize`. The values for `\f@size` and `\f@baselineskip` will be left unchanged.

```
\DeclareRobustCommand\linespread[1]
    {\set@fontsize{#1}\f@size\f@baselineskip}
```

(*End of definition for* `\linespread`.)

`\fontsize` We also define a macro that allows to specify a size. In this case, however, we also need the value of `\baselineskip`. As the first argument to `\set@fontsize` we pass the current value of `\baselinestretch`. This will either match the internal value (in which case nothing changes) or it will be an updated value due to a user change of that macro using `\renewcommand`. If we would pass the internal `\f@linespread` such a change would be effectively overwritten by a size change.

```
\DeclareRobustCommand\fontsize[2]
    {\set@fontsize\baselinestretch{#1}{#2}}
```

(*End of definition for* `\fontsize`.)

`\f@linespread` This macro holds the current internal value for `\baselinestretch`.

```
\let\f@family\@empty
\let\f@series\@empty
\let\f@shape\@empty
\let\f@size\@empty
\let\f@baselineskip\@empty
\let\f@linespread\@empty
```

(*End of definition for* `\f@linespread`.)

`\cf@encoding`

```
\let\f@encoding\@empty
\let\cf@encoding\@empty
```

(*End of definition for* `\cf@encoding`.)

\@defaultunits The function \@defaultunits when wrapped around a dimen or skip assignment supplies default units. Usage:

\@defaultunits\dimen@=#1pt\relax\@nnil

Note: the \relax is *important*. Other units can be substituted for the 'pt' if desired.

We use \remove@to@nnil as an auxiliary macros for \@defaultunits. It just has to gobble the supplied default unit 'pt' or whatever, if it wasn't used in the assignment.

```
\def\@defaultunits{\afterassignment\remove@to@nnil}
```

(*End of definition for* \@defaultunits.)

\strip@pt This macro strips the characters pt produced by using \the on a dimen register.
\rem@pt

```
\begingroup
  \catcode`P=12
  \catcode`T=12
  \lowercase{
    \def\x{\def\rem@pt##1.##2PT{##1\ifnum##2>\z@.##2\fi}}}
  \expandafter\endgroup\x
\def\strip@pt{\expandafter\rem@pt\the}
```

(*End of definition for* \strip@pt *and* \rem@pt.)

\mathversion \mathversion takes the math *version* name as argument, defines \math@version appropriately and switches to the font selected forcing a call to \glb@settings if the *version* is known to the system.
\math@version

```
\DeclareRobustCommand\mathversion[1]
         {\@nomath\mathversion
          \expandafter\ifx\csname mv@#1\endcsname\relax
          \@latex@error{Math version `#1' is not defined}\@eha\else
```

If there has been a frozen math version reset locally. See GH 1028.

```
          \ifcsname mv@\math@version @frozen\endcsname
            \expandafter\let
            \csname mv@\math@version @frozen\expandafter\endcsname
            \csname mv@\math@version\endcsname
          \fi
```

```
          \edef\math@version{#1}%
```

We need to force a math font setup both now and at the point where we return to the previous math version. Forcing a math font setup can simply be done by setting \glb@currsize to an invalid value since this will trigger the setup when the formula starts.

```
          \gdef\glb@currsize{}%
```

When the scope of the current \mathversion ends we need to restore the old setup. However this time we need to force it directly at least if we are inside math, otherwise we could wait. Another way to enhance this code here is todo the setting only if the version really has changed after all. This might be interesting in case of amstext and boldsymbol.

```
          \aftergroup\glb@settings
          \fi}
```

(*End of definition for* `\mathversion` *and* `\math@version`.)

If TEX would support a hook just before the end of a formula (opposite of `\everymath` so to speak) the implementation of the algorithm would be much simpler because in that case we would set up the correct math fonts at this point without having to worry about incorrect settings due to nesting. The same would be true if in LATEX the use of `$` (as the primitive TEX command) would be impossible and instead only a higher-level interface would be available. Note that this does not mean that a `$` couldn't be the short-hand for starting and stopping that higher-level interface, it only means that the direct TEX function must be hidden.

Anyway, since we don't have this and won't have it in LATEX 2_ε we need to implement it in a somewhat slower way.

We test for the current math font setup on entry of a formula, i.e., on the hooks `\everymath` and `\everydisplay`. But since these hooks may contain user data we provide ourselves with an internal version of these hooks which stays frozen.

`\frozen@everymath`
`\frozen@everydisplay`

New internal names for `\everymath` and `\everydisplay`.

```
\let\frozen@everymath\everymath
\let\frozen@everydisplay\everydisplay
```

(*End of definition for* `\frozen@everymath` *and* `\frozen@everydisplay`.)

`\everymath`
`\everydisplay`

Now we provide now user hooks that will be called in the frozen internals.

```
\newtoks\everymath
\newtoks\everydisplay
```

(*End of definition for* `\everymath` *and* `\everydisplay`.)

`\frozen@everydisplay`

Now we define the behaviour of the frozen hooks: first check the math setup then call the user hook.

The check code may push tokens after the math formula with `\aftergroup` and they would prevent a `$$` from dropping following spaces. We therefore use a switch to be set as the first thing after the group so that following code can determine if there was a display or some inline math (in the latter case we better not drop spaces). After setting the switch we also have to place `\ignorespaces` because setting the switch may be the only thing that happens after the display. The issue with handling of spaces was found in 2022, but it is really a bug fix for the code added in 2021/11.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}
⟨latexrelease⟩  {\frozen@everydisplay}{Handle spaces after math}%
⟨*2ekernel | latexrelease⟩
\frozen@everydisplay = {%
  \aftergroup\@ignoretrue  \aftergroup\ignorespaces
  \check@mathfonts
  \the\everydisplay}
```

(*End of definition for* `\frozen@everydisplay`.)

`\frozen@everymath`

The frozen code for inline math is similar, except that here we do not want to drop following spaces.

```
\frozen@everymath = {%
  \aftergroup\@ignorefalse
  \check@mathfonts
  \the\everymath}
```

(*End of definition for* `\frozen@everymath`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}
⟨latexrelease⟩  {\frozen@everydisplay}{Handle spaces after math}%
⟨latexrelease⟩
⟨latexrelease⟩\frozen@everydisplay = {\check@mathfonts
⟨latexrelease⟩                        \the\everydisplay}
⟨latexrelease⟩\frozen@everymath = {\check@mathfonts
⟨latexrelease⟩                     \the\everymath}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\curr@math@size` This holds locally the current math size.

```
\let\curr@math@size\@empty
```

(*End of definition for* `\curr@math@size`.)

3.2 Macros for loading fonts

`\pickup@font` The macro `\pickup@font` which is used in `\selectfont` is very simple: if the font name is undefined (i.e. not known yet) it calls `\define@newfont` to load it.

```
\def\pickup@font{%
    \expandafter \ifx \font@name \relax
       \define@newfont
    \fi}
```

(*End of definition for* `\pickup@font`.)

`\split@name` `\pickup@font` assumes that `\font@name` is set but it is sometimes called when `\f@family`, `\f@series`, `\f@shape`, or `\f@size` may have the wrong settings (see, e.g., the definition of `\getanddefine@fonts`). Therefore we need a macro to extract font *family*, *series*, *shape*, and *size* from the font name. To this end we define `\split@name` which takes the font name as a list of characters of `\catcode` 12 (without the backslash at the beginning) delimited by the special control sequence `\@nil`. This is not very complicated: we first ensure that `/` has the right `\catcode`

```
{\catcode`\/=12
```

and define `\split@name` so that it will define our private `\f@encoding`, `\f@family`, `\f@series`, `\f@shape`, and `\f@size` macros.

```
\gdef\split@name#1/#2/#3/#4/#5\@nil{\def\f@encoding{#1}%
                                    \def\f@family{#2}%
                                    \def\f@series{#3}%
                                    \def\f@shape{#4}%
                                    \def\f@size{#5}}}
```

(*End of definition for* `\split@name`.)

`\curr@fontshape` Abbreviation which may get removed again for speed.

```
\def\curr@fontshape{\f@encoding/\f@family/\f@series/\f@shape}
```

(*End of definition for* `\curr@fontshape`.)

\define@newfont Now we can tackle the problem of defining a new font.

```
\def\define@newfont{%
```

We have already mentioned that the token list that \split@name will get as argument must not start with a backslash. To reach this goal we will set the \escapechar to −1 so that the \string primitive will not generate an escape character. To keep this change local we open a group. We use \begingroup for this purpose since \define@newfont might be called in math mode, and an empty \bgroup... \egroup would add an empty Ord atom to the math list and thus affect the spacing.

Also locally redefine \typeout so that 'No file ...fd' Warnings become Font Info message just sent to the log file.

```
  \begingroup
    \let\typeout\@font@info
    \escapechar\m@ne
```

Then we extract *encoding scheme*, *family*, *series*, *shape*, and *size* from the font name. Note the four \expandafter's so that \font@name is expanded first, then \string, and finally \split@name.

```
    \expandafter\expandafter\expandafter
       \split@name\expandafter\string\font@name\@nil
```

If the \curr@fontshape combination is not available, (i.e. undefined) we call the macro \wrong@fontshape to take care of this case. Otherwise \extract@font will load the external font for us.

```
%    \expandafter\ifx
%        \csname\curr@fontshape\endcsname \relax
      \try@load@fontshape % try always
%    \fi
    \expandafter\ifx
       \csname\curr@fontshape\endcsname \relax
      \wrong@fontshape\else
```

To allow substitution we call the curr@fontshape macro which usually will expand to \relax but may hold code for substitution (see \subst@fontshape definition).

```
%      \csname\curr@fontshape\endcsname
      \extract@font\fi
```

We are nearly finished and must only restore the \escapechar by closing the group.

```
  \endgroup}
```

```
\def\try@load@fontshape{%
   \expandafter
   \ifx\csname \f@encoding+\f@family\endcsname\relax
     \@font@info{Trying to load font information for
                    \f@encoding+\f@family}%
```

We predefine this combination to be \@empty which means that next time we don't try again unnecessary in case we don't find a .fd file. If the file contains a \DeclareFontFamily command than this setting will be overwritten.

```
    \global\expandafter\let
       \csname\f@encoding+\f@family\endcsname\@empty
```

Set the catcodes used in the syntax, but do it only once (this will be restored at the end of the font loading group).

```
     \nfss@catcodes
     \let\nfss@catcodes\relax
```

For increased portability make the external filename monocase, but look for the (old style) mixed case filename if the first attempt fails.

On any monocase system this means that the file is looked for twice which takes up time and string space, but at least for this release Check for both names to give people time to re-install their private fd files with lowercase names.

```
    \edef\reserved@a{%
      \lowercase{%
        \noexpand\InputIfFileExists{\f@encoding\f@family.fd}}}%
    \reserved@a\relax
         {\@input@{\f@encoding\f@family.fd}}%
  \fi}
```

(*End of definition for* `\define@newfont`.)

`\nfss@catcodes` This macro should contain the standard `\catcode` assignments to all characters which are used in the commands found in an `.fd` file and which might have special `\catcode`s in the middle of a document. If necessary, this list can be extended in a package file using a suitable number of `\expandafter`, i.e.,

```
\expandafter\def\expandafter\nfss@catcodes
      \expandafter{\nfss@catcodes <additional settings>}
```

Note, that this macro might get executed several times since it is also called by `\DeclareFontShape`, thus it probably should not be misused as a general purpose hook.

```
\def\nfss@catcodes{%
```

We start by making `@` a letter and ignoring all blanks and newlines.

```
    \makeatletter
    \catcode`\ 9%
    \catcode`\^^I9%
    \catcode`\^^M9%
```

Then we set up `\`, `{`, `}`, `#` and `%` in case an `.fd` file is loaded during a verbatim environment.

```
    \catcode`\\\z@
    \catcode`\{\@ne
    \catcode`\}\tw@
    \catcode`\#6%
    \catcode`\^7%
    \catcode`\%14%
```

The we make sure that the important syntax parts have the right `\catcode`.

```
  \@makeother\<%
  \@makeother\>%
  \@makeother\*%
  \@makeother\.%
  \@makeother\-%
  \@makeother\/%
  \@makeother\[%
  \@makeother\]%
  \@makeother\`%
  \@makeother\'%
  \@makeother\"%
}
```

(*End of definition for* `\nfss@catcodes`.)

`\LoadFontDefinitionFile` Load and `.fd` files for some encoding and family (if it exists).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {\LoadFontDefinitionFile}{Loading .fd files}%
\def\LoadFontDefinitionFile#1#2{%
  \begingroup
    \edef\f@encoding{#1}%
    \edef\f@family{#2}%
    \try@load@fontshape
  \endgroup
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\LoadFontDefinitionFile}{Loading .fd files}%
⟨latexrelease⟩
⟨latexrelease⟩\let\LoadFontDefinitionFile\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\LoadFontDefinitionFile`.)

`\DeclareFontFamilySubstitution` The idea for this macro is stolen from the `substitutefont` package by Günter Milde, with some modifications and a new name.

Its purpose is to provide characters in a special encoding that are not available in the current font family to be taken from a different family that is visually compatible (or not if you choose badly). For example, you can match the GFS Didot Greek characters with TEX Gyre Pagella (Palatino) by specifying

```
\DeclareFontFamilySubstitution{LGR}{qpl}{udidot}
```

This way if you ask for the `LGR` encoding in for the `qpl` family you get the characters from the `udidot` family substituted.

We need to ensure that the macro is defined with `\nfss@catcodes` in force (not quite sure why at the moment to be honest).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩        {\DeclareFontFamilySubstitution}{Provide family substitution}%
\begingroup
\nfss@catcodes
\gdef\DeclareFontFamilySubstitution#1#2#3{%
```

We only provide a set of silent substitutions. The package also (re)declared the family, but this is incorrect in my eyes and it is better to handle that differently.

Of course the families may still need loading at this point and so we arrange for this. Otherwise we might run into trouble because the necessary `\DeclareFontFamily` has not been seen.

```
  \LoadFontDefinitionFile{#1}{#2}%
  \LoadFontDefinitionFile{#1}{#3}%
```

```
    \DeclareFontShape{#1}{#2}{m}{it}{<->ssub * #3/m/it}{}%
    \DeclareFontShape{#1}{#2}{m}{n}{<->ssub * #3/m/n}{}%
    \DeclareFontShape{#1}{#2}{m}{sc}{<->ssub * #3/m/sc}{}%
    \DeclareFontShape{#1}{#2}{m}{sl}{<->ssub * #3/m/sl}{}%
```

These days a few more shapes might be around, so we declare those too. If they don't exist then after the first substitution normal fallbacks will happen.

```
    \DeclareFontShape{#1}{#2}{m}{sw}{<->ssub * #3/m/sw}{}%
    \DeclareFontShape{#1}{#2}{m}{scit}{<->ssub * #3/m/scit}{}%
    \DeclareFontShape{#1}{#2}{m}{scsl}{<->ssub * #3/m/scsl}{}%
```

Same game with `b` and `bx`, for other weights you are on your own:

```
    \DeclareFontShape{#1}{#2}{b}{it}{<->ssub * #3/b/it}{}%
    \DeclareFontShape{#1}{#2}{b}{n}{<->ssub * #3/b/n}{}%
    \DeclareFontShape{#1}{#2}{b}{scit}{<->ssub * #3/b/scit}{}%
    \DeclareFontShape{#1}{#2}{b}{scsl}{<->ssub * #3/b/scsl}{}%
    \DeclareFontShape{#1}{#2}{b}{sc}{<->ssub * #3/b/sc}{}%
    \DeclareFontShape{#1}{#2}{b}{sl}{<->ssub * #3/b/sl}{}%
    \DeclareFontShape{#1}{#2}{b}{sw}{<->ssub * #3/b/sw}{}%
    \DeclareFontShape{#1}{#2}{bx}{it}{<->ssub * #3/bx/it}{}%
    \DeclareFontShape{#1}{#2}{bx}{n}{<->ssub * #3/bx/n}{}%
    \DeclareFontShape{#1}{#2}{bx}{scit}{<->ssub * #3/bx/scit}{}%
    \DeclareFontShape{#1}{#2}{bx}{scsl}{<->ssub * #3/bx/scsl}{}%
    \DeclareFontShape{#1}{#2}{bx}{sc}{<->ssub * #3/bx/sc}{}%
    \DeclareFontShape{#1}{#2}{bx}{sl}{<->ssub * #3/bx/sl}{}%
    \DeclareFontShape{#1}{#2}{bx}{sw}{<->ssub * #3/bx/sw}{}%
}
\endgroup
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩      {\DeclareFontFamilySubstitution}{Provide family substitution}%
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareFontFamilySubstitution\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\DeclareFontFamilySubstitution`.)

`\DeclareErrorFont` Declare the last resort shape! We assume that in this fontshape there is a 10pt font but it doesn't really matter. We only loose one macro name if the assumption is false. But at least the font should be there!

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\DeclareErrorFont}{No side effects please}%
\def\DeclareErrorFont#1#2#3#4#5{%
      \xdef\error@fontshape{%
          \noexpand\expandafter\noexpand\split@name\noexpand\string
          \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
          \noexpand\@nil}%
```

Initialize all those internal variables which may or may not have values in the first seconds of NFSS' bootstrapping process. Later on such values will be updated when an encoding is selected, etc.

We definitely don't want to set `\f@encoding`; we can set all the others since if they are left "blank" any selection would grab "error default values" as well. However, this probably should go also—and now it did.

```
%       \gdef\f@encoding{#1}%
      \gdef\default@family{#2}%
      \gdef\default@series{#3}%
      \gdef\default@shape{#4}%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\DeclareErrorFont}{No side effects please}%
⟨latexrelease⟩
⟨latexrelease⟩\def\DeclareErrorFont#1#2#3#4#5{%
⟨latexrelease⟩      \xdef\error@fontshape{%
⟨latexrelease⟩          \noexpand\expandafter\noexpand\split@name\noexpand\string
⟨latexrelease⟩          \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
⟨latexrelease⟩          \noexpand\@nil}%
⟨latexrelease⟩      \gdef\default@family{#2}%
⟨latexrelease⟩      \gdef\default@series{#3}%
⟨latexrelease⟩      \gdef\default@shape{#4}%
⟨latexrelease⟩      \global\let\f@family\default@family
⟨latexrelease⟩      \global\let\f@series\default@series
⟨latexrelease⟩      \global\let\f@shape\default@shape
⟨latexrelease⟩      \gdef\f@size{#5}%
⟨latexrelease⟩      \gdef\f@baselineskip{#5pt}%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\DeclareErrorFont
```

(*End of definition for* `\DeclareErrorFont`.)

`\wrong@fontshape` Before we come to the macro `\extract@font` we have to take care of unknown `\curr@fontshape` combinations. The general strategy is to issue a warning and to try a default *shape*, then a default *series*, and finally a default *family*. If this last one also fails TEX will go into an infinite loop. But if the defaults are set incorrectly one deserves nothing else!

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\wrong@fontshape}%
⟨latexrelease⟩                 {Font substitution in preamble}%
⟨*2ekernel | latexrelease⟩
\def\wrong@fontshape{%
    \csname D@\f@encoding\endcsname   % install defaults if in math
```

We remember the wanted `\curr@fontshape` combination which we will need in a moment.

```
    \edef\reserved@a{\csname\curr@fontshape\endcsname}%
  \ifx\last@fontshape\reserved@a
     \errmessage{Corrupted NFSS tables}%
     \error@fontshape
  \else
```

Then we warn the user about the mess and set the shape to its default.

```
    \let\f@shape\default@shape
```

If the combination is not known, try the default *series*.

```
    \expandafter\ifx\csname\curr@fontshape\endcsname\relax
       \let\f@series\default@series
```

If this is still undefined, try the default *family*. Otherwise give up. We never try to change the encoding scheme!

```
         \expandafter
           \ifx\csname\curr@fontshape\endcsname\relax
            \let\f@family\default@family
```

If we change the font family and we are in the preamble then the corresponding `.fd` file may not been loaded yet. Therefore we try this now. Otherwise equating the requested font shape with the finally selected fontshape below will fail and can result in "NFSS tables corrupted". After begin document that will not happen as all `.fd` files involved in substitution are loaded at `\begin{document}`.

```
            \begingroup
               \try@load@fontshape
            \endgroup
         \fi \fi
  \fi
```

At this point a valid `\curr@fontshape` combination must have been found. We inform the user about this fact.

The `\expandafter\string` here stops TEX adding the space that it usually puts after command names in messages. The similar construction with `\@undefined` just produces 'undefined', but saves a few tokens.

`\@wrong@font@char` is locally redefined in `\UseTextSymbol` from its normal (empty) definition, to report the symbol generating the font switch.

```
     \@font@warning{Font shape `\expandafter\string\reserved@a'
                      \expandafter\@gobble\string\@undefined\MessageBreak
                    using `\curr@fontshape' instead\@wrong@font@char}%
    \global\let\last@fontshape\reserved@a
```

We change `\@defaultsubs` to produce a warning at the end of the document. The macro `\@defaultsubs` is initially `\relax` but gets changed here if some default font substitution happens. It is then executed in `\enddocument`.

```
    \gdef\@defaultsubs{%
      \@font@warning{Some font shapes were not available, defaults
                       substituted.\@gobbletwo}}%
```

If we substitute a `\curr@fontshape` combination by the default one we don't want the warning to be printed out whenever this (unknown) combination is used. Therefore we globally `\let` the macro corresponding to the wanted combination equal to its substitution. This requires the use of four `\expandafter`'s since `\csname`...`\endcsname` has to be expanded before `\reserved@a` (i.e. the requested combination), and this must happen before the `\let` is executed.

```
    \global\expandafter\expandafter\expandafter\let
       \expandafter\reserved@a
           \csname\curr@fontshape\endcsname
```

Now we can redefine `\font@name` accordingly. This *must* be done globally since it might occur in the group opened by `\define@newfont`. If we would this definition were local the closing `\endgroup` there would restore the old meaning of `\font@name` and then switch to the wrong font at the end of `\selectfont` although the correct font was loaded.

```
    \xdef\font@name{%
      \csname\curr@fontshape/\f@size\endcsname}%
```

The last thing this macro does is to call `\pickup@font` again to load the font if it is not defined yet. At this point this code will loop endlessly if the defaults are not well defined.

```
    \pickup@font}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\wrong@fontshape}%
⟨latexrelease⟩                 {Font substitution in preamble}%
⟨latexrelease⟩\def\wrong@fontshape{%
⟨latexrelease⟩    \csname D@\f@encoding\endcsname
⟨latexrelease⟩    \edef\reserved@a{\csname\curr@fontshape\endcsname}%
⟨latexrelease⟩  \ifx\last@fontshape\reserved@a
⟨latexrelease⟩     \errmessage{Corrupted NFSS tables}%
⟨latexrelease⟩     \error@fontshape
⟨latexrelease⟩  \else
⟨latexrelease⟩    \let\f@shape\default@shape
⟨latexrelease⟩    \expandafter\ifx\csname\curr@fontshape\endcsname\relax
⟨latexrelease⟩       \let\f@series\default@series
⟨latexrelease⟩        \expandafter
⟨latexrelease⟩          \ifx\csname\curr@fontshape\endcsname\relax
⟨latexrelease⟩           \let\f@family\default@family
⟨latexrelease⟩        \fi \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩     \@font@warning{Font shape
⟨latexrelease⟩            `\expandafter\string\reserved@a'
⟨latexrelease⟩            \expandafter\@gobble\string\@undefined
⟨latexrelease⟩            \MessageBreak
⟨latexrelease⟩            using `\curr@fontshape' instead\@wrong@font@char}%
⟨latexrelease⟩    \global\let\last@fontshape\reserved@a
⟨latexrelease⟩    \gdef\@defaultsubs{%
⟨latexrelease⟩      \@font@warning{Some font shapes were not available,
⟨latexrelease⟩                     defaults substituted.\@gobbletwo}}%
⟨latexrelease⟩    \global\expandafter\expandafter\expandafter\let
⟨latexrelease⟩       \expandafter\reserved@a
⟨latexrelease⟩           \csname\curr@fontshape\endcsname
⟨latexrelease⟩    \xdef\font@name{%
⟨latexrelease⟩      \csname\curr@fontshape/\f@size\endcsname}%
⟨latexrelease⟩    \pickup@font}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\wrong@fontshape`.)

`\@wrong@font@char` Normally empty but redefined in `\UseTextSymbol` so that the Font shape undefined message can refer to the symbol causing the problem.

```
\let\@wrong@font@char\@empty
```

(*End of definition for* `\@wrong@font@char`.)

`\@@defaultsubs` `\@defaultsubs` See above.

```
\let\@defaultsubs\relax
```

(*End of definition for* `\@@defaultsubs` *and* `\@defaultsubs`.)

\strip@prefix In \extract@font we will need a way to recover the replacement text of a macro. This is done by the primitive \meaning together with the macro \strip@prefix (for the details see appendix D of the TEXbook, p. 382).

```
\def\strip@prefix#1>{}
```

(*End of definition for* \strip@prefix.)

4 Assigning math fonts to *versions*

\install@mathalphabet This is just another name for \gdef but we can redefine it if necessary later on.

```
\let\install@mathalphabet\gdef
```

(*End of definition for* \install@mathalphabet.)

\math@fonts

```
\let\math@fonts\@empty
```

(*End of definition for* \math@fonts.)

\select@group \select@group has four arguments: the new ⟨*math alphabet identifier*⟩ (a control sequence), the ⟨*math group number*⟩, the extra macro for math mode and the \curr@fontshape definition macro name. We first check if we are in math mode.

```
%\def\select@group#1#2#3{\relax\ifmmode
```

We do these things locally using \begingroup instead of \bgroup to avoid the appearance of an empty Ord atom on the math list.

```
%  \begingroup
```

We set the math fonts for the *family* in question by calling \getanddefine@fonts in the correct environment.

```
%      \escapechar\m@ne
%    \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
```

We globally select the math fonts...

```
%    \globaldefs\@ne  \math@fonts
```

... and close the group to restore \globaldefs and \escapechar.

```
%  \endgroup
```

As long as no *size* or *version* change occurs the ⟨*math alphabet identifier*⟩ should simply switch to the installed *math group* instead of calling \select@group unnecessarily. So we globally redefine the first argument (the new ⟨*math alphabet identifier*⟩) to expand into a \mathgroup switch and then select this *alphabet*. Note that this redefinition will be overwritten by the next call to a *version* macro. The original code for the end of \select@group was

```
\gdef#1{#3\mathgroup #2}#1\fi}
```

i.e. first redefining the ⟨*math alphabet identifier*⟩ and then calling the new definition to switch to the wanted ⟨*math group*⟩. Now we define the ⟨*math alphabet identifier*⟩ as a call to the \use@mathgroup command.

```
%  \xdef#1{\noexpand\use@mathgroup\noexpand#2%
%          {\number\csname c@mv@\math@version\endcsname}}%
```

But this is not sufficient, as we learned the hard way. The problem here is that the loading of the fonts that comprise the alphabet identifier #1, as well as the necessary math font assignments is deferred until it is used. This is OK so far, but if the fonts are switched within the current formula (which may happen if a sub-formula is a box that contains a math version switch) the font assignments for #1 are not restored unless #1 is used again. This is disastrous since TeX sees the wrong fonts at the end of the math formula, when it converts the math list into a horizontal list.

This is taken into account as follows: When a math alphabet identifier is used for the first time in a certain version it modifies the corresponding macro \mv@⟨*version*⟩ so that it calls \getanddefine@fonts directly in future as well. We use the macro \extract@alph@from@version to do this. It takes the math alphabet identifier #1 and the math version macro as arguments.

```
%     \expandafter\extract@alph@from@version
%        \csname mv@\math@version\expandafter\endcsname
%       \expandafter{\number\csname c@mv@\math@version\endcsname}%
%       #1%
%     \stepcounter{mv@\math@version}%
```

Finally, it is not possible to simply call the new definition since we have an argument (the third argument of \use@mathgroup or more exactly the argument of \math@egroup if the margid option is in force) which would swallow our closing \fi. So we use the \expandafter technique to remove the \fi before the \use@mathgroup is expanded.

```
%\expandafter #1\fi}
```

(*End of definition for* \select@group.)

\extract@alph@from@version

We proceed to the definition of the macro \extract@alph@from@version. As stated above, it takes a math alphabet identifier and a math version macro (e.g. \mv@normal) as its arguments.

```
\def\extract@alph@from@version#1#2#3{%
```

To extract and replace the definition of math alphabet identifier #3 in macro #1 we have to recall how this definition looks like: Somewhere in the replacement text of #1 there is the sequence

\install@mathalphabet⟨*math alphabet identifier*⟩ #3{%
⟨*Definitions for* ⟩#3}

Hence, the first thing we do is to extract the tokens preceding this definitions, the definition itself, and the tokens following it. To this end we define one auxiliary macro \reserved@a.

```
    \def\reserved@a##1\install@mathalphabet#3##2##3\@nil{%
```

When \reserved@a is expanded, it will have the tokens preceding the definition in question in its first argument (##1), the following tokens in its third argument (##3), and the replacement text for the math alphabet identifier #3 in its second argument. (##2). This is then recorded for later use in a temporary macro \reserved@b.

```
        \def\reserved@b{##2}%
```

Additionally, we define a macro \reserved@c to reconstruct the definitions for the math version in question from the tokens that will remain unchanged (##1 and ##3) and the yet to build new definitions for the math alphabet identifier #3.

```
        \def\reserved@c####1{\gdef#1{##1####1##3}}}%
```

Then we execute our auxiliary macro.

```
632     \expandafter\reserved@a#1\@nil
```

OK, so now we have to build the new definition for #3. To do so, we first extract the interesting parts out of the old one. The old definition looks like:

\select@group⟨*math alphabet identifier*⟩
⟨*math group number*⟩⟨*math extra part*⟩
⟨curr@fontshape *definition*⟩

So we define a new temporary macro \reserved@a that extracts these parts.

```
633     \def\reserved@a\select@group#3##1##2\@nil{%
```

This macro can now directly rebuild the math version definition by calling \reserved@c:

```
634        \reserved@c{%
635           \getanddefine@fonts{#2}##2%
636           \install@mathalphabet#3{%
637              \relax\ifmmode \else \non@alpherr#3\fi
638              \use@mathgroup##1{#2}}}%
```

In addition it defines the alphabet the way it should be used from now on.

```
639       \gdef#3{\relax\ifmmode \else \non@alpherr#3\fi
640               \use@mathgroup##1{#2}}}%
```

Finally, we only have to call this macro \reserved@a on the old definitions recorded in \reserved@b:

```
641     \expandafter\reserved@a\reserved@b\@nil
642     }
```

(*End of definition for* \extract@alph@from@version.)

\math@bgroup
\math@egroup

Here are the default definitions for \math@bgroup and \math@egroup. We use \bgroup instead of \begingroup to avoid 'leaking out' of style changes. This has the side effect of always producing mathord atoms.

```
643 \let\math@bgroup\bgroup
644 \def\math@egroup#1{#1\egroup}
```

(*End of definition for* \math@bgroup *and* \math@egroup.)

\calculate@math@sizes

Here is the default definition for \calculate@math@sizes a more elaborate interface is under testing in mthscale.sty.

```
645 \gdef\calculate@math@sizes{%
646   \@font@info{Calculating\space math\space sizes\space for\space
647               size\space <\f@size>}%
648   \dimen@\f@size \p@
649   \@tempdimb \defaultscriptratio \dimen@
650   \dimen@ \defaultscriptscriptratio \dimen@
651   \expandafter\xdef\csname S@\f@size\endcsname{%
652     \gdef\noexpand\tf@size{\f@size}%
653     \gdef\noexpand\sf@size{\strip@pt\@tempdimb}%
654     \gdef\noexpand\ssf@size{\strip@pt\dimen@}%
655     \noexpand\math@fontstrue}}
```

(*End of definition for* \calculate@math@sizes.)

\defaultscriptratio
\defaultscriptscriptratio

The default ratio for math sizes is:
1 to \defaultscriptratio to \defaultscriptscriptratio.
By default this is 1 to .7 to .5.

```
\def\defaultscriptratio{.7}
\def\defaultscriptscriptratio{.5}
```

(*End of definition for* \defaultscriptratio *and* \defaultscriptscriptratio.)

\noaccents@

If we don't have a definition for \noaccents@ we provide a dummy.

```
\ifx\noaccents@\@undefined
  \let\noaccents@\@empty
\fi
```

(*End of definition for* \noaccents@.)

\showhyphens

The \showhyphens command must be redefined since the version in plain.tex uses \tenrm. We have also made some further adjustments for its use in LaTeX.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/01/01}{\showhyphens}%
⟨latexrelease⟩                  {XeTeX support for \showhyphens}%
⟨*2ekernel | latexrelease⟩
\ifx\XeTeXcharclass\@undefined
```

Version for engines other than XeTeX.

```
\DeclareRobustCommand\showhyphens[1]{%
  \setbox0\vbox{%
    \color@begingroup
    \everypar{}%
    \parfillskip\z@skip\hsize\maxdimen
    \normalfont
    \pretolerance\m@ne\tolerance\m@ne\hbadness\z@\showboxdepth\z@\ #1%
    \color@endgroup}}
```

```
\else
```

XeTeX version. When using system fonts XeTeX reports consecutive runs of characters as a single item in box logging, which means the standard \showhyphens does not work. This version typesets the text into a narrow box to force hyphenation and then reconstructs a horizontal list with explicit hyphens to generate the display. Note that the lmr OpenType font is forced, this works even if the characters are not in the font as hyphenation is attempted due to the width of the space and hyphen character. It would generate spurious Missing Character warnings in the log, these are however suppressed from the terminal and log output by ensuring that \tracinglostchars is locally zero.

```
\DeclareRobustCommand\showhyphens[1]{%
  \setbox0\vbox{%
    \usefont{TU}{lmr}{m}{n}%
    \hsize 1sp %
    \hbadness\@M
    \hfuzz\maxdimen
    \tracingonline\z@
    \tracinglostchars\z@
    \everypar={}%
    \leftskip\z@skip
    \rightskip\z@skip
```

```
    \parfillskip\z@skip
    \hyphenpenalty=-\@M
    \pretolerance\m@ne
    \interlinepenalty\z@
    \clubpenalty\z@
    \widowpenalty\z@
    \brokenpenalty1127 %
    \setbox\z@\hbox{}%
    \noindent
    \hskip\z@skip
    #1%
    \par
```

Note here we stop the loop if made no progress, non-removable items may mean that we can not process the whole list (which would be testable as `\lastnodetype=-1`).

```
      \loop
      \@tempswafalse
      \ifnum\lastnodetype=11\unskip\@tempswatrue\fi
      \ifnum\lastnodetype=12\unkern\@tempswatrue\fi
      \ifnum\lastnodetype=13 %
       \count@\lastpenalty
       \unpenalty\@tempswatrue
    \fi
    \ifnum\lastnodetype=\@ne
      \setbox\tw@\lastbox\@tempswatrue
      \setbox0\hbox{\unhbox\tw@\unskip\unskip\unpenalty
                    \ifnum\count@=1127 \else\ \fi
                    \unhbox0}%
      \count@\z@
    \fi
    \if@tempswa
    \repeat
   \hbadness\z@
   \hsize\maxdimen
   \showboxdepth\z@
   \tolerance\m@ne
   \hyphenpenalty\z@
   \noindent\unhbox\z@
}}
\fi
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\showhyphens}%
⟨latexrelease⟩                  {XeTeX support for \showhyphens}%
⟨latexrelease⟩\gdef\showhyphens#1{%
⟨latexrelease⟩  \setbox0\vbox{%
⟨latexrelease⟩    \color@begingroup
⟨latexrelease⟩    \everypar{}%
⟨latexrelease⟩    \parfillskip\z@skip\hsize\maxdimen
⟨latexrelease⟩    \normalfont
⟨latexrelease⟩    \pretolerance\m@ne\tolerance\m@ne
⟨latexrelease⟩    \hbadness\z@\showboxdepth\z@\ #1%
⟨latexrelease⟩    \color@endgroup}}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨*2ekernel⟩
```

(*End of definition for* `\showhyphens`.)

`\addto@hook` We need a macro to add tokens to a hook.

```
\long\def\addto@hook#1#2{#1\expandafter{\the#1#2}}
```

(*End of definition for* `\addto@hook`.)

`\@vpt`

```
\def\@vpt{5}
```

(*End of definition for* `\@vpt`.)

`\@vipt`

```
\def\@vipt{6}
```

(*End of definition for* `\@vipt`.)

`\@viipt`

```
\def\@viipt{7}
```

(*End of definition for* `\@viipt`.)

`\@viiipt`

```
\def\@viiipt{8}
```

(*End of definition for* `\@viiipt`.)

`\@ixpt`

```
\def\@ixpt{9}
```

(*End of definition for* `\@ixpt`.)

`\@xpt`

```
\def\@xpt{10}
```

(*End of definition for* `\@xpt`.)

`\@xipt`

```
\def\@xipt{10.95}
```

(*End of definition for* `\@xipt`.)

`\@xiipt`

```
\def\@xiipt{12}
```

(*End of definition for* `\@xiipt`.)

`\@xivpt`

```
\def\@xivpt{14.4}
```

(*End of definition for* `\@xivpt`.)

`\@xviipt`

```
\def\@xviipt{17.28}
```

(*End of definition for* `\@xviipt`.)

`\@xxpt`

```
\def\@xxpt{20.74}
```

(*End of definition for* `\@xxpt`.)

`\@xxvpt`

```
\def\@xxvpt{24.88}
```

(*End of definition for* `\@xxvpt`.)

⟨/2ekernel⟩

File 25
ltfssaxes.dtx

This file contains the implementation for handling extra axes splitting the series and the values into sub-categories. selection commands. See other parts of the LaTeX distribution, or *The LaTeX Companion* for higher level documentation of the LaTeX Font Selection Scheme.

Everything in the this file got introduced 2020/02/02, so we use large rollback chunks, only interrupted if necessary.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩    {\DeclareFontSeriesChangeRule}{Series change rules}%
```

1 Changing the font series

In the original NFSS implementation the series was a single attribute stored in `\f@series` and so one always had to specify both weight and width together. This means it was impossible to typeset, a paragraph in a condensed font and inside have a few words in bold weight (but still condensed) without doing this manually by requesting `\fontseries{bc}\selectfont`.

The new implementation now works differently by looking both at the current value of `\f@series` and the requested new series and out of that combination selects a resulting series value. Thus, if the current series is `c` and we ask for `b` we now get `bc`.

This is done by consulting a simple lookup table. This table is configurable (though most likely that flexibility will seldom of ever be needed) Adding or changing entries in this table are done with `\DeclareFontSeriesChangeRule`.

1.1 The series lookup table

`\DeclareFontSeriesChangeRule`

The `\DeclareFontSeriesChangeRule` defines entries in a simple database (implemented as a set of commands) that define mappings between from an existing series and requested new series and maps that to a result series (and additionally offers an alternative if the desired one is not existing):

- `#1` current `\f@series`
- `#2` requested new series
- `#3` result (if that exist for the given font family
- `#4` alternative result (if `#3` does not exist)

If an `.fd` file has its own substitution rules then `#3` exist and thus `#4` is not applied.

If there is no matching database entry or if neither the result nor the alternate result exist in the font family the requested new series is used (which then may trigger substitutions later on.

```
\def\DeclareFontSeriesChangeRule#1#2#3#4{%
  \@namedef{series@#1@#2}{{#3}{#4}}}
```

(*End of definition for* `\DeclareFontSeriesChangeRule`.)

1.2 Mapping rules for series changes

The rules set up use explicit series values not `\..default` indirections; my current feeling is that this is in fact better.

With 9 weights and 9 width classes this table is getting a bit large in the end (324 entries) but on the other hand it doesn't change and accessing speed and it is fast this way.

We could alternatively split the axis and maintain weight and width separately, but that would take more processing time and would not allow for setting up explicit exceptions nicely (not sure that this would ever get used though).

Design considerations for mapping entries:

- We make `m` to reset both weight and width (as this is how it always worked). To reset just the width `?m` is provided and to reset just the weight `m?`.
- We do support "`m`*width*" and "*weight*`m`", e.g., `mec` to mean "go to medium weight and extra-condensed width". At the end of the process we automatically drop any leftover `m` in the series name (unless it is just a single `m`).
- If there is no table entry then the target series is used unconditionally. This means that any request to set both weight and width (e.g. `bx` or `ulc`) needs no table entries. For that reason there are no entries which have a weight+width as request (i.e., second argument).

  In particular this is also true for cases involving `m`, e.g., `bm` (bold medium width) which automatically gets reduced result in `b` or `mc` (medium weight condensed) which becomes `c` as a result.
- Only a few entries have "alternative" values and perhaps most of them should get dropped. Or maybe not ... needs some thought perhaps.

  The idea is that you don't want the normal substitution to kick in because that would reset the shape first and it may be better to stay with `b` when a change to `c` is requested and `bc` doesn't exist, than to go to first change the shape to `n` and then find that `bc/n` doesn't exist either and thus ending up with `m/n`.
- Also: while I did set up all nine standard weight values from `ul` to `ub` I only bothered to provide entries for `ec`, `sc`, `c` and `x`, because other levels of compression/expansion are not in any real fonts that I know.

  Could and perhaps should be eventually extended to cover the whole set.

```
\DeclareFontSeriesChangeRule {bc}{b}{bc}{}
\DeclareFontSeriesChangeRule {bc}{c}{bc}{}
\DeclareFontSeriesChangeRule {bc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {bc}{ec}{bec} {bc}
\DeclareFontSeriesChangeRule {bc}{el}{elc}{}
\DeclareFontSeriesChangeRule {bc}{l}{lc}{}
\DeclareFontSeriesChangeRule {bc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {bc}{sc}{bsc} {bc}
\DeclareFontSeriesChangeRule {bc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {bc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {bc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {bc}{x}{bx}{}
```

```
\DeclareFontSeriesChangeRule {bx}{b}{bx}{}
\DeclareFontSeriesChangeRule {bx}{c}  {bc}  {bx} %<-----
\DeclareFontSeriesChangeRule {bx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {bx}{ec}  {bec}  {bx} %<-----
\DeclareFontSeriesChangeRule {bx}{el}{elx}{}
\DeclareFontSeriesChangeRule {bx}{l}{lx}{}
\DeclareFontSeriesChangeRule {bx}{sb} {sbx} {}
\DeclareFontSeriesChangeRule {bx}{sc}  {bsc}  {bx} %<-----
\DeclareFontSeriesChangeRule {bx}{sl}{slx}  {}
\DeclareFontSeriesChangeRule {bx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {bx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {bx}{x}{bx}{}

\DeclareFontSeriesChangeRule {b}{bx} {bx}  {b}  %<-----
\DeclareFontSeriesChangeRule {b}{c}  {bc}  {b}  %<-----
\DeclareFontSeriesChangeRule {b}{ec}  {bec}  {b} %<-----
\DeclareFontSeriesChangeRule {b}{sb} {sb}  {b}  %<-----
\DeclareFontSeriesChangeRule {b}{sc}  {bsc}  {b} %<-----
\DeclareFontSeriesChangeRule {b}{x}  {bx}  {b}  %<-----

\DeclareFontSeriesChangeRule {c}{bx} {bx}  {b}   %<-----
\DeclareFontSeriesChangeRule {c}{b}{bc}{}
\DeclareFontSeriesChangeRule {c}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {c}{el}{elc}{}
\DeclareFontSeriesChangeRule {c}{l}{lc}{}
\DeclareFontSeriesChangeRule {c}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {c}{sl}{slc}{}
\DeclareFontSeriesChangeRule {c}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {c}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {c}{x}{x}{m}               %<-----

\DeclareFontSeriesChangeRule {ebc}{b}{bc}{}
\DeclareFontSeriesChangeRule {ebc}{c}{ebc}{}
\DeclareFontSeriesChangeRule {ebc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {ebc}{ec}{ebec}{ebc}
\DeclareFontSeriesChangeRule {ebc}{el}{elc}{}
\DeclareFontSeriesChangeRule {ebc}{l}{lc}{}
\DeclareFontSeriesChangeRule {ebc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {ebc}{sc}{ebsc}{ebc}
\DeclareFontSeriesChangeRule {ebc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {ebc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {ebc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {ebc}{x}{ebx}{}

\DeclareFontSeriesChangeRule {ec}{bx} {bx}  {b}  %<-----
\DeclareFontSeriesChangeRule {ec}{b}{bec}{}
\DeclareFontSeriesChangeRule {ec}{eb}{ebec}{}
\DeclareFontSeriesChangeRule {ec}{el}{elec}{}
\DeclareFontSeriesChangeRule {ec}{l}{lec}{}
\DeclareFontSeriesChangeRule {ec}{sb}{sbec}{}
\DeclareFontSeriesChangeRule {ec}{sl}{slec}{}
\DeclareFontSeriesChangeRule {ec}{ub}{ubec}{}
\DeclareFontSeriesChangeRule {ec}{ul}{ulec}{}
\DeclareFontSeriesChangeRule {ec}{x}{x}{m}              %<-----

\DeclareFontSeriesChangeRule {sc}{bx} {bx}  {b}  %<-----
\DeclareFontSeriesChangeRule {sc}{b}{bsc}{}
```

```
\DeclareFontSeriesChangeRule {sc}{eb}{ebsc}{}
\DeclareFontSeriesChangeRule {sc}{el}{elsc}{}
\DeclareFontSeriesChangeRule {sc}{l}{lsc}{}
\DeclareFontSeriesChangeRule {sc}{sb}{sbsc}{}
\DeclareFontSeriesChangeRule {sc}{sl}{slsc}{}
\DeclareFontSeriesChangeRule {sc}{ub}{ubsc}{}
\DeclareFontSeriesChangeRule {sc}{ul}{ulsc}{}
\DeclareFontSeriesChangeRule {sc}{x}{x}{m}                %<-----

\DeclareFontSeriesChangeRule {ebx}{b}{bx}{}
\DeclareFontSeriesChangeRule {ebx}{c}{ebc}{}
\DeclareFontSeriesChangeRule {ebx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {ebx}{ec}{ebec}{}
\DeclareFontSeriesChangeRule {ebx}{el}{elx}{}
\DeclareFontSeriesChangeRule {ebx}{l}{lx}{}
\DeclareFontSeriesChangeRule {ebx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {ebx}{sc}{ebsc}{}
\DeclareFontSeriesChangeRule {ebx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {ebx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {ebx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {ebx}{x}{ebx}{}

\DeclareFontSeriesChangeRule {eb}{c}{ebc}{}
\DeclareFontSeriesChangeRule {eb}{ec}{ebec}{}
\DeclareFontSeriesChangeRule {eb}{sc}{ebsc}{}
\DeclareFontSeriesChangeRule {eb}{x}{ebx}{}

\DeclareFontSeriesChangeRule {elc}{b}{bc}{}
\DeclareFontSeriesChangeRule {elc}{c}{elc}{}
\DeclareFontSeriesChangeRule {elc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {elc}{ec}{elec}{}
\DeclareFontSeriesChangeRule {elc}{el}{elc}{}
\DeclareFontSeriesChangeRule {elc}{l}{lc}{}
\DeclareFontSeriesChangeRule {elc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {elc}{sc}{elsc}{}
\DeclareFontSeriesChangeRule {elc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {elc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {elc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {elc}{x}{elx}{}

\DeclareFontSeriesChangeRule {elx}{b}{bx}{}
\DeclareFontSeriesChangeRule {elx}{c}{elc}{}
\DeclareFontSeriesChangeRule {elx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {elx}{ec}{elec}{}
\DeclareFontSeriesChangeRule {elx}{el}{elx}{}
\DeclareFontSeriesChangeRule {elx}{l}{lx}{}
\DeclareFontSeriesChangeRule {elx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {elx}{sc}{elsc}{}
\DeclareFontSeriesChangeRule {elx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {elx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {elx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {elx}{x}{elx}{}

\DeclareFontSeriesChangeRule {el}{c}{elc}{}
\DeclareFontSeriesChangeRule {el}{ec}{elec}{}
\DeclareFontSeriesChangeRule {el}{sc}{elsc}{}
\DeclareFontSeriesChangeRule {el}{x}{elx}{}
```

```
\DeclareFontSeriesChangeRule {lc}{b}{bc}{}
\DeclareFontSeriesChangeRule {lc}{c}{lc}{}
\DeclareFontSeriesChangeRule {lc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {lc}{ec}{lec}{}
\DeclareFontSeriesChangeRule {lc}{el}{elc}{}
\DeclareFontSeriesChangeRule {lc}{l}{lc}{}
\DeclareFontSeriesChangeRule {lc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {lc}{sc}{lsc}{}
\DeclareFontSeriesChangeRule {lc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {lc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {lc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {lc}{x}{lx}{}

\DeclareFontSeriesChangeRule {lx}{b}{bx}{}
\DeclareFontSeriesChangeRule {lx}{c}{lc}{}
\DeclareFontSeriesChangeRule {lx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {lx}{ec}{lec}{}
\DeclareFontSeriesChangeRule {lx}{el}{elx}{}
\DeclareFontSeriesChangeRule {lx}{l}{lx}{}
\DeclareFontSeriesChangeRule {lx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {lx}{sc}{lsc}{}
\DeclareFontSeriesChangeRule {lx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {lx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {lx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {lx}{x}{lx}{}

\DeclareFontSeriesChangeRule {l}{bx} {bx}  {b}  %<-----
\DeclareFontSeriesChangeRule {l}{b}  {b}   {bx}  %<-----
\DeclareFontSeriesChangeRule {l}{c}  {lc}  {l}  % ? %<-----
\DeclareFontSeriesChangeRule {l}{ec}  {lec}  {l}  % ? %<-----
\DeclareFontSeriesChangeRule {l}{sb} {sb}  {b}  % ? %<-----
\DeclareFontSeriesChangeRule {l}{sc}  {lsc}  {l}  % ? %<-----
\DeclareFontSeriesChangeRule {l}{x}  {lx}  {l}  % ? %<-----

\DeclareFontSeriesChangeRule {m}{bx} {bx} {b}    %<-----
\DeclareFontSeriesChangeRule {m}{b}  {b}  {bx}  %<-----
\DeclareFontSeriesChangeRule {m}{c}  {c}  {m}  %<-----
\DeclareFontSeriesChangeRule {m}{ec}  {ec}  {m}  %<-----
\DeclareFontSeriesChangeRule {m}{l}  {l}  {m}  %<-----
\DeclareFontSeriesChangeRule {m}{sb} {sb} {b}  %<-----
\DeclareFontSeriesChangeRule {m}{sc}  {sc}  {m}  %<-----
\DeclareFontSeriesChangeRule {m}{x}  {x}  {m}  %<-----

\DeclareFontSeriesChangeRule {sbc}{b}{bc}{}
\DeclareFontSeriesChangeRule {sbc}{c}{sbc}{}
\DeclareFontSeriesChangeRule {sbc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {sbc}{ec}{sbec}{sbc}
\DeclareFontSeriesChangeRule {sbc}{el}{elc}{}
\DeclareFontSeriesChangeRule {sbc}{l}{lc}{}
\DeclareFontSeriesChangeRule {sbc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {sbc}{sc}{sbsc}{sbc}
\DeclareFontSeriesChangeRule {sbc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {sbc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {sbc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {sbc}{x}{sbx}{}

\DeclareFontSeriesChangeRule {sbx}{b}{bx}{}
```

```
\DeclareFontSeriesChangeRule {sbx}{c}{sbc}{}
\DeclareFontSeriesChangeRule {sbx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {sbx}{ec}{sbec}{}
\DeclareFontSeriesChangeRule {sbx}{el}{elx}{}
\DeclareFontSeriesChangeRule {sbx}{l}{lx}{}
\DeclareFontSeriesChangeRule {sbx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {sbx}{sc}{sbsc}{}
\DeclareFontSeriesChangeRule {sbx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {sbx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {sbx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {sbx}{x}{sbx}{}

\DeclareFontSeriesChangeRule {sb}{c}  {sbc} {bc} %? %<-----
\DeclareFontSeriesChangeRule {sb}{ec}  {sbec} {sbc} %? %<-----
\DeclareFontSeriesChangeRule {sb}{sc}  {sbsc} {sbc} %? %<-----
\DeclareFontSeriesChangeRule {sb}{x}  {sbx} {bx} %? %<-----

\DeclareFontSeriesChangeRule {slc}{b}{bc}{}
\DeclareFontSeriesChangeRule {slc}{c}{slc}{}
\DeclareFontSeriesChangeRule {slc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {slc}{ec}{slec}{}
\DeclareFontSeriesChangeRule {slc}{el}{elc}{}
\DeclareFontSeriesChangeRule {slc}{l}{lc}{}
\DeclareFontSeriesChangeRule {slc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {slc}{sc}{slsc}{}
\DeclareFontSeriesChangeRule {slc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {slc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {slc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {slc}{x}{slx}{}

\DeclareFontSeriesChangeRule {slx}{b}{bx}{}
\DeclareFontSeriesChangeRule {slx}{c}{slc}{}
\DeclareFontSeriesChangeRule {slx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {slx}{ec}{slec}{}
\DeclareFontSeriesChangeRule {slx}{el}{elx}{}
\DeclareFontSeriesChangeRule {slx}{l}{lx}{}
\DeclareFontSeriesChangeRule {slx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {slx}{sc}{slsc}{}
\DeclareFontSeriesChangeRule {slx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {slx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {slx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {slx}{x}{slx}{}

\DeclareFontSeriesChangeRule {sl}{c}{slc}{}
\DeclareFontSeriesChangeRule {sl}{ec}{slec}{}
\DeclareFontSeriesChangeRule {sl}{sc}{slsc}{}
\DeclareFontSeriesChangeRule {sl}{x}{slx}{}

\DeclareFontSeriesChangeRule {ubc}{b}{bc}{}
\DeclareFontSeriesChangeRule {ubc}{c}{ubc}{}
\DeclareFontSeriesChangeRule {ubc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {ubc}{ec}{ubec}{}
\DeclareFontSeriesChangeRule {ubc}{el}{elc}{}
\DeclareFontSeriesChangeRule {ubc}{l}{lc}{}
\DeclareFontSeriesChangeRule {ubc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {ubc}{sc}{ubsc}{}
\DeclareFontSeriesChangeRule {ubc}{sl}{slc}{}
```

```
\DeclareFontSeriesChangeRule {ubc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {ubc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {ubc}{x}{ubx}{}

\DeclareFontSeriesChangeRule {ubx}{b}{bx}{}
\DeclareFontSeriesChangeRule {ubx}{c}{ubc}{}
\DeclareFontSeriesChangeRule {ubx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {ubx}{ec}{ubec}{}
\DeclareFontSeriesChangeRule {ubx}{el}{elx}{}
\DeclareFontSeriesChangeRule {ubx}{l}{lx}{}
\DeclareFontSeriesChangeRule {ubx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {ubx}{sc}{ubsc}{}
\DeclareFontSeriesChangeRule {ubx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {ubx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {ubx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {ubx}{x}{ubx}{}

\DeclareFontSeriesChangeRule {ub}{c}{ubc}{}
\DeclareFontSeriesChangeRule {ub}{ec}{ubec}{}
\DeclareFontSeriesChangeRule {ub}{sc}{ubsc}{}
\DeclareFontSeriesChangeRule {ub}{x}{ubx}{}

\DeclareFontSeriesChangeRule {ulc}{b}{bc}{}
\DeclareFontSeriesChangeRule {ulc}{c}{ulc}{}
\DeclareFontSeriesChangeRule {ulc}{eb}{ebc}{}
\DeclareFontSeriesChangeRule {ulc}{ec}{ulec}{ulc}
\DeclareFontSeriesChangeRule {ulc}{el}{elc}{}
\DeclareFontSeriesChangeRule {ulc}{l}{lc}{}
\DeclareFontSeriesChangeRule {ulc}{sb}{sbc}{}
\DeclareFontSeriesChangeRule {ulc}{sc}{ulsc}{ulc}
\DeclareFontSeriesChangeRule {ulc}{sl}{slc}{}
\DeclareFontSeriesChangeRule {ulc}{ub}{ubc}{}
\DeclareFontSeriesChangeRule {ulc}{ul}{ulc}{}
\DeclareFontSeriesChangeRule {ulc}{x}{ulx}{}

\DeclareFontSeriesChangeRule {ulx}{b}{bx}{}
\DeclareFontSeriesChangeRule {ulx}{c}{ulc}{}
\DeclareFontSeriesChangeRule {ulx}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {ulx}{ec}{ulec}{}
\DeclareFontSeriesChangeRule {ulx}{el}{elx}{}
\DeclareFontSeriesChangeRule {ulx}{l}{lx}{}
\DeclareFontSeriesChangeRule {ulx}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {ulx}{sc}{ulsc}{}
\DeclareFontSeriesChangeRule {ulx}{sl}{slx}{}
\DeclareFontSeriesChangeRule {ulx}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {ulx}{ul}{ulx}{}
\DeclareFontSeriesChangeRule {ulx}{x}{ulx}{}

\DeclareFontSeriesChangeRule {ul}{c}{ulc}{}
\DeclareFontSeriesChangeRule {ul}{ec}{ulec}{}
\DeclareFontSeriesChangeRule {ul}{sc}{ulsc}{}
\DeclareFontSeriesChangeRule {ul}{x}{ulx}{}

\DeclareFontSeriesChangeRule {x}{b}{bx}{}
\DeclareFontSeriesChangeRule {x}{c}{c}{}
\DeclareFontSeriesChangeRule {x}{eb}{ebx}{}
\DeclareFontSeriesChangeRule {x}{ec}{ec}{}
\DeclareFontSeriesChangeRule {x}{el}{elx}{}
```

```
\DeclareFontSeriesChangeRule {x}{l}{lx}{}
\DeclareFontSeriesChangeRule {x}{sb}{sbx}{}
\DeclareFontSeriesChangeRule {x}{sc}{sc}{}
\DeclareFontSeriesChangeRule {x}{sl}{slx}{}
\DeclareFontSeriesChangeRule {x}{ub}{ubx}{}
\DeclareFontSeriesChangeRule {x}{ul}{ulx}{}
```

Special rules for `lm` etc. aren't needed because if the target `lm` is request it will used if there is no rule and that id then reduced to `l` automatically. Same for `mc` and friends. Only `?m` and `m?` need rules.

So here are the special rules for `m?`:

```
\DeclareFontSeriesChangeRule {bc}{m?}{c}{}
\DeclareFontSeriesChangeRule {bec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {bsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {bx}{m?}{x}{}
\DeclareFontSeriesChangeRule {b}{m?}{m}{}
\DeclareFontSeriesChangeRule {c}{m?}{c}{}
\DeclareFontSeriesChangeRule {ebc}{m?}{c}{}
\DeclareFontSeriesChangeRule {ebec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {ebsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {ebx}{m?}{x}{}
\DeclareFontSeriesChangeRule {eb}{m?}{m}{}
\DeclareFontSeriesChangeRule {ec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {elc}{m?}{c}{}
\DeclareFontSeriesChangeRule {elec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {elsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {elx}{m?}{x}{}
\DeclareFontSeriesChangeRule {el}{m?}{m}{}
\DeclareFontSeriesChangeRule {lc}{m?}{c}{}
\DeclareFontSeriesChangeRule {lec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {lsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {lx}{m?}{x}{}
\DeclareFontSeriesChangeRule {l}{m?}{m}{}
\DeclareFontSeriesChangeRule {m}{m?}{m}{}
\DeclareFontSeriesChangeRule {sbc}{m?}{c}{}
\DeclareFontSeriesChangeRule {sbec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {sbsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {sbx}{m?}{x}{}
\DeclareFontSeriesChangeRule {sb}{m?}{m}{}
\DeclareFontSeriesChangeRule {sc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {slc}{m?}{c}{}
\DeclareFontSeriesChangeRule {slec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {slsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {slx}{m?}{x}{}
\DeclareFontSeriesChangeRule {sl}{m?}{m}{}
\DeclareFontSeriesChangeRule {ubc}{m?}{c}{}
\DeclareFontSeriesChangeRule {ubec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {ubsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {ubx}{m?}{x}{}
\DeclareFontSeriesChangeRule {ub}{m?}{ub}{}
\DeclareFontSeriesChangeRule {ulc}{m?}{c}{}
\DeclareFontSeriesChangeRule {ulec}{m?}{ec}{}
\DeclareFontSeriesChangeRule {ulsc}{m?}{sc}{}
\DeclareFontSeriesChangeRule {ulx}{m?}{x}{}
```

```
\DeclareFontSeriesChangeRule {ul}{m?}{m}{}
\DeclareFontSeriesChangeRule {x}{m?}{x}{}
```

And there the special rules for `?m`:

```
\DeclareFontSeriesChangeRule {bc}{?m}{b}{}
\DeclareFontSeriesChangeRule {bec}{?m}{b}{}
\DeclareFontSeriesChangeRule {bsc}{?m}{b}{}
\DeclareFontSeriesChangeRule {bsc}{?m}{b}{}
\DeclareFontSeriesChangeRule {bx}{?m}{b}{}
\DeclareFontSeriesChangeRule {b}{?m}{b}{}
\DeclareFontSeriesChangeRule {c}{?m}{m}{}
\DeclareFontSeriesChangeRule {ebc}{?m}{eb}{}
\DeclareFontSeriesChangeRule {ebec}{?m}{eb}{}
\DeclareFontSeriesChangeRule {ebsc}{?m}{eb}{}
\DeclareFontSeriesChangeRule {ebsc}{?m}{eb}{}
\DeclareFontSeriesChangeRule {ebx}{?m}{eb}{}
\DeclareFontSeriesChangeRule {eb}{?m}{eb}{}
\DeclareFontSeriesChangeRule {ec}{?m}{m}{}
\DeclareFontSeriesChangeRule {elc}{?m}{el}{}
\DeclareFontSeriesChangeRule {elec}{?m}{el}{}
\DeclareFontSeriesChangeRule {elsc}{?m}{el}{}
\DeclareFontSeriesChangeRule {elsc}{?m}{el}{}
\DeclareFontSeriesChangeRule {elx}{?m}{el}{}
\DeclareFontSeriesChangeRule {el}{?m}{el}{}
\DeclareFontSeriesChangeRule {lc}{?m}{l}{}
\DeclareFontSeriesChangeRule {lec}{?m}{l}{}
\DeclareFontSeriesChangeRule {lsc}{?m}{l}{}
\DeclareFontSeriesChangeRule {lsc}{?m}{l}{}
\DeclareFontSeriesChangeRule {lx}{?m}{l}{}
\DeclareFontSeriesChangeRule {l}{?m}{l}{}
\DeclareFontSeriesChangeRule {m}{?m}{m}{}
\DeclareFontSeriesChangeRule {sbc}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sbec}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sbsc}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sbsc}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sbx}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sb}{?m}{sb}{}
\DeclareFontSeriesChangeRule {sc}{?m}{m}{}
\DeclareFontSeriesChangeRule {sc}{?m}{m}{}
\DeclareFontSeriesChangeRule {slc}{?m}{sl}{}
\DeclareFontSeriesChangeRule {slec}{?m}{sl}{}
\DeclareFontSeriesChangeRule {slsc}{?m}{sl}{}
\DeclareFontSeriesChangeRule {slsc}{?m}{sl}{}
\DeclareFontSeriesChangeRule {slx}{?m}{sl}{}
\DeclareFontSeriesChangeRule {sl}{?m}{sl}{}
\DeclareFontSeriesChangeRule {ubc}{?m}{ub}{}
\DeclareFontSeriesChangeRule {ubec}{?m}{ub}{}
\DeclareFontSeriesChangeRule {ubsc}{?m}{ub}{}
\DeclareFontSeriesChangeRule {ubsc}{?m}{ub}{}
\DeclareFontSeriesChangeRule {ubx}{?m}{ub}{}
\DeclareFontSeriesChangeRule {ub}{?m}{m}{}
\DeclareFontSeriesChangeRule {ulc}{?m}{ul}{}
\DeclareFontSeriesChangeRule {ulec}{?m}{ul}{}
\DeclareFontSeriesChangeRule {ulsc}{?m}{ul}{}
\DeclareFontSeriesChangeRule {ulsc}{?m}{ul}{}
```

```
\DeclareFontSeriesChangeRule {ulx}{?m}{ul}{}
\DeclareFontSeriesChangeRule {ul}{?m}{ul}{}
\DeclareFontSeriesChangeRule {x}{?m}{m}{}
```

Supporting rollback ...

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩   {\DeclareFontSeriesChangeRule}{Series change rules}%
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareFontSeriesChangeRule\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

1.3 Changing to a new series

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                  {\fontseries}{delay fontseries update}%
```

`\fontseries` The `\fontseries` command takes one argument which is the requested new font series. In the orginal implementation it simply saved the expanded value in `\f@series`. Now we do a bit more processing and look up the final value in the font series data base. This is done by `\merge@font@series`. But the lookup should be done within the target family and call to `\fontseries` might be followed by a `\fontfamily` call. So we delay the processing to `\selectfont` and only record the necessary action in `\delayed@f@adjustment`.

```
\DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse
    \expandafter\def\expandafter\delayed@f@adjustment\expandafter
        {\delayed@f@adjustment\delayed@merge@font@series{#1}}}
```

(*End of definition for* `\fontseries`.)

`\delayed@f@adjustment` The macro holding the delayed action(s) for use in `\selectfont`.

```
\let\delayed@f@adjustment\@empty
```

(*End of definition for* `\delayed@f@adjustment`.)

`\fontseriesforce` To change unconditionally to a new series you can use `\fontseriesforce`. Of course, if the series doesn't exist for the current family substitution still happens, but there is not dependency on the current series.

```
\DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue
    \expandafter\def\expandafter\delayed@f@adjustment\expandafter
      {\delayed@f@adjustment\edef\f@series{#1}}}
```

(*End of definition for* `\fontseriesforce`.)

`\if@forced@series` If the series gets forced we need to know that fact later on.

```
\newif\if@forced@series
```

(*End of definition for* `\if@forced@series`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\fontseries}{delay fontseries update}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse
⟨latexrelease⟩                                    \merge@font@series{#1}}
⟨latexrelease⟩\DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue
⟨latexrelease⟩                                         \edef\f@series{#1}}
⟨latexrelease⟩\let\delayed@f@adjustment\@undefined
⟨latexrelease⟩
```

For a roll forward we may have to define `\if@forced@series` but this needs doing in a way that TEX doesn't see it when skipping over conditionals.

```
⟨latexrelease⟩\expandafter\newif\csname if@forced@series\endcsname
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\fontseries}{delay fontseries update}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
⟨latexrelease⟩\let\fontseriesforce\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩    {\merge@font@series}{Merge series values}%
```

`\merge@font@series` We look up the data base value by expanding the right command twice. If no such value exist then the result will be `\relax` otherwise it will be the two brace groups: the desired result and the alternate result. The first case means that the third argument to `\merge@font@series` will be empty.

```
\def\merge@font@series#1{%
  \expandafter\expandafter\expandafter
  \merge@font@series@
    \csname series@\f@series @#1\endcsname
    {#1}%
    \@nil
}
```

(*End of definition for* `\merge@font@series`.)

`\merge@font@series@` This now defines the new `\f@series`:

```
\def\merge@font@series@#1#2#3\@nil{%
```

If the third argument is empty there is no database entry for the combination and the second argument holds the new series so we return that.

Originally the test was simply `\ifx!#3!` but that actually dies if `#3` starts with a conditional and in the definition of `\AmSfont` that is actually the case.

```
%\ifcat\expandafter X\detokenize{#1}X%
  \def\reserved@a{#3}%
  \ifx\reserved@a\@empty
    \set@target@series{#2}%
  \else
```

Otherwise we check if the desired result for the series (#1) exists for the font family and the current shape. All this happens inside \selectfont which has already taken care to load the .fd, file if necessary.

```
    \edef\reserved@a{\f@encoding /\f@family /#1/\f@shape}%
     \ifcsname \reserved@a \endcsname
```

If the desired result is available then we use that. However, we do need some post-processing because we need to drop surplus ms due to the way naming convention was designed in the '90s (sigh).

```
        \set@target@series{#1}%
```

If not, then we try the alternate result (#2).

```
    \else
        \ifcsname \f@encoding /\f@family /#2/\f@shape \endcsname
```

If the alternate result exist we use that and also issue a warning (or rather a log entry) that we didn't managed to change to the desired font.

```
          \set@target@series{#2}%
          \@font@shape@subst@warning
```

If that doesn't exist either, then we use the requested series unmodified (again with a warning).

```
        \else
          \set@target@series{#3}%
          \@font@shape@subst@warning
        \fi
    \fi
  \fi
}
```

It is possible that the previous font and the new one are actually identical (and the font was not found because it still needs loading) in which case a warning would look rather odd. So we make a quick check for that (which is the reason why we defined \@reserveda above instead of doing inline testing inside \ifcsname).

```
\def\@font@shape@subst@warning{%
   \edef\reserved@b{\curr@fontshape}%
   \ifx\reserved@a\reserved@b \else
     \@font@warning{Font shape `\reserved@a' undefined\MessageBreak
                    using `\reserved@b' instead}%
   \fi
}
```

(*End of definition for* \merge@font@series@.)

\merge@font@series@without@substitution
\merge@font@series@without@substitution@
\delayed@merge@font@series

\merge@font@series@without@substitution works like \merge@font@series, i.e., it looks up the combination in the rule base and if there exists an entry it uses it and if not it uses the new series value. However, it doesn't check if there is actually a font face with the new series value as \merge@font@series does. This simplified command is used in \selectfont at a point where other font attributes are not yet updated so that checking the font face might result incorrect in substitutions.

```
\def\merge@font@series@without@substitution#1{%
  \expandafter\expandafter\expandafter
  \merge@font@series@without@substitution@
    \csname series@\f@series @#1\endcsname
```

```
    {#1}%
    \@nil
}
```

```
\def\merge@font@series@without@substitution@#1#2#3\@nil{%
  \def\reserved@a{#3}%
  \ifx\reserved@a\@empty
    \set@target@series{#2}%
  \else
    \set@target@series{#1}%
  \fi
}
```

(*End of definition for* `\merge@font@series@without@substitution`, `\merge@font@series@without@substitution@`, *and* `\delayed@merge@font@series`.)

`\delayed@merge@font@series` When we delay the merge action in `\fontseries` we first attempt to use merging without substitution. If that results in a non-existing font face the merge is redone in `\selectfont` using a version with substitution. See `\selectfont` for details.

```
\let\delayed@merge@font@series\merge@font@series@without@substitution
```

(*End of definition for* `\delayed@merge@font@series`.)

`\maybe@load@fontshape` A small helper that we use a couple of times: try loading a fontshape (in a group because `\try@load@fontshape` normalizes catcodes and we also want to change `\typeout` so that it doesn't report missing `.fd` files on the terminal).

```
\def\maybe@load@fontshape{%
  \begingroup
    \let \typeout \@font@info
    \try@load@fontshape
  \endgroup}
```

(*End of definition for* `\maybe@load@fontshape`.)

`\set@target@series` Finally the code for normalizing the `\f@series` value.

The combined series value determined by the mapping may still contain an `m` that we have to remove (as the `.fd` files use `c` not `mc` to denote a medium weight condensed series, etc.). We do this in all branches above because a user might have written

```
\DeclareFontSeriesChangeRule {m}{sc}{msc}{mc}
```

instead of using `sc` and `c` as needed in the `.fd` file.

```
\def\set@target@series#1{%
```

We need to `\edef` the argument first in case it starts with a conditional. Then we check (and perhaps drop) an "m" from the value and assign the result to `\f@series`.

```
    \edef\f@series{#1}%
    \series@maybe@drop@one@m\f@series\f@series
}
```

(*End of definition for* `\set@target@series`.)

\series@maybe@drop@one@m If the series value is in NFSS notation then it should not contain any "m" unless it is just an "m" by it own. So we need to drop surplus "m"s. But we better don't do this for full names, such as "semibold" as used by autoinst, for example. So we test against the possible explicit values that should drop an "m". After that we assign the result to #2 for further use.

```
\def\series@maybe@drop@one@m#1{%
  \expandafter\series@maybe@drop@one@m@x\expandafter{#1}}

\def\series@maybe@drop@one@m@x#1#2{%
```

The code below is an inline version of the \in@ macro without the group, so that it works in \accent.

```
  \def\in@@##1,#1,{}%
  \series@check@toks\expandafter{\in@@
    ,ulm,elm,lm,slm,mm,sbm,bm,ebm,ubm,muc,mec,mc,msc,msx,mx,mex,mux,{}{},#1,}%
  \edef\in@@{\the\series@check@toks}%
  \ifx\in@@\@empty
```

The default definition for \bfdefault etc is actually b\@empty so that we can detect if the user has changed the default. However that means a) the above test will definitely fail (maybe something to change) and b) we better use \edef on the next line to get rid of it as otherwise the test against #2 (e.g. \bfdef@ult) will fail in other places.

```
    \edef#2{#1}%
  \else
    \edef#2{\expandafter\series@drop@one@m #1m\series@drop@one@m}%
  \fi
}
```

As a precaution we use a private toks register not \toks@ as that is no longer hidden inside the group.

```
\newtoks\series@check@toks
```

(*End of definition for* \series@maybe@drop@one@m.)

\series@drop@one@m Drop up to two ms but keep one if that makes the series value empty. Actually, with the current implementation we know that there is at least one in the series value itself and we added one after it, so all we have to do is now returning #1#2 and dropping the rest.

```
\def\series@drop@one@m#1m#2m#3\series@drop@one@m{%
%  \ifx\relax#1#2\relax m\else#1#2\fi
   #1#2%
}
```

(*End of definition for* \series@drop@one@m.)

Supporting rollback ...

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩    {\merge@font@series}{Merge series values}%
⟨latexrelease⟩
⟨latexrelease⟩\let\merge@font@series\@undefined
⟨latexrelease⟩\let\merge@font@series@\@undefined
⟨latexrelease⟩\let\@font@shape@subst@warning\@undefined
⟨latexrelease⟩\let\merge@font@series@without@substitution\@undefined
⟨latexrelease⟩\let\merge@font@series@without@substitution@\@undefined
```

```
⟨latexrelease⟩\let\delayed@merge@font@series\@undefined
⟨latexrelease⟩\let\maybe@load@fontshape\@undefined
⟨latexrelease⟩\let\set@target@series\@undefined
⟨latexrelease⟩\let\series@maybe@drop@one@m\@undefined
⟨latexrelease⟩\let\series@drop@one@m\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

2 Changing the shape

Shapes are also split in two axes (though it could be more if that is desirable), essentially building in an "sc" axis.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩    {\DeclareFontShapeChangeRule}{Font shape change rules}%
```

\DeclareFontShapeChangeRule

The database for shapes is done in exactly the same way, only that it is much smaller and we usually have no alternative shape (or rather it is empty thus not used).

```
\def\DeclareFontShapeChangeRule #1#2#3#4{%
  \@namedef{shape@#1@#2}{{#3}{#4}}}
```

(*End of definition for* \DeclareFontShapeChangeRule.)

There is kind of the same problem with returning back from `sc` to normal. It sort of needs its own letter. In `fontspec` this was solved by the first time `\upshape` changes `it` or `sl` back (so only `sc` remains) and second time it changes then `sc` back to normal. Maybe that's not a bad way to handle it, but decided for a slightly different approach: `n` always returns to "normal", ie resets everything and `up` changes italic or slanted to upright and `ulc` undoes small caps.

So we now offer `\normalshape` (using `\shapedefault`) which is normally the same as calling both `\ulcshape` and `\upshape`, only more efficient.

\ulcshape
\textulc
\ulcdefault

To request going back to upper/lowercase we need a new command. It uses `ulc` as shape name but this shape is virtual, i.e., it doesn't exist as a real shape, it is only used as part of the database table entries and thus only appears in the second argument there (but not in the first).

```
\DeclareRobustCommand\ulcshape
        {\not@math@alphabet\ulcshape\relax
         \fontshape\ulcdefault\selectfont}
\let\ulcdefault\@undefined      % for rollback
\newcommand\ulcdefault{ulc}
```

(*End of definition for* \ulcshape, \textulc, *and* \ulcdefault.)

\swshape
\textsw
\swdefault

New command to select a swash shape. The standard rules put this in the same category as italics or slanted, i.e., if you ask for it then italics are undone. One could provide more complicated rules so that `it` + `sw` becomes `swit` but given that there are only very few fonts that have swash letters that level of flexibility (these days) would be just resulting in a lot of combinations that do not exist.

```
\DeclareRobustCommand\swshape
        {\not@math@alphabet\swshape\relax
         \fontshape\swdefault\selectfont}
\let\swdefault\@undefined      % for rollback
\newcommand\swdefault{sw}
```

(*End of definition for* `\swshape`, `\textsw`, *and* `\swdefault`.)

`\sscshape` `\textssc` `\sscdefault`

New command to select spaced small capitals. This is only here because `fontaxes` offered it. There isn't a single free font that supports it. However, some commercial ones do, so we offer it so that at some point `fontaxes` could be retired.

So far there aren't any rules for it—probably there should be some putting it in the same category as `sc`.

```
\DeclareRobustCommand\sscshape
        {\not@math@alphabet\sscshape\relax
         \fontshape\sscdefault\selectfont}
\let\sscdefault\@undefined      % for rollback
\newcommand\sscdefault{ssc}
```

(*End of definition for* `\sscshape`, `\textssc`, *and* `\sscdefault`.)

2.1 Mapping rules for shape combinations

Many of the entries are commented out as we will get that result without any entry.

```
%\DeclareFontShapeChangeRule {n}{n}   {n}   {}
\DeclareFontShapeChangeRule {n}{it}  {it}  {sl}
\DeclareFontShapeChangeRule {n}{sl}  {sl}  {it}
%\DeclareFontShapeChangeRule {n}{sw}  {sw}  {}
%\DeclareFontShapeChangeRule {n}{sc}  {sc}  {}
\DeclareFontShapeChangeRule {n}{ulc} {n}   {}
\DeclareFontShapeChangeRule {n}{up} {n}    {}
```

```
%\DeclareFontShapeChangeRule {it}{n}   {n}     {}
%\DeclareFontShapeChangeRule {it}{it}  {it}    {}
\DeclareFontShapeChangeRule {it}{sl}  {sl}    {it}
%\DeclareFontShapeChangeRule {it}{sw}  {sw}    {}
```

If neither `scit` nor `scsl` exist then `sc` will be used as a fallback albeit with a log entry, so except for the latter there will be no change for CM or Latin Modern fonts.

```
\DeclareFontShapeChangeRule {it}{sc}  {scit}  {scsl}
\DeclareFontShapeChangeRule {it}{ulc} {it}    {}
\DeclareFontShapeChangeRule {it}{up} {n}      {}
```

```
%\DeclareFontShapeChangeRule {sl}{n}   {n}     {}
\DeclareFontShapeChangeRule {sl}{it}  {it}    {sl}
%\DeclareFontShapeChangeRule {sl}{sl}  {sl}    {}
%\DeclareFontShapeChangeRule {sl}{sw}  {sw}    {}
\DeclareFontShapeChangeRule {sl}{sc}  {scsl}  {scit}
\DeclareFontShapeChangeRule {sl}{ulc} {sl}    {}
\DeclareFontShapeChangeRule {sl}{up} {n}      {}
```

```
%\DeclareFontShapeChangeRule {sc}{n}   {n}     {}
\DeclareFontShapeChangeRule {sc}{it}  {scit}  {scsl}
\DeclareFontShapeChangeRule {sc}{sl}  {scsl}  {scit}
\DeclareFontShapeChangeRule {sc}{sw}  {scsw}  {sw}
%\DeclareFontShapeChangeRule {sc}{sc}  {sc}    {}
\DeclareFontShapeChangeRule {sc}{ulc} {n}     {}
```

The next rule might be a bit surprising and rightly so. Correct would be that `sc` is not affected by `up`, i.e., remains `sc` as showed in the commented out rule. However, for nearly three decades commands such as `\sc` or `\textup` changed small caps back to the "normal" shape. So for backward compatibility we keep that behavior.

As a result you are currently typesetting in `scit` or `scsl` using `\upshape` twice will return you to the normal shape too, the first will change to `sc` and the second (because of the rule below) change that to `n`. This is the way `fontspec` implemented its version on this interface, so this rule means we are also compatible with the way `fontspec` behaved. Still it remains an oddity which I would rather liked to have avoided.

```
%\DeclareFontShapeChangeRule {sc}{up} {sc}     {}
\DeclareFontShapeChangeRule {sc}{up} {n}     {}
```

```
%\DeclareFontShapeChangeRule {scit}{n}   {n}    {}
\DeclareFontShapeChangeRule {scit}{it}  {scit}  {}
\DeclareFontShapeChangeRule {scit}{sl}  {scsl}  {scit}
\DeclareFontShapeChangeRule {scit}{sw}  {scsw}  {sc}    % or scit?
\DeclareFontShapeChangeRule {scit}{sc}  {scit}  {}
\DeclareFontShapeChangeRule {scit}{ulc} {it}    {}
\DeclareFontShapeChangeRule {scit}{up} {sc}     {}
```

The previous rule assumes that if `scit` exists then `it` exists as well. If not, the mechanism will save `ulc` in `\f@series` which most certainly doesn't exist. So when a font is later selected that would result in a substitution (so no harm done really). Alternatively, we could in this case use `n` as alternative, which may be a bit faster, but such a setup would be so weird in the first place that this isn't worth the effort.

```
%\DeclareFontShapeChangeRule {scsl}{n}   {n}     {}
\DeclareFontShapeChangeRule {scsl}{it}  {scit}  {scsl}
\DeclareFontShapeChangeRule {scsl}{sl}  {scsl}  {}
\DeclareFontShapeChangeRule {scsl}{sw}  {scsw}  {sc}    % or scsl?
\DeclareFontShapeChangeRule {scsl}{sc}  {scsl}  {}
\DeclareFontShapeChangeRule {scsl}{ulc} {sl}    {}
\DeclareFontShapeChangeRule {scsl}{up}   {sc}   {}
```

```
%\DeclareFontShapeChangeRule {scsw}{n}   {n}     {}
\DeclareFontShapeChangeRule {scsw}{it}  {scit}  {scsw}
\DeclareFontShapeChangeRule {scsw}{sl}  {scsl}  {}
\DeclareFontShapeChangeRule {scsw}{sw}  {scsw}  {}
\DeclareFontShapeChangeRule {scsw}{sc}  {scsw}  {}
\DeclareFontShapeChangeRule {scsw}{ulc} {sw}    {}
\DeclareFontShapeChangeRule {scsw}{up} {sc}     {}
```

```
%\DeclareFontShapeChangeRule {sw}{n}   {n}     {}
%\DeclareFontShapeChangeRule {sw}{it}  {it}    {}
%\DeclareFontShapeChangeRule {sw}{sl}  {sl}    {}
%\DeclareFontShapeChangeRule {sw}{sw}  {sw}    {}
\DeclareFontShapeChangeRule {sw}{sc}  {scsw}  {}
\DeclareFontShapeChangeRule {sw}{ulc} {sw}    {}
\DeclareFontShapeChangeRule {sw}{up} {n}      {}
```

Supporting rollback ...

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩   {\DeclareFontShapeChangeRule}{Font shape change rules}%
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareFontShapeChangeRule\@undefined
⟨latexrelease⟩\let\ulcshape\@undefined
⟨latexrelease⟩\let\ulcdefault\@undefined
⟨latexrelease⟩\let\swshape\@undefined
```

```
⟨latexrelease⟩\let\swdefault\@undefined
⟨latexrelease⟩\let\sscshape\@undefined
⟨latexrelease⟩\let\sscdefault\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

2.2 Changing to a new shape

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩    {\fontshape}{Font shape change}%
```

`\fontshape` Again the `\fontshape` now has to do a lookup to get to its new value in `\f@shape`. The method is exactly the same as in `\fontseries`.

```
\DeclareRobustCommand\fontshape[1]
   {\expandafter\def\expandafter\delayed@f@adjustment\expandafter
        {\delayed@f@adjustment\delayed@merge@font@shape{#1}}}
```

(*End of definition for* `\fontshape`.)

`\fontshapeforce` The unconditional version:

```
\DeclareRobustCommand\fontshapeforce[1]
   {\expandafter\def\expandafter\delayed@f@adjustment\expandafter
        {\delayed@f@adjustment\edef\f@shape{#1}}}
```

(*End of definition for* `\fontshapeforce`.)

Supporting rollback ...

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩    {\fontshape}{Font shape change}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\fontshape[1]{\merge@font@shape{#1}}
⟨latexrelease⟩\DeclareRobustCommand\fontshapeforce[1]{\edef\f@shape{#1}}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩    {\fontshape}{Font shape change}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
⟨latexrelease⟩\let\fontshapeforce\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩    {\merge@font@shape}{Font shape change rules}%
```

`\merge@font@shape` Look up the database entry (if existing) and act accordingly.

```
\def\merge@font@shape#1{%
  \expandafter\expandafter\expandafter
  \merge@font@shape@
    \csname shape@\f@shape @#1\endcsname
    {#1}%
```

```
    \@nil
}
```

(*End of definition for* `\merge@font@shape`.)

`\merge@font@shape@` Same game now, except that we look at shapes not series values and we can set the shape without the complication of dropping "m"s from the name as we had to for the series.

```
\def\merge@font@shape@#1#2#3\@nil{%
  \def\reserved@a{#3}%
  \ifx\reserved@a\@empty
    \edef\f@shape{#2}%
  \else
```

`\reserved@a` is used in `\@font@shape@subst@warning` so we have to define it in addition to do the `\ifcsname` test

```
    \edef\reserved@a{\f@encoding /\f@family /\f@series/#1}%
    \ifcsname \reserved@a\endcsname
       \edef\f@shape{#1}%
    \else
       \ifcsname \f@encoding /\f@family /\f@series/#2\endcsname
         \edef\f@shape{#2}%
         \@font@shape@subst@warning
       \else
         \edef\f@shape{#3}%
         \@font@shape@subst@warning
       \fi
    \fi
  \fi
}
```

(*End of definition for* `\merge@font@shape@`.)

`\merge@font@shape@without@substitution` `\merge@font@shape@without@substitution@` `\delayed@merge@font@shape` See definition of `\selectfont` for how these macros are used.

```
\def\merge@font@shape@without@substitution#1{%
  \expandafter\expandafter\expandafter
  \merge@font@shape@without@substitution@
    \csname shape@\f@shape @#1\endcsname
    {#1}%
    \@nil
}
```

```
\def\merge@font@shape@without@substitution@#1#2#3\@nil{%
  \def\reserved@a{#3}%
  \ifx\reserved@a\@empty
    \edef\f@shape{#2}%
  \else
    \edef\f@shape{#1}%
  \fi
}
```

```
\let\delayed@merge@font@shape\merge@font@shape@without@substitution
```

(*End of definition for* `\merge@font@shape@without@substitution`, `\merge@font@shape@without@substitution@`, *and* `\delayed@merge@font@shape`.)

\normalshape

\normalshape resets both sub-axes if the default rules are used.

```
\protected\def\normalshape
    {\not@math@alphabet\normalshape\relax
      \fontshape\shapedefault\selectfont}%
```

(*End of definition for* \normalshape.)

3 Make sure we win ...

This code implements one aspect of what the package fontaxes provide. So its redefinitions for the various shape commands, such as \itshape should no longer happen. We therefore force the standard definitions at \AtBeginDocument (later when this is defined). Once fontaxes is no longer doing such redefinitions that could be taken out again.

We use a separate macro so that we can easily disable this (in case of rollback).

\reinstall@nfss@defs

I use \protected here not \DeclareRobustCommand to avoid extra status lines.

```
\def\reinstall@nfss@defs{%
  \protected\def\upshape
          {\not@math@alphabet\upshape\relax
           \fontshape\updefault\selectfont}%
  \protected\def\slshape
          {\not@math@alphabet\slshape\relax
           \fontshape\sldefault\selectfont}%
  \protected\def\scshape
          {\not@math@alphabet\scshape\relax
           \fontshape\scdefault\selectfont}%
  \protected\def\itshape
          {\not@math@alphabet\itshape\mathit
           \fontshape\itdefault\selectfont}%
  \protected\def\ulcshape
          {\not@math@alphabet\ulcshape\relax
           \fontshape{ulc}\selectfont}%
  \protected\def\swshape
          {\not@math@alphabet\swshape\relax
           \fontshape\swdefault\selectfont}%
  \protected\def\sscshape
          {\not@math@alphabet\sscshape\relax
           \fontshape\sscdefault\selectfont}%
}
```

(*End of definition for* \reinstall@nfss@defs.)

Supporting rollback ...

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩   {\merge@font@shape}{Font shape change rules}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
⟨latexrelease⟩\let\fontshapeforce\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\merge@font@shape\@undefined
⟨latexrelease⟩\let\merge@font@shape@\@undefined
⟨latexrelease⟩
```

```
⟨latexrelease⟩\let\merge@font@shape@without@substitution\@undefined
⟨latexrelease⟩\let\merge@font@shape@without@substitution@\@undefined
⟨latexrelease⟩\let\delayed@merge@font@shape\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\normalshape\@undefined
⟨latexrelease⟩
```

This is always called in `\document` so don't make it undefined.

```
⟨latexrelease⟩
⟨latexrelease⟩\let\reinstall@nfss@defs\relax
⟨latexrelease⟩\EndIncludeInRelease
```

This initializes the 2020/02/02 extensions to NFSS after any changes in the preamble.

```
⟨∗2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\reinstall@nfss@defs}{NFSS series init}%
\g@addto@macro\@kernel@after@begindocument@before
               {\reinstall@nfss@defs\init@series@setup}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

The initialization was introduced in 2020/02/02 but

```
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {\reinstall@nfss@defs}{NFSS series init}%
⟨latexrelease⟩\AtBeginDocument{\reinstall@nfss@defs\init@series@setup}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\reinstall@nfss@defs}{NFSS series init}%
⟨latexrelease⟩\EndIncludeInRelease
⟨∗2ekernel⟩
⟨/2ekernel⟩
```

File 26
ltfsstrc.dtx

1 Introduction

This package contains the code for tracing font loading and font changes. It basically overlays some of the low-level functions of NFSS with additional code used for tracing.

The package accepts the following options:

errorshow Write all information about font changes etc. only to the transcript file unless an error happens. This means that information about font substitution will not be shown on the terminal.

warningshow Show all NFSS warnings on the terminal. This setting corresponds to the default behaviour of NFSS if the `tracefnt` package is *not* loaded!

infoshow Show all NFSS warning and all NFSS info messages (that are normally only written to the transcript file) also on the terminal. This is the default if the `tracefnt` package is loaded.

debugshow In addition to `infoshow` show also changing of math fonts as far as possible (this option can produce a large amount of output).

loading Show the name of external fonts when they are loaded. This option shows only "newly" loaded fonts not those already preloaded in the format or the class file before the `tracefnt` package became active.

pausing Turn all font warnings into errors so that LaTeX will stop.

2 A driver for this document

The next bit of code contains the documentation driver file for TeX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DocStrip program.

When this file is processed directly by LaTeX this will produce the documentation as well.

```
⟨*driver⟩
\documentclass{ltxdoc}


%\OnlyDescription  % comment out for implementation details

\begin{document}
   \DocInput{ltfsstrc.dtx}
\end{document}
⟨/driver⟩
```

3 The Implementation

Warning: Read the macro documentation with a grain of salt. It is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

If we are making a package file it is a good idea to test whether we are running under 2e. This code is actually placed at the very beginning of this file for easier maintenance, thus commented out here.

```
⟨*package⟩
%\NeedsTeXFormat{LaTeX2e}
%\ProvidesPackage{tracefnt}[??/??/?? v?.??
%                           Standard LaTeX package (font tracing)]
⟨/package⟩
```

The debug module makes use of commands contained in a special package file named trace.sty.[33]

```
⟨+debug⟩ \input trace.sty
```

4 Handling Options

\tracingfonts

Here is the definition of the integer register for the font trace. As a default in a package file we use 1 to give error messages if fonts are substituted. If this code is used for debugging or tracing reasons in the format file (i.e. in fam.dtx) we use 0 as the default. But if no font trace is used we build a definition that will produce a warning message.

```
⟨*2ekernel⟩
\def\tracingfonts{%
  \@font@warning{Command \noexpand\tracingfonts
           not provided.\MessageBreak
           Use the ‘tracefnt’ package.\MessageBreak Command found:}%
       \count@}
⟨/2ekernel⟩
```

The \count@ in the line above will remove the number after \tracingfonts. Note that this definition will be overwritten be the next line if one of these modules are included.

```
⟨*package, trace, debug⟩
\newcount\tracingfonts
\tracingfonts=0
⟨/package, trace, debug⟩
```

(*End of definition for* \tracingfonts.)

The option errorshow turns off all warnings so that only real errors are shown. warningshow corresponds to the NFSS default (when tracefnt is not loaded). infoshow is the default for this package here; and debugshow, loading, and pausing extend the amount of information even further.

```
⟨*package⟩
\DeclareOption{errorshow}{%
   \def\@font@info#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                     {LaTeX Font Info: \space\space\space#1}}%
```

[33] This package is not in distribution at the moment (and probably doesn't any longer work). Think of this part of the code as being historical artifacts.

```
    \def\@font@warning#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                        {LaTeX Font Warning: #1}}%
     }
\DeclareOption{warningshow}{%
   \def\@font@info#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                      {LaTeX Font Info: \space\space\space#1}}%
    \def\@font@warning#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                        {LaTeX Font Warning: #1}}%
     }
\DeclareOption{infoshow}{%
   \def\@font@info#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                      {LaTeX Font Info: \space\space\space#1}}%
    \def\@font@warning#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                        {LaTeX Font Warning: #1}}%
     }
\DeclareOption{loading}{%
    \tracingfonts\tw@
   }
\DeclareOption{debugshow}{%
    \ExecuteOptions{infoshow}%
    \tracingfonts\thr@@
   }

\DeclareOption{pausing}{%
    \def\@font@warning#1{%
      \GenericError
             {(Font)\@spaces\@spaces\@spaces\space\space}%
             {LaTeX Font Warning: #1}%
             {See the LaTeX Companion for details.}%
             {I'll stop for every LaTeX Font Warning because
              you requested\MessageBreak the 'pausing' option
              to the tracefnt package.}}%
   }
```

We make `infoshow` the default, which in turn defines `\font@warning` and `\font@info`.

```
\ExecuteOptions{infoshow}
\ProcessOptions
⟨/package⟩
```

We also need a default definition inside the kernel:

```
⟨*2ekernel⟩
\def\@font@info#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                      {LaTeX Font Info: \space\space\space#1}}%
\def\@font@warning#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                        {LaTeX Font Warning: #1}}%
⟨/2ekernel⟩
```

5 Macros common to fam.tex and tracefnt.sty

In the first versions of tracefnt.dtx some macros of fam.dtx[34] were redefined to included the extra tracing information. Now these macros are all defined in this file (i.e. removed from fam.dtx) and different production versions can be obtained simply by specifying a different set of modules to include when generating ltfss.dtx.

5.1 General font loading

\extract@font

This macro organizes the font loading. It first calls \get@external@font which will return in \external@font the name of the external font file (the .tfm) as it was determined by the NFSS tables.

```
⟨*2ekernel | package⟩
\def\extract@font{%
   \get@external@font
```

Then the external font is loaded and assigned to the font identifier stored inside \font@name (for this reason we need \expandafter).

```
   \global\expandafter\font\font@name\external@font\relax
```

When tracing we typeout the internal and external font name.

```
⟨*trace⟩
    \ifnum \tracingfonts >\@ne
    \@font@info{External font `\external@font'
              loaded as\MessageBreak \font@name}\fi
⟨/trace⟩
```

Finally we call the corresponding "loading action" macros to finish things. First the font is locally selected to allow the use of \font inside the loading action macros.

```
    \font@name \relax
```

The next two lines execute the "loading actions" for the family and then for the individual font shape.

```
    \csname \f@encoding+\f@family\endcsname
    \csname\curr@fontshape\endcsname
    \relax
       }
⟨/2ekernel | package⟩
```

The \relax at the end needs to be explained. This is inserted to prevent TEX from scanning too far when it is executing the replacement text of the loading code macros.

(*End of definition for* \extract@font.)

\get@external@font

This function tries to find an external font name. It will place the name into the macro \external@font. If no font is found it will return the one that was defined via \DeclareErrorFont.

```
⟨*2ekernel⟩
\def\get@external@font{%
```

We don't know the external font name at the beginning.

```
   \let\external@font\@empty
   \edef\font@info{\expandafter\expandafter\expandafter\string
        \csname \curr@fontshape \endcsname}%
   \try@size@range
```

[34] This file is currently not distributed in documented form. Its code is part of ltfss.dtx.

If this failed, we'll try to substitute another size of the same font. This is done by the `\try@size@substitution` macro. It "knows about" `\do@extract@font`, `\font@name`, `\f@size`, and so on.

```
   \ifx\external@font\@empty
      \try@size@substitution
      \ifx\external@font\@empty
         \@latex@error{Font \expandafter \string\font@name\space
                       not found}\@eha
         \error@fontshape
         \get@external@font
   \fi\fi
}
⟨/2ekernel⟩
```

(*End of definition for* `\get@external@font`.)

```
⟨*2ekernel | latexrelease | package⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                    {\selectfont}{Add hook to \selectfont}%
```

`\selectfont` The macro `\selectfont` is called whenever a font change must take place.

```
\DeclareRobustCommand\selectfont
        {%
```

When `debug` is specified we actually want something like 'undebug'. The font selection is now stable so that using `\tracingall` on some other macros will show us a lot of unwanted information about font loading. Therefore we disable tracing during font loading as long as `\tracingfonts` is less than 4.

```
⟨+debug⟩  \pushtracing
⟨+debug⟩  \ifnum\tracingfonts<4 \tracingoff
⟨+debug⟩  \else \tracingon\p@selectfont \fi
```

If `\baselinestretch` was redefined by the user it will not longer match its internal counterpart `\f@linespread`. If so we call `\set@fontsize` to prepare `\size@update`.

```
   \ifx\f@linespread\baselinestretch \else
     \set@fontsize\baselinestretch\f@size\f@baselineskip \fi
```

The series and shape updates are only prepared by `\fontseries` and `\fontshape` but not executed until after we are ready to change the font face. This way they happen after a possibly new family is set which is important because they look at the available font faces in that family and alter the selection based on availibility. Several calls to `\fontseries` or `\fontshape` are delayed in the order in which they appear, so that by switching them one can work around missing intermediate font faces and avoid substitutions.

We first attempt to do the merge without any substitution. As we might end up with a non-existing font face we may have to restart and therefore save the current values of `\f@series` and `\f@shape` before the merge.

But first we make a quick test to see if there are any delayed actions, because if not it is pointless to make all the assignments and try loading a missing fontshape.

```
   \ifx\delayed@f@adjustment\@empty
   \else
     \let\f@shape@saved\f@shape
     \let\f@series@saved\f@series
```

Then we run the delayed adjustments (which use the `\..@without@substitution` commands):

```
      \delayed@f@adjustment
```

We then check if the resulting cominbation is valid but for this we have to make sure that the appropiate `.fd` is loaded if that hasn't happened so far.

```
      \maybe@load@fontshape
      \ifcsname \f@encoding/\f@family/\f@series/\f@shape \endcsname
```

If this macro is defined then we are good and no further action is necessary.

Otherwise the combination is not valid, so we redo the merge but this time with substitutions.

```
      \else
        \let\f@shape\f@shape@saved
        \let\f@series\f@series@saved
        \let\delayed@merge@font@shape\merge@font@shape
        \let\delayed@merge@font@series\merge@font@series
        \delayed@f@adjustment
        \let\delayed@merge@font@shape\merge@font@shape@without@substitution
        \let\delayed@merge@font@series\merge@font@series@without@substitution
      \fi
```

Now the series and shape values are updated and we clear `\delayed@f@adjustment`. This is important because on the next execution of `\selectfont` we should not mistakenly redo the delayed actions if there wasn't any series or shape change.

```
      \let\delayed@f@adjustment\@empty
    \fi
```

If the series was forced we should now cancel that in case the next series change is done with some low-level setting to `\f@series`.

```
    \@forced@seriesfalse
```

Then we generate the internal name of the font by concatenating *family*, *series*, *shape*, and current *size*, with slashes as delimiters between them. This is much more readable than standard LaTeX's `\twfbf`, etc. We define `\font@name` globally, as always. The reason for this is explained later on.

```
    \xdef\font@name{%
      \csname\curr@fontshape/\f@size\endcsname}%
```

We call the macro `\pickup@font` which will load the font if necessary.

```
    \pickup@font
```

Then we select the font.

```
    \font@name
```

After switching fonts we run a hook, so that packages can make last minute alterations based on the new font (originally provided in everysel but using a different interface).

```
    \UseHook{selectfont}%
```

Finally we call `\size@update`. This macro is normally empty but will contain actions (like setting the `\baselineskip`) that have to be carried out when the font size, the base `\baselineskip` or the `\baselinestretch` have changed.

```
    \size@update
```

A similar function is called to handle anything related to encoding updates. This one is changed from `\relax` by `\fontencoding`.

```
    \enc@update
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
⟨+debug⟩ \poptracing
    }
```

(*End of definition for* `\selectfont`.)

`selectfont` Declare the hook used in selecfont in the kernel, but not inside the tracefnt package.

```
⟨-trace⟩\NewHook{selectfont}
```

(*End of definition for* `selectfont`.)

If `\tracingfonts` is greater than 2 we also show the font switch inside `\selectfont`. We do this by adding this code to the hook in the tracefnt package: macro might redefine `\font@name`.

```
⟨*trace⟩
\AddToHook{selectfont}
   {\ifnum \tracingfonts>\tw@
       \@font@info{Switching to \font@name}\fi}
⟨/trace⟩
```

```
⟨/2ekernel | latexrelease | package⟩
⟨latexrelease⟩\EndIncludeInRelease
```

With `\selectfont` having different definitions in different kernels we also have to provide them in the `tracefnt` package to support rollback. In packages that works a bit differently and therefore we have to provide an empty block there.

```
⟨package⟩\IncludeInRelease{2021/06/01}%
⟨package⟩                  {\selectfont}{Add hook to \selectfont}%
⟨package⟩\EndIncludeInRelease
```

```
⟨latexrelease | package⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | package⟩                  {\selectfont}{Add hook to \selectfont}%
⟨latexrelease | package⟩
⟨latexrelease | package⟩\DeclareRobustCommand\selectfont
⟨latexrelease | package⟩   {%
⟨latexrelease | package⟩    \ifx\f@linespread\baselinestretch \else
⟨latexrelease | package⟩      \set@fontsize\baselinestretch\f@size\f@baselineskip \fi
⟨latexrelease | package⟩    \xdef\font@name{%
⟨latexrelease | package⟩      \csname\curr@fontshape/\f@size\endcsname}%
⟨latexrelease | package⟩    \pickup@font
⟨latexrelease | package⟩    \font@name
⟨latexrelease | package⟩    \size@update
⟨latexrelease | package⟩    \enc@update
⟨latexrelease | package⟩   }
⟨latexrelease | package⟩
⟨latexrelease | package⟩\EndIncludeInRelease
```

`\set@fontsize` The macro `\set@fontsize` does the actual work. First it assigns new values to `\f@size`, `\f@baselineskip` and `\f@linespread`.

```
⟨*2ekernel | package⟩
\def\set@fontsize#1#2#3{%
    \@defaultunits\@tempdimb#2pt\relax\@nnil
```

```
    \edef\f@size{\strip@pt\@tempdimb}%
    \@defaultunits\@tempskipa#3pt\relax\@nnil
    \edef\f@baselineskip{\the\@tempskipa}%
    \edef\f@linespread{#1}%
```

For backward compatibility and for later testing within `\selectfont` the internal value of `\f@linespread` is passed back to `\baselinestretch`.

```
    \let\baselinestretch\f@linespread
```

Additional processing will happen within `\selectfont`. For this reason the macro `\size@update` (which will be called in `\selectfont`) will be defined to be:

```
      \def\size@update{%
```

First calculate the new `\baselineskip` and also store it in `normalbaselineskip`

```
        \baselineskip\f@baselineskip\relax
        \baselineskip\f@linespread\baselineskip
        \normalbaselineskip\baselineskip
```

then to set up a new `\strutbox`

```
        \setbox\strutbox\hbox{%
          \vrule\@height.7\baselineskip
                \@depth.3\baselineskip
                \@width\z@}%
```

We end with a bit of tracing information.

```
⟨*trace⟩
   \ifnum \tracingfonts>\tw@
      \ifx\f@linespread\@empty
        \let\reserved@a\@empty
      \else
        \def\reserved@a{\f@linespread x}%
      \fi
      \@font@info{Changing size to \f@size/\reserved@a
                \f@baselineskip}%
      \aftergroup\type@restoreinfo \fi
⟨/trace⟩
```

When all this is processed `\size@update` redefines itself to `\relax` so that in later calls of `\selectfont` no extra code will be executed.

```
        \let\size@update\relax}%
  }
```

Instead of defining this macro internally we might speed things up by placing the code into a separate macro and use `\let`!

(*End of definition for* `\set@fontsize`.)

`\size@update` Normally this macro does nothing; it will be redefined by `\set@fontsize` to initiate an update.

```
\let\size@update\relax
```

(*End of definition for* `\size@update`.)

\type@restoreinfo This macro produces some info when a font size and/or baseline change will get restored.

```
⟨*trace⟩
  \def\type@restoreinfo{%
    \ifx\f@linespread\@empty
      \let\reserved@a\@empty
    \else
      \def\reserved@a{\f@linespread x}%
    \fi
    \@font@info{Restoring size to
                \f@size/\reserved@a\f@baselineskip}}
⟨/trace⟩
```

(*End of definition for* \type@restoreinfo.)

\glb@settings The macro \glb@settings globally selects all math fonts for the current size if necessary.
\glb@currsize

```
\def\glb@settings{%
```

When \glb@settings gains control a size change was requested and all previous font assignments need to be replaced. Therefore the old values of the fonts are no longer needed. For every *math group* the new assignments are appended to \math@fonts. But this happens only if the math@fonts switch is set to true. However, we always set up the correct math sizes for script and scriptscript fonts since they may be needed even if we don't set up the whole math machinery.

Here we set the math size, script size and scriptscript size. If the S@... macro is not defined we have to first calculate the three sizes.

```
    \expandafter\ifx\csname S@\f@size\endcsname\relax
      \calculate@math@sizes
    \fi
```

The effect of this is that \calculate@math@sizes may or may not define the S@... macro. In the first case the next time the same size is requested this macro is used, otherwise \calculate@math@sizes is called again. This also sets the math@fonts switch. If it is true we must switch the math fonts.

```
    \csname S@\f@size\endcsname
    \ifmath@fonts
⟨*trace⟩
      \ifnum \tracingfonts>\tw@
        \@font@info{Setting up math fonts for
                    \f@size/\f@baselineskip}\fi
⟨/trace⟩
```

Inside a group we execute the macro for the current math *version*. This sets \math@fonts to a list of \textfont... assignments. \getanddefine@fonts (which may be called at this point) needs the \escapechar parameter to be set to −1.

```
      \begingroup
        \escapechar\m@ne
        \csname mv@\math@version \endcsname
```

Then we set \globaldefs to 1 so that all following changes are done globally. The math font assignments recorded in \math@fonts are executed and \glb@currsize is set equal to \f@size. This signals that the fonts for math in this size are set up.

```
        \globaldefs\@ne
        \math@fonts
```

```
          \let \glb@currsize \f@size
        \endgroup
```

Finally we execute any code that is supposed to happen whenever the math font setup changes. This register will be executed in local mode which means that everything that is supposed to have any effect should be done globally inside. We can't execute it within `\globaldefs\@ne` as we don't know what ends up inside this register, e.g., it might contain calculations which use some local registers to calculate the final (global) value.

```
         \the\every@math@size
```

Otherwise we announce that the math fonts are not set up for this size.

```
⟨*trace⟩
        \else
          \ifnum \tracingfonts>\tw@
            \@font@info{No math setup for
                        \f@size/\f@baselineskip}\fi
⟨/trace⟩
      \fi
}
⟨/2ekernel | package⟩
```

(*End of definition for* `\glb@settings` *and* `\glb@currsize`.)

`\baselinestretch` In `\selectfont` we used `\baselinestretch` as a factor when assigning a value to `\baselineskip`. We use 1 as a default (i.e. no stretch).

```
⟨*2ekernel⟩
\def\baselinestretch{1}
```

(*End of definition for* `\baselinestretch`.)

`\every@math@size` We must still define the hook `\every@math@size` we used in `\glb@settings`. We initialize it to nothing. It is important to remember that everything that goes into this hook should to global updates, local changes will have weird effects.

```
\newtoks\every@math@size
\every@math@size={}
⟨/2ekernel⟩
```

(*End of definition for* `\every@math@size`.)

5.2 Math fonts setup

5.2.1 Outline of algorithm for math font sizes

TEX uses the math fonts that are current when the end of a formula is reached. If we don't want to keep font setups local to every formula (which would result in an enormous overhead), we have to be careful not to end up with the wrong setup in case formulas are nested, e.g., we need to be able to handle

```
$ a=b+c \mbox{ \small for all $b$ and $c\in Z$}$
```

Here the inner formulae `b` and `c\in Z` are typeset in `\small` but we have to return to `\normalsize` before we reach the closing `$` of the outer formula.

This is handled in the following way:

1. At any point in the document the global variable `\gbl@currsize` contains the point size for which the math fonts currently are set up.

2. Whenever we start a formula we compare its value with the local variable `\f@size` that describes the current text font size.

3. If both are the same we assume that we can use the current math font setup without adjustment.

4. If they differ we call `\gbl@settings` which changes the math font setup and updates `\gbl@currsize`.

   (a) If we are recursively inside another formula (`\if@inmath`) we ensure that `\gbl@settings` is executed again in the outer formula, so that the old setup is automatically restored.

   (b) Otherwise, we set the switch `@inmath` locally to `true` so that all nested formulae will be able to detect that they are nested in some outer formula.

The above algorithm has the following features:

- For sizes which are not containing any formula no math setup is done. Compared to the original algorithm of NFSS this results in the following savings:
  - No unnecessary loading of math fonts for sizes that are not used to typeset any math formulae (explicit or implicit ones).
  - No time overhead due to unnecessary changes of the math font setup on entrance and exit of the text font size.
- Math font setup changes for top-level formulae will survive (there is no restoration after the formula) thus any following formula in the same size will be directly typesetable. Compared to original implementation in NFSS2 the new algorithm has the overhead of one test per formula to see if the current math setup is valid (in the original algorithm the setup was always valid, thus no test was necessary).
- In nested formulae the math font setup is restored in the outer formula by a series of `\aftergroup` commands and checks. Compared to the original algorithm this involves additional checks ($2 \times \langle$non-math levels$\rangle$ per inner formula).

5.2.2 Code for math font size setting

`\check@mathfonts` In the `\check@mathfonts` macros we implement the steps 2 to 4 except that instead of a switch the macro `\init@restore@glb@settings` is used.

```
⟨*2ekernel | package⟩
\def\check@mathfonts{%
  \ifx \glb@currsize \f@size
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: no change \f@size\space
         curr/global (\curr@math@size/\glb@currsize)}\fi
⟨/trace⟩
  \else
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: setting up \f@size\space
         curr/global (\curr@math@size/\glb@currsize)}\fi
⟨/trace⟩
```

```
    \glb@settings
    \init@restore@glb@settings
  \fi
  \let\curr@math@size\f@size
  \def\init@restore@glb@settings{\aftergroup\restglb@settings}%
}
```

(*End of definition for* `\check@mathfonts`.)

`\init@restore@glb@settings` This macros does by default nothing but get redefined inside `\check@mathfonts` to initiate fontsize restoring in nested formulas.

```
⟨-trace⟩\let\init@restore@glb@settings\relax
⟨*trace⟩
\def\init@restore@glb@settings{%
       \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: no resetting (not in
                    nested math)}\fi
}
⟨/trace⟩
```

(*End of definition for* `\init@restore@glb@settings`.)

`\restglb@settings` This macro will be executed the first time after the current formula.

```
\def\restglb@settings{%
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: restoring}\fi
⟨/trace⟩
      \begingroup
        \let\f@size\curr@math@size
        \ifx\glb@currsize \f@size
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: ... already okay (\f@size)}\fi
⟨/trace⟩
        \else
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: ... to \f@size}\fi
⟨/trace⟩
          \glb@settings
        \fi
      \endgroup
}
```

(*End of definition for* `\restglb@settings`.)

5.2.3 Other code for math

`\use@mathgroup` The `\use@mathgroup` macro should be used in user macros to select a math group. Depending on whether or not the `margid` option is in force it has two or three arguments. For this reason it should be called as the last macro.

First we test if we are inside math mode since we don't want to apply a useless definition.

```
\def\use@mathgroup#1#2{\relax\ifmmode
```

```
⟨*trace⟩
  \ifnum \tracingfonts>\tw@
    \count@#2\relax
    \@font@info{Using \noexpand\mathgroup
              (\the\count@) #2}\fi
⟨/trace⟩
```

If so we first call the '=' macro (i.e. argument three) to set up special things for the selected math group. Then we call `\mathgroup` to select the group given by argument two and finally we place `#1` (i.e., the argument of the ⟨*math alphabet identifier*⟩) at the end. This part of the code is surrounded by two commands which behave like `\begingroup` and `\endgroup` if we want ⟨*math alphabet identifier*⟩s but will expand into `\@empty` if we want simply switches to a new math group. Since argument number 2 may be a digit instead of a control sequence we add a `\relax`. Otherwise something like `\mit{1}` would switch to math group 11 (and back) instead of printing an oldstyle `1`.

```
    \math@bgroup
        \expandafter\ifx\csname M@\f@encoding\endcsname#1\else
        #1\fi
        \mathgroup#2\relax
```

Before we reinsert the swallowed token (arg. three) into the input stream, in the case that the ⟨*math alphabet identifier*⟩ isn't called in math mode, we remove the `\fi` with the `\expandafter` trick. This is necessary if the token is actually an macro with arguments. In such a case the `\fi` will be misinterpreted as the first argument which would be disastrous.

```
    \expandafter\math@egroup\fi}%
```

The surrounding macros equal `\begingroup` and `\endgroup`. But using internal names makes it possible to overwrite their meaning in certain cases. This is for example used in $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX macros for placing accents.

(*End of definition for* `\use@mathgroup`.)

`\math@egroup` If the `margid` option is in force (which can be tested by looking at the definition of `\math@bgroup`) we change the `\math@egroup` command a bit to display the current ⟨*math group number*⟩ after it closes the scope of ⟨*math alphabet*⟩ with `\endgroup`.

```
⟨*trace⟩
  \ifx\math@bgroup\bgroup
    \def\math@egroup#1{#1\egroup
      \ifnum \tracingfonts>\tw@
     \@font@info{Restoring \noexpand\mathgroup
        (\ifnum\mathgroup=\m@ne default\else \the\mathgroup \fi)%
        }\fi}
  \fi
⟨/trace⟩
```

(*End of definition for* `\math@egroup`.)

`\getanddefine@fonts` `\getanddefine@fonts` has two arguments: the ⟨*math group number*⟩ and the *family/series/shape* name as a control sequence.

```
\def\getanddefine@fonts#1#2{%
```

First we turn of tracing when \tracingfonts is less than 4.

```
⟨+debug⟩  \pushtracing
⟨+debug⟩  \ifnum\tracingfonts<4 \tracingoff
⟨+debug⟩  \else \tracingon\getanddefine@fonts \fi
```

```
⟨*trace⟩
  \ifnum \tracingfonts>\tw@
  \count@#1\relax
    \@font@info{\noexpand\mathgroup (\the\count@) #1 :=\MessageBreak
                \string#2 \tf@size/\sf@size/\ssf@size}\fi
⟨/trace⟩
```

We append the current \tf@size to #2 to obtain the font name.[35] Again, font@name is defined globally, for the reasons explained in the description of \wrong@fontshape.

```
  \xdef\font@name{\csname \string#2/\tf@size\endcsname}%
```

Then we call \pickup@font to load it if necessary. We remember the internal name as \textfont@name.

```
  \pickup@font \let\textfont@name\font@name
```

Same game for \scriptfont and \scriptscriptfont:

```
  \xdef\font@name{\csname \string#2/\sf@size\endcsname}%
  \pickup@font \let\scriptfont@name\font@name
  \xdef\font@name{\csname \string#2/\ssf@size\endcsname}%
  \pickup@font
```

Then we append the new \textfont... assignments to the \math@fonts.

```
  \edef\math@fonts{\math@fonts
                \textfont#1\textfont@name
                \scriptfont#1\scriptfont@name
                \scriptscriptfont#1\font@name}%
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
⟨+debug⟩ \poptracing
    }
⟨/2ekernel | package⟩
```

(*End of definition for* \getanddefine@fonts.)

6 Scaled font extraction

\ifnot@nil We begin with a simple auxiliary macro. It checks whether its argument is the token \@nil. If so, it expands to \@gobble which discards the following argument, otherwise it expands to \@firstofone which reproduces it argument.

```
⟨*2ekernel⟩
\def\ifnot@nil#1{\def\reserved@a{#1}%
  \ifx\reserved@a\@nnil \expandafter\@gobble
  \else \expandafter\@firstofone\fi}
```

(*End of definition for* \ifnot@nil.)

[35] One might ask why this expansion does not generate a macro name that starts with an additional \ character. The solution is that \escapechar is set to −1 before \getanddefine@fonts is called.

\remove@to@nnil
\remove@angles
\remove@star

Three other auxiliary macros will be needed in the following: \remove@to@nnil gobbles up everything up to, and including, the next \@nnil token, and \remove@angles and \remove@star do the same for the character > and *, respectively, instead of \@nnil.

```
\def\remove@to@nnil#1\@nnil{}
\def\remove@angles#1>{\set@simple@size@args}
\def\remove@star#1*{#1}
```

(*End of definition for* \remove@to@nnil*,* \remove@angles*, and* \remove@star*.*)

\extract@sizefn

This macro takes a size specification and parses it into size function and the optional and mandatory arguments.

```
\def\extract@sizefn#1*#2\@nil{%
  \if>#2>\set@size@funct@args#1\@nil
         \let\sizefn@info\@empty
  \else\expandafter\set@size@funct@args\remove@star#2\@nil
       \def\sizefn@info{#1}\fi
  }
```

(*End of definition for* \extract@sizefn*.*)

\try@simple@size

This function tries to extract the given size (specified by \f@size) for the requested font shape. The font information must already be present in \font@info. The central macro that does the real work is \extract@fontinfo. We will first give a simple example how this macro works, and describe it in full generality later.

Assume that the requested parameters are: *encoding scheme* 'OT1', *family* 'cm', *series* 'sansserif', *shape* 'normal', and *size* '12'. The corresponding font definitions have already been extracted from the macro \OT1/cm/sansserif/normal and stored in font@info. (Otherwise \extract@fontinfo doesn't get called.) This information consists of a token list made of characters of category code 12 of the form

```
<10*>cmss10<12*>cmss12<17*>cmss17
```

For reasonable packages one usually needs more sizes but this is sufficient to get the flavour. We will define a macro \extract@fontinfo to find the external font name ('cmss12') for us:

```
\def\extract@fontinfo#1<12*#2>#3<#4\@nnil{%
    \set@simple@size@args#3<#4\@nnil
    \execute@size@function{#2}}
```

so that when it gets called via

```
\extract@fontinfo<10*>cmss10<12*>cmss12<17*>cmss17\@nnil
```

#1 will contain all characters before <12*>, #2 will be empty, #3 will be exactly cmss12, and #3 will be 17>cmss17. The expansion is therefore

```
\set@simple@size@args cmss12<17*>cmss17\@nnil
\execute@size@function{}
```

This means: the default (empty) size function will be executed, with its optional argument set to empty and its mandatory argument set to `cmss12` by `\set@simple@size@args`. As we discussed earlier, the effect of the default size function is to load the given external font (cmss12) at the specified size (12)—which is exactly what was intended.

But this is only part of the whole story. It may be that the size requested does not occur in the token list `\font@info`. And the simple definition of `\extract@fontinfo` we gave above does not allow to specify give more than one size specification in front of the external font name.

Let's address these two problems separately. The first one is solved with the following trick: We define `\extract@fontinfo` as follows:

```
\def\extract@fontinfo#1<12*#2>#3<#4\@nnil{%
  \ifnot@nil{#3}%
    {\set@simple@size@args#3<#4\@nnil
     \execute@size@function{#2}%
    }}%
```

How does this work? We call `\extract@fontinfo` via

```
\expandafter\extract@fontinfo\font@info<12*>\@nil<\@nnil
```

i.e. by appending `<12*>\@nil<\@nnil`. If the size ('12' in this case) appears in `\font@info` everything works as explained above, the only difference being that argument `#4` of `\extract@fontinfo` additionally gets the tokens `<12*>\@nil<\@nnil`. However, if the size is not found everything up to the final `<12*>` is in argument `#1`, `#3` gets `\@nil`, and `#2` and `#4` are empty. The macro `\ifnot@nil` will discard the calls to `\set@simple@size@args` and `execute@size@function`, and hence `\font@info` will continue to be equal to `\@empty`. This means that no simple size specification matching the requested size could be found.

The second problem (more than one simple size specification for one external font name) will be addressed in `\set@simple@size@args` below.

The macros are hidden inside other control sequences so that we have to build `\extract@fontinfo` in several steps.

So here's the actual definition of `\extract@font` in `\try@simple@size`.

```
% % this could be replaced by \try@size@range making the subst slower!
\def\try@simple@size{%
```

`\reserved@a` is made an abbreviation for the head of the definition of the macro `\extract@fontinfo`.

```
    \def\reserved@a{\def\extract@fontinfo####1}%
```

Now we can define `\extract@fontinfo`. Here we handle a small but convenient variation: in case of the default (empty) size function it is allowed to omit the `*` character.

```
    \expandafter\reserved@a\expandafter<\f@size>##2<##3\@nnil{%
          \ifnot@nil{##2}%
            {\set@simple@size@args##2<##3\@nnil
             \execute@size@function\sizefn@info
            }}%
```

Now we call `\extract@fontinfo`. Note the `<\@nil` tokens at the end.

```
    \expandafter\expandafter
    \expandafter\extract@fontinfo\expandafter\font@info
    \expandafter<\f@size>\@nil<\@nnil
}
```

(*End of definition for* `\try@simple@size`.)

`\set@simple@size@args` As promised above, the macro `\set@simple@size@args` will handle the case of several size specifications in a row. If another size specification follows, the very first token of its argument list is the character `<`. By starting the definition as follows,

```
\def\set@simple@size@args#1<{%
```

parameter `#1` is empty in this case, and contains the size function's arguments otherwise. We distinguish these two cases (Note that the character `<` cannot appear in `#1`) by calling `\remove@angles` for empty `#1` and `\extract@sizefn` otherwise. In the latter case we have to take care of the remaining character tokens and discard them. This is done by `\remove@to@nnil`. Note also the use of Kabelschacht's method.

```
          \if<#1<%
            \expandafter\remove@angles
          \else
            \extract@sizefn#1*\@nil
            \expandafter\remove@to@nnil
          \fi}
```

(*End of definition for* `\set@simple@size@args`.)

Now, we are through with the case of a simple size, except for calling the size function. This will be handled later, as it is the same mechanism for all types of size specification. We will now proceed to macros for extraction of size range specification.

`\extract@rangefontinfo` `\extract@rangefontinfo` goes through a font shape definition in the input until it recognizes the tokens `<\@nil->`. It looks for font ranges with font size functions. It's operation is rather simple: it discards everything up to the next size specification and passes this on to `\is@range` for inspection. The specification (parameter `#2`) is inserted again, in case it is needed later.

```
\def\extract@rangefontinfo#1<#2>{%
       \is@range#2->\@nil#2>}
```

(*End of definition for* `\extract@rangefontinfo`.)

`\is@range` `\is@range` is again a sort of dispatcher macro: if the size specification it is looking at is not a range specification it discards it and calls `\extract@rangefontinfo` to continue the search. Otherwise it calls `\check@range` to check the requested size against the specified range.

From the way `\is@range` is called inside `\extract@rangefontinfo` we see that `#2` is the character `>` if the size specification found is a simple one (that does not contain a `-` character). This is checked easily enough and `\extract@rangefontinfo` called again. Note that the extra tokens inserted after the `\@nil` in the call to `\is@range` appear at the beginning of the first argument to `\extract@rangefontinfo` and are hence ignored.

```
\def\is@range#1-#2\@nil{%
   \if>#2\expandafter\check@single\else
      \expandafter\check@range\fi}
```

(*End of definition for* `\is@range`.)

`\check@range` `\check@range` takes lower bound as parameter `#1`, upper bound as `#2`, size function as `#3` and the size function's arguments as `#4`. If `#3` is the special token `\@nil` `\font@info` is exhausted and we can stop searching.

```
\def\check@range#1-#2>#3<#4\@nnil{%
  \ifnot@nil{#3}{%
```

If #3 wasn't \@nil we have a range. We start by assuming that we have to recurse. Note that we have to reinsert an < as it was already removed by scanning.

```
      \def\reserved@f{\extract@rangefontinfo<#4\@nnil}%
```

We have to make sure that both boundaries are present, if not we have to set them. Here we check the upper bound. If \upper@bound is zero after the assignment we set it to \maxdimen (upper open range). We need to use a ⟨*dimen*⟩ register for the scan since we may have a decimal number as the boundary.

```
      \upper@bound0#2\p@
      \ifdim\upper@bound=\z@ \upper@bound\maxdimen\fi
```

Now we check the upper boundary against \f@size. If it is larger or equal than \f@size this range is no good and we have to recurse.

```
      \ifdim \f@size \p@<\upper@bound
```

Otherwise we have to check the lower bound. This time it is not necessary to scan the boundary value into a register because if it is empty we get zero as desired. We could even omit the 0 which would result in 1pt as default lower boundary. If \f@size is smaller than the boundary we have to recurse.

```
        \lower@bound0#1\p@
        \ifdim \f@size \p@<\lower@bound
         \else
```

If both tests are passed we can try executing the size function.

```
          \set@simple@size@args#3<#4\@nnil
          \execute@size@function\sizefn@info
```

If the function was successful it should have left an external font name in \external@font. We use this to see if we can stop scanning. Otherwise we recurse.

```
          \ifx\external@font\@empty
          \else
            \let\reserved@f\@empty
          \fi
        \fi
      \fi
      \reserved@f}}
```

(*End of definition for* \check@range.)

\lower@bound
\upper@bound

We use two dimen registers \lower@bound and \upper@bound to store the lower and upper endpoints of the range we found.

```
\newdimen\lower@bound
\newdimen\upper@bound
```

(*End of definition for* \lower@bound *and* \upper@bound.)

\check@single

\check@single takes the size as parameter #1, size function as #2 and the size function's arguments as #3. We can assume that there is always something in the pipeline since the very last entry is a faked range (see above).

```
\def\check@single#1>#2<#3\@nnil{%
```

We start by assuming that we have to recurse. Note that we have to reinsert an < as it was already removed by scanning.

```
      \def\reserved@f{\extract@rangefontinfo<#3\@nnil}%
```

Now we check the size against \f@size. If it is not equal \f@size it is no good and we have to recurse.

```
    \ifdim \f@size \p@=#1\p@
```

Otherwise if this test is passed we can try executing the size function.

```
        \set@simple@size@args#2<#3\@nnil
        \execute@size@function\sizefn@info
```

If the function was successful it should have left an external font name in \external@font. We use this to see if we can stop scanning. Otherwise we recurse.

```
        \ifx\external@font\@empty
        \else
          \let\reserved@f\@empty
        \fi
    \fi
    \reserved@f}
```

(*End of definition for* \check@single.)

\set@size@funct@args
\set@size@funct@args@

This macro sets the optional and mandatory arguments for a size function. If the optional argument is not present it is set to the empty token list. The mandatory argument is delimited by the token \@nil.

```
\def\set@size@funct@args{\@ifnextchar[%
  \set@size@funct@args@{\set@size@funct@args@[]}}
```

```
\def\set@size@funct@args@[#1]#2\@nil{%
  \def\mandatory@arg{#2}%
  \def\optional@arg{#1}}
⟨/2ekernel⟩
```

(*End of definition for* \set@size@funct@args *and* \set@size@funct@args@.)

\DeclareSizeFunction

This function defines a new size function hiding the internal from the designer. The body of the size function may use \optional@arg and \mandatory@arg denoting the optional and mandatory argument that may follow the size specification <...>.

```
⟨*2ekernel⟩
\def\DeclareSizeFunction#1#2{\@namedef{s@fct@#1}{#2}}
\@onlypreamble\DeclareSizeFunction
⟨/2ekernel⟩
```

(*End of definition for* \DeclareSizeFunction.)

\execute@size@function

This macro is very simple. The only point worth noting is that calling an undefined size function will do nothing (actually execute a \relax).

```
⟨*2ekernel | package⟩
\def\execute@size@function#1{%
⟨*trace⟩
          \@ifundefined{s@fct@#1}%
            {\errmessage{Undefined font size function #1}%
             \s@fct@}%
            {\csname s@fct@#1\endcsname}%
⟨/trace⟩
⟨-trace⟩       \csname s@fct@#1\endcsname
}
⟨/2ekernel | package⟩
```

(*End of definition for* `\execute@size@function`.)

`\try@size@range` This macro tries to find a suitable range for requested size (specified by `\f@size`) in `\font@info`. All the relevant action is done in `\extract@rangefontinfo`. All that needs to be done is to stuff in the token list in `\font@info` so that `\extract@rangefontinfo` can inspect it. Note the `<-*\@nil><` token at the end to stop scanning.

```
⟨*2ekernel⟩
\def\try@size@range{%
    \expandafter\extract@rangefontinfo\font@info <-*>\@nil<\@nnil
}
```

(*End of definition for* `\try@size@range`.)

`\try@size@substitution` This is the last thing that can be tried. If the desired `\f@size` is found neither among the simple size specifications nor in one of the ranges the whole list of size specifications is searched for a nearby simple size.

```
\gdef\try@size@substitution{%
```

First we do some initializations. `\@tempdimb` will hold the difference between the wanted size and the best solution found so far, so we initialise it with `\maxdimen`. The macro `\best@size` will hold the best size found, nothing found is indicated by the empty value.

```
  \@tempdimb \maxdimen
  \let \best@size \@empty
```

Now we loop over the specification

```
  \expandafter \try@simples \font@info <\number\@M>\@nil<\@nnil
}
```

(*End of definition for* `\try@size@substitution`.)

`\font@submax` `\fontsubfuzz` The macro `\font@submax` records the maximal deviation from the desired size encountered so far. Its value is used in a warning message at `\end{document}`. The macro `\fontsubfuzz` contains the amount that will not cause terminal warnings (warnings still go into the transcript file).

```
\def\font@submax{0pt}
\def\fontsubfuzz{.4pt}
⟨/2ekernel⟩
⟨+package⟩\def\fontsubfuzz{0pt}
```

(*End of definition for* `\font@submax` *and* `\fontsubfuzz`.)

`\try@simples` `\try@simples` goes through a font shape definition in the input until it recognizes the tokens `<*\@nil><`. It looks for simple sizes to determine the two closest sizes. It is assumed that simple sizes are in increasing order.

```
⟨*2ekernel⟩
\gdef\try@simples#1<#2>{%
  \tryif@simple#2->\tryif@simple}
```

(*End of definition for* `\try@simples`.)

\tryis@simple

\tryis@simple is similar to \is@range. If it sees a simple size, it checks it against the value of \f@size and sets \lower@font@size or \higher@font@size. In the latter case, it stops the iteration. By adding <\number\@M> at the end of the line we always have an end point. This is a hack which probably should be corrected.

First it checks whether it is finished already, then whether the size specification in question is a simple one.

```
\gdef\tryif@simple#1-#2\tryif@simple{%
```

Most common case for \reserved@f first:

```
  \let \reserved@f \try@simples
  \if>#2%
```

If so, it compares it to the value of \f@size. This is done using a dimen register since there may be fractional numbers.

```
    \dimen@ #1\p@
    \ifdim \dimen@<\@M\p@
```

If \dimen@ is \@M\p@ we have reached the end of the fontspec (hopefully) otherwise we compare the value with \f@size and compute in \@tempdimc the absolute value of the difference between the two values.

```
      \ifdim \f@size\p@<\dimen@
        \@tempdimc \dimen@
        \advance\@tempdimc -\f@size\p@
      \else
        \@tempdimc \f@size\p@
        \advance\@tempdimc -\dimen@
      \fi
```

The result is then compared with the smallest difference we have encountered, if the new value (in \@tempdimc is smaller) we have found a size which is a better approximation so we make it the \best@size and adjust \@tempdimb.

```
      \ifdim \@tempdimc<\@tempdimb
        \@tempdimb \@tempdimc
        \def \best@size{#1}%
      \fi
```

When we have reached the end of the fontspec we substitute the best size found (if any). We code this inline to save macro space; in the past this was done by a macro called \subst@size.

```
    \else
```

\subst@size

This macro substitutes the size recorded in \best@size for the unavailable size \f@size. \font@submax records the maximum difference between desired size and selected size in the whole run.

```
% %\subst@size             %% coded inline
% %\def\subst@size{%
  \ifx \external@font\@empty
    \ifx \best@size\@empty
    \else
      \ifdim \@tempdimb>\font@submax \relax
        \xdef \font@submax {\the\@tempdimb}%
      \fi
      \let \f@user@size \f@size
      \let \f@size \best@size
```

```
      \ifdim \@tempdimb>\fontsubfuzz\relax
        \@font@warning{Font\space shape\space
            `\curr@fontshape'\space in\space size\space
             <\f@user@size>\space not\space available\MessageBreak
             size\space <\f@size>\space substituted}%
      \fi
      \try@simple@size
      \do@subst@correction
    \fi
  \fi
% %}
```

This brings us back into the main part of `\tryif@simple`. Finally we get rid of any rubbish left over on the input stack.

```
      \let \reserved@f \remove@to@nnil
    \fi
  \fi
```

If it's a range iterate also.

```
  \reserved@f}
```

(*End of definition for* `\tryis@simple` *and* `\subst@size`.)

6.1 Sizefunctions

In the following we define some useful size functions.

`\s@fct@` This is the default size function. Mandatory argument is an external font name, optional argument a scale factor. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```
\DeclareSizeFunction{}{\empty@sfcnt\@font@warning}
\DeclareSizeFunction{s}{\empty@sfcnt\@font@info}
```

```
\def\empty@sfcnt#1{%
      \@tempdimb \f@size\p@
      \ifx\optional@arg\@empty
      \else
        \@tempdimb \optional@arg\@tempdimb
        #1{Font\space shape\space `\curr@fontshape'\space
           will\space be\MessageBreak
           scaled\space to\space size\space \the\@tempdimb}%
      \fi
      \edef\external@font{\mandatory@arg\space at\the\@tempdimb}}
```

(*End of definition for* `\s@fct@`.)

`\s@fct@gen` `\s@fct@sgen` This size function generates the external name from the mandatory argument and the requested user size, and thus can be used for external names where the size is encoded in the font name. The optional argument a scale factor. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```
\DeclareSizeFunction{gen}{\gen@sfcnt\@font@warning}
\DeclareSizeFunction{sgen}{\gen@sfcnt\@font@info}
```

```
\def\gen@sfcnt{%
      \edef\mandatory@arg{\mandatory@arg\f@size}%
      \empty@sfcnt}
```

(*End of definition for* `\s@fct@gen` *and* `\s@fct@sgen`.)

`\s@fct@genb` `\s@fct@sgenb`

This size function is similar to `gen`, but for fonts where the size is encoded in the font name in centipoints, as in the DC fonts version 1.2. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```
\DeclareSizeFunction{genb}{\genb@sfcnt\@font@warning}
\DeclareSizeFunction{sgenb}{\genb@sfcnt\@font@info}
```

```
\def\genb@sfcnt{%
    \edef\mandatory@arg{\mandatory@arg\expandafter\genb@x\f@size..\@@}%
    \empty@sfcnt}
```

(*End of definition for* `\s@fct@genb` *and* `\s@fct@sgenb`.)

`\genb@x` `\genb@y`

The auxiliary macros `\genb@x` and `\genb@y` are used to convert the `\f@size` into centipoints.

```
\def\genb@x#1.#2.#3\@@{\two@digits{#1}\genb@y#200\@@}
\def\genb@y#1#2#3\@@{#1#2}
```

(*End of definition for* `\genb@x` *and* `\genb@y`.)

`\s@fct@sub`

This size function handles font substitution. The mandatory argument is a family/series/shape combination, the optional argument (if present) is ignored. The font encoding scheme cannot be changed. Therefore, the first thing we do is to prepend the encoding scheme.

```
\DeclareSizeFunction{sub}{\sub@sfcnt\@font@warning}
\DeclareSizeFunction{ssub}{\sub@sfcnt\@font@info}
```

```
\def\sub@sfcnt#1{%
    \edef\mandatory@arg{\f@encoding/\mandatory@arg}%
```

Next action is split the arg into its individual components and allow for a late font shape load.

```
    \begingroup
     \expandafter\split@name\mandatory@arg/\@nil
     \try@load@fontshape
    \endgroup
```

Then we record the current `\f@size` since it may get clobbered.

```
    \let\f@user@size\f@size
```

Then we check whether this new combination is defined and give an error message if not. In this case we also switch to `\error@fontshape`.

```
    \expandafter
    \ifx\csname\mandatory@arg\endcsname\relax
      \errmessage{No\space declaration\space for\space
                  shape\space \mandatory@arg}%
      \error@fontshape
    \else
```

Otherwise we warn the user about the substitution taking place.

```
      #1{Font\space shape\space `\curr@fontshape'\space in\space
         size\space <\f@size>\space not\space available\MessageBreak
         Font\space shape\space `\mandatory@arg'\space tried\space
         instead}%
      \expandafter\split@name\mandatory@arg/\@nil
    \fi
```

Then we restart the font specification scan by calling `\get@external@font`.

```
    \edef\f@size{\f@user@size}%
    \get@external@font
```

Finally `\do@subst@correction` is called to get the font name right.

```
    \do@subst@correction
}
```

(*End of definition for* `\s@fct@sub`.)

`\@font@aliasinfo` Sometimes a substitution is only done to map a long font name to a standard shape or series, e.g.,

```
DeclareFontShape{T1}{Roboto-LF}{b}{it}{<-> alias * Roboto-LF/bold/it}{}
```

Using the `ssub` function in that case will give a strange (and incorrect) warning. As an alternative we therefore offer the size function `alias`. It will still add some info into the `.log` file, but no longer complains that the font shape is not available. It is implemented by grabbing the default warning text and replacing it with a new one.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {\@font@aliasinfo}{alias size function}%
\DeclareSizeFunction{alias}{\sub@sfcnt\@font@aliasinfo}
\def\@font@aliasinfo#1{%
  \@font@info{Font\space shape\space `\curr@fontshape'\space
              aliased\space to\MessageBreak `\mandatory@arg'}%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@font@aliasinfo}{alias size function}%
⟨latexrelease⟩\let\s@fct@alias\@undefined
⟨latexrelease⟩\let\@font@aliasinfo\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@font@aliasinfo`.)

`\s@fct@subf` The `subf` size function allows substitution of another font. The mandatory argument is the external name of the font to be substituted, the optional argument a size scaling factor like in the default size function. The main difference to the default size function is the warning message.

```
\DeclareSizeFunction{subf}{\subf@sfcnt\@font@warning}
\DeclareSizeFunction{ssubf}{\subf@sfcnt\@font@info}
```

```
\def\subf@sfcnt#1{%
      #1{Font\space shape\space `\curr@fontshape'\space in\space
         size\space \f@size\space not\space available\MessageBreak
        external\space font\space `\mandatory@arg'\space used}%
      \empty@sfcnt#1%
      }
```

(*End of definition for* `\s@fct@subf`.)

`\s@fct@fixed` The `fixed` size function is for using a font at a different size than requested. A warning message is printed, and the external font to be used is taken from the mandatory argument. If an optional argument is present it is used as the 'at' size for the font. Otherwise the font is loaded at its design size.

```
\DeclareSizeFunction{fixed}{\fixed@sfcnt\@font@warning}
\DeclareSizeFunction{sfixed}{\fixed@sfcnt\@font@info}
```

```
\def\fixed@sfcnt#1{%
  \ifx\optional@arg\@empty
    \let\external@font\mandatory@arg
  \else
    \edef\external@font{\mandatory@arg\space at\optional@arg pt}%
  \fi
  #1{External\space font\space `\external@font'\space loaded\space
     for\space size\MessageBreak
     <\f@size>}%
}
⟨/2ekernel⟩
```

(*End of definition for* `\s@fct@fixed`.)

File 27
ltfsscmp.dtx

This file contains the implementation of commands giving compatibility with the original 'NFSS1' release of the Font Selection Scheme.

> **Warning:** The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

Version 1 of NFSS is obsolete now for about 20 years (and was "current" only for a short intermediate time) so with the 2015 release these internal interface commands are removed from the kernel and made available via latexrelease package so that backward compatibility remains ensured for very old documents.

```
⟨*latexrelease⟩
\IncludeInRelease{2015/01/01}{\new@fontshape}%
                             {NFSS version1 commands}%
\let\new@fontshape\@undefined
\let\warn@rel@i\@undefined
\let\scan@fontshape\@undefined
\let\scan@@fontshape\@undefined
\let\subst@fontshape\@undefined
\let\extra@def\@undefined
\let\default@mextra\@undefined
\let\preload@sizes\@undefined
\let\err@rel@i\@undefined
\let\newmathalphabet\@undefined
\let\newmathalphabet@\@undefined
\let\newmathalphabet@@@\@undefined
\let\if@no@font@opt\@undefined
\let\@no@font@optfalse\@undefined
\let\define@mathalphabet\@undefined
\let\define@mathgroup\@undefined
\let\addtoversion\@undefined
\EndIncludeInRelease
```

In older releases we provide the original definitions.

```
\IncludeInRelease{0000/00/00}{\new@fontshape}%
                             {NFSS version1 commands}%
```

`\new@fontshape` The interface is now `\DeclareFontShape`.

```
\gdef\new@fontshape#1#2#3#4{%
     \warn@rel@i\new@fontshape\DeclareFontShape
     \expandafter\scan@fontshape\@gobble#4<\@nil><<%
     \DeclareFontShape U{#1}{#2}{#3}\reserved@f}%
\@onlypreamble\new@fontshape
```

(*End of definition for* `\new@fontshape`.)

`\warn@rel@i` The warning message used above.

```
\gdef\warn@rel@i#1#2{%
 \@font@warning{***  NFSS release 1 command
```

```
                \noexpand#1found\MessageBreak
   ***  Update by using release 2 command
        \string#2.\MessageBreak
   ***  Recovery is probably possible}%
}%
\@onlypreamble\warn@rel@i
```

(*End of definition for* `\warn@rel@i`.)

`\scan@fontshape` This will scan the old font shape definition syntax.

```
\gdef\scan@fontshape{%
  \let\reserved@f\@empty
  \let\reserved@e\@empty %        holds last info
  \scan@@fontshape
}%
\@onlypreamble\scan@fontshape
```

(*End of definition for* `\scan@fontshape`.)

`\scan@@fontshape`

```
\gdef\scan@@fontshape#1>#2#3<{%
  \ifx\@nil#1%
    \edef\reserved@f{\reserved@f\reserved@e}%
  \else
    \def\reserved@b{#1}%       nick names
    \def\reserved@c{#3}%
    \in@{ at}{#3}%
    \ifin@
      \in@{pt}{#3}%  not a proof but a good chance
      \ifin@
```

We grab also everything after pt and discard it if people have forgotten to place a percent sign there.

```
        \def\reserved@a##1 at##2pt##3\@nil{%
           \def\reserved@b{##2}%
           \def\reserved@c{##1}%
           }%
        \reserved@a#3\@nil
      \fi
    \fi
    \ifnum 0<0#2
      \edef\reserved@d{subf*\reserved@c}%
      \ifcase #2\or
      \or
      \else
        \errmessage{*** What's this? NFSS release 0? ***}%
      \fi
    \else
      \edef\reserved@d{#2\reserved@c}%
    \fi
    \ifx\reserved@d\reserved@e
      \edef\reserved@f{\reserved@f<\reserved@b>}%
    \else
      \edef\reserved@f{\reserved@f\reserved@e<\reserved@b>}%add old info
      \let\reserved@e\reserved@d
```

```
    \fi
    \expandafter\scan@@fontshape
  \fi
}%
\@onlypreamble\scan@@fontshape
```

(*End of definition for* `\scan@@fontshape`.)

`\subst@fontshape` This is now also handled by the extend syntax of `\DeclareFontShape`.

```
\gdef\subst@fontshape#1#2#3#4#5#6{%
     \warn@rel@i\subst@fontshape\DeclareFontShape
     \DeclareFontShape{U}{#1}{#2}{#3}{<->sub*#4/#5/#6}{}}%
\@onlypreamble\subst@fontshape
```

(*End of definition for* `\subst@fontshape`.)

`\extra@def` This was replaced by `\DeclareFontFamily`.

```
\gdef\extra@def#1#2#3{%
     \warn@rel@i\extra@def\DeclareFontFamily
     \DeclareFontFamily{U}{#1}{}%
}%
\@onlypreamble\extra@def
```

(*End of definition for* `\extra@def`.)

`\default@mextra` The new name is `\DeclareFontEncodingDefaults` but in this case we don't feel comfortable with this either.

```
\gdef\default@mextra{%
  \warn@rel@i\default@mextra\DeclareFontEncodingDefaults
```

We pick up the argument to `\default@mextra` implicitly as the second argument of `\DeclareFontEncodingDefaults`.

```
  \DeclareFontEncodingDefaults\relax
}%
\@onlypreamble\default@mextra
```

(*End of definition for* `\default@mextra`.)

`\preload@sizes` The new interface is `\DeclarePreloadSizes`.

```
\gdef\preload@sizes{%
     \warn@rel@i\preload@sizes\DeclarePreloadSizes
     \DeclarePreloadSizes U%
}%
\@onlypreamble\preload@sizes
```

(*End of definition for* `\preload@sizes`.)

`\err@rel@i` This macro is used in cases where emulation with NFSS2 features is not really possible.

```
\gdef\err@rel@i#1#2{%
  \@latex@error{***  NFSS release 1 command \noexpand#1found%
          ^^J***  Recovery not possible. Use \string#2}%
       {The new release of NFSS doesn't support the
        \noexpand#1command^^Jany longer.
        Please upgrade your file to the syntax of NFSS
        release 2^^Jusing the \noexpand#2command.}%
```

Let's die.

```
  \batchmode\input.\relax
}%
\@onlypreamble\err@rel@i
```

(*End of definition for* `\err@rel@i`.)

`\newmathalphabet` `\newmathalphabet@@` `\newmathalphabet@@@`

`\newmathalphabet` is the old form.

```
\gdef\newmathalphabet{%
  \if@no@font@opt
    \@latex@error{*** NFSS release 1 command
                    \noexpand\newmathalphabet found%
     ^^J \space*** Automatic recovery not possible.%
     ^^J \space*** TYPE H for Help%
              }%
       {Please look at the file usrguide.tex for hints on
        how to resolve this problem.}%
  \else
     \warn@rel@i\newmathalphabet\DeclareMathAlphabet
  \fi
  \@ifstar\newmathalphabet@@@
          \newmathalphabet@@}%
\gdef\newmathalphabet@@#1{\DeclareMathAlphabet#1{U}{}{}{}}%
\gdef\newmathalphabet@@@#1#2#3#4{%
       \DeclareMathAlphabet{#1}{U}{#2}{#3}{#4}}%
\@onlypreamble\newmathalphabet
\@onlypreamble\newmathalphabet@@
\@onlypreamble\newmathalphabet@@@
```

(*End of definition for* `\newmathalphabet`, `\newmathalphabet@@`, *and* `\newmathalphabet@@@`.)

`\if@no@font@opt` `\@no@font@optfalse`

```
\global\let\if@no@font@opt\iftrue
\gdef\@no@font@optfalse{\let\if@no@font@opt\iffalse}%
```

(*End of definition for* `\if@no@font@opt` *and* `\@no@font@optfalse`.)

`\define@mathalphabet`

This is a case where dying is best.

```
\gdef\define@mathalphabet{%
      \err@rel@i\define@mathalphabet\DeclareMathAlphabet
}%
\@onlypreamble\define@mathalphabet
```

(*End of definition for* `\define@mathalphabet`.)

`\define@mathgroup`

And here is another one

```
\gdef\define@mathgroup{%
      \err@rel@i\define@mathgroup\DeclareSymbolFont
}%
\@onlypreamble\define@mathgroup
```

(*End of definition for* `\define@mathgroup`.)

\addtoversion \addtoversion is the old form.

```
\def\addtoversion#1#2{%
  \warn@rel@i\addtoversion\SetMathAlphabet
  \SetMathAlphabet#2{#1}{U}}%
\@onlypreamble\addtoversion
```

(*End of definition for* \addtoversion.)

Finishing off this huge \IncludeInRelease argument:

```
\EndIncludeInRelease
⟨/latexrelease⟩
```

File 28
ltfssdcl.dtx

This file contains the main implementation of the font selection scheme commands. See other parts of the LaTeX distribution, or *The LaTeX Companion* for higher level documentation of these commands.

> **Warning:** The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

1 Interface Commands

`\in@` `\ifin@`

`\@in` is a utility macro with two arguments. It determines whether its first argument occurs in its second and sets the switch `\ifin@` accordingly. The first argument may not contain braces nor `#` (more precisely, tokens of category code 1, 2, or 6).

```
⟨*2ekernel⟩
\def\in@#1#2%
 {%
   \begingroup
     \def\in@@##1#1{}%
     \toks@\expandafter{\in@@#2{}{}#1}%
     \edef\in@@{\the\toks@}%
   \expandafter\endgroup
   \ifx\in@@\@empty
     \in@false
   \else
     \in@true
   \fi
 }
\newif\ifin@
```

(*End of definition for* `\in@` *and* `\ifin@`.)

Before the `\begin{document}` command several ⟨*math versions*⟩ and ⟨*math alphabet identifiers*⟩ may be declared. In principle, there should be exactly one family/series/shape combination be declared for each version/alphabet pair. But we want to allow for defaults as well for automagical filling of holes.

While building the tables for math alphabet identifiers and math versions we keep several lists:

- the list of all math versions, `\version@list`, each entry prefixed by the control sequence `\version@elt`, i.e. this list has the following form

  `\version@elt`⟨*version*$_1$⟩`\version@elt`⟨*version*$_2$⟩...
  `\version@elt`⟨*version*$_n$⟩

- the list of all math alphabet identifiers. Here every entry has the form:

  `\group@elt`⟨*math group number*⟩
  `{{`⟨*default family*⟩`}{`⟨*default series*⟩`}{`⟨*default shape*⟩`}}`.

- Each defined math alphabet identifier holds a list containing Information about the *versions* for which it is defined. This list has a more complicated structure: it looks as follows:

  ```
  \set@alpha⟨the alphabet identifier itself⟩
     \reserved@c⟨math version⟩⟨font info⟩
    ...
  \@nil
  ```

  where ⟨*font info*⟩ is either `\reserved@e` (if the combination is not defined yet) or

  ```
  {{⟨family⟩}{⟨series⟩}{⟨shape⟩}}
  ```

`\version@list` We initialize the version list to be empty.

```
\let\version@list=\@empty
\@onlypreamble\version@list
```

(*End of definition for* `\version@list`.)

`\version@elt`

```
\let\version@elt\relax
\@onlypreamble\version@elt
```

(*End of definition for* `\version@elt`.)

`\new@mathversion` The macro `\new@mathversion` is called with the version control sequence as its argument.

```
%\def\new@mathversion#1{%
```

The first thing this macro does is to check if the version identifier is already present in `\version@list`. We enclose `\version@list` in braces since it might be empty (if no *version* is defined yet). But this means that we need a suitable number of `\expandafter` primitives.

```
%  \expandafter\in@\expandafter#1\expandafter{\version@list}%
%  \ifin@
```

If so it prints an error message. The `\next` macro is used to get rid of the four characters `\mv@` that would otherwise appear at the begin of the version name in the error message.

```
%    \@latex@error{Math version
%                  `\expandafter\@gobblefour\string#1'
%                  already defined}\@eha
```

Otherwise we have a new version, and we can proceed with entering it into the tables. We add it to `\version@list`. This is very easy: we define `\version@elt` (which is the delimiter in `\version@list`) to protect itself and the following token from being expanded and simply redefine `\version@list`.

```
%  \else
%     \global\expandafter\newcount\csname c@\expandafter
%                                 \@gobble\string#1\endcsname
%     \global\csname c@\expandafter
%                                 \@gobble\string#1\endcsname\@ne
%     \def\version@elt{\noexpand\version@elt\noexpand}%
%     \edef\version@list{\version@list\version@elt#1}%
```

Then we prepare to enter the new version into all math alphabet identifier lists. Remember that these lists use `\reserved@c` as delimiter, and that there appears the control sequence `\reserved@e` that must not be expanded. Therefore we take suitable precautions.

```
%     \def\reserved@c{\noexpand\reserved@c\noexpand}%
%     \let\reserved@e\relax
```

We will now go through the `\alpha@list` to process every ⟨*math alphabet identifier*⟩ in turn. Since this list has `\group@elt` as a delimiter we define this control sequence. It has three arguments as every entry consists of three items (as explained above).

```
%     \def\group@elt##1##2##3{%
```

The first of these arguments is the ⟨*math alphabet identifier*⟩. We redefine it by appending the information about the new version at the end of the list contained in it. However, there is one subtlety: the definitions for `\reserved@c` and `\reserved@e` made above prevent the main part of the list from being expanded. But we still have to take care of the header and the trailer. To do this we remove the trailer by means of the macro `\remove@nil` which also protect the header from being expanded. Its definition is given below. Now we can prepare to add the new version.

```
%          \edef##1{\expandafter\remove@nil##1%
%                   \reserved@c
%                   #1%
%                   \reserved@e
%                   \noexpand\@nil}}%
```

Finally we call `\alpha@list` which will now execute the macro `\group@elt` once for every defined ⟨*math alphabet identifier*⟩. And that's all for now.

```
%     \alpha@list
%  \fi}
```

(*End of definition for* `\new@mathversion`.)

`\alpha@list` As we explained above every entry in `\alpha@list` has the form

`\alpha@elt`
⟨*alphabet identifier*⟩⟨*internal group number*⟩⟨*default font assignments*⟩...

We initialize it to `\@empty`.

```
\let\alpha@list\@empty
\@onlypreamble\alpha@list
```

(*End of definition for* `\alpha@list`.)

`\alpha@elt`

```
\let\alpha@elt\relax
\@onlypreamble\alpha@elt
```

(*End of definition for* `\alpha@elt`.)

`\newgroup` Start the group (fam) allocation at 0. (Doesn't belong here.)

```
\count18=-1
```

(*End of definition for* `\newgroup`.)

`\stepcounter`

(*End of definition for* `\stepcounter`.)

`\select@group` We surround `\select@group` with braces so that functions using it can be used directly after `_` or `^`. However, if we use oldstyle syntax where the math alphabet doesn't have arguments (ie if `\math@bgroup` is not `\bgroup`) we need to get rid of the extra group.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}
⟨latexrelease⟩                  {\select@group}{\select@group}%
⟨*2ekernel | latexrelease⟩
\def\select@group#1#2#3#4{%
 \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
 {%
 \ifmmode
  \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
     \begingroup
       \escapechar\m@ne
       \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
       \globaldefs\@ne  \math@fonts
     \endgroup
     \init@restore@version
     \xdef#1{\noexpand\use@mathgroup\noexpand#2%
             {\number\csname c@mv@\math@version\endcsname}}%
     \global\advance\csname c@mv@\math@version\endcsname\@ne
   \else
     \let#1\relax
     \@latex@error{Too many math alphabets used in
                   version \math@version}%
        \@eha
   \fi
 \else \expandafter\non@alpherr\fi
 #1{#4}%
 }%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}
⟨latexrelease⟩                  {\select@group}{\select@group}%
⟨latexrelease⟩\def\select@group#1#2#3#4{%
⟨latexrelease⟩ \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
⟨latexrelease⟩ {%
⟨latexrelease⟩ \ifmmode
⟨latexrelease⟩  \ifnum\csname c@mv@\math@version\endcsname<\sixt@@n
⟨latexrelease⟩     \begingroup
⟨latexrelease⟩       \escapechar\m@ne
⟨latexrelease⟩       \getanddefine@fonts
⟨latexrelease⟩         {\csname c@mv@\math@version\endcsname}#3%
⟨latexrelease⟩       \globaldefs\@ne  \math@fonts
⟨latexrelease⟩     \endgroup
⟨latexrelease⟩     \init@restore@version
⟨latexrelease⟩     \xdef#1{\noexpand\use@mathgroup\noexpand#2%
⟨latexrelease⟩             {\number\csname c@mv@\math@version\endcsname}}%
⟨latexrelease⟩     \global\advance\csname c@mv@\math@version\endcsname\@ne
⟨latexrelease⟩   \else
⟨latexrelease⟩     \let#1\relax
⟨latexrelease⟩     \@latex@error{Too many math alphabets used in
⟨latexrelease⟩                   version \math@version}%
```

```
⟨latexrelease⟩         \@eha
⟨latexrelease⟩    \fi
⟨latexrelease⟩ \else \expandafter\non@alpherr\fi
⟨latexrelease⟩ #1{#4}%
⟨latexrelease⟩ }%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\restore@mathversion
```

(*End of definition for* `\select@group`.)

`\init@restore@version`

```
\def\init@restore@version{%
        \global\let\init@restore@version\relax
        \xdef\restore@mathversion
             {\expandafter\noexpand\csname mv@\math@version\endcsname
              \global\csname c@mv@\math@version\endcsname
              \number\csname c@mv@\math@version\endcsname\relax}%
        \aftergroup\dorestore@version
}
\@onlypreamble\init@restore@version
```

(*End of definition for* `\init@restore@version`.)

`\non@alpherr`

```
\gdef\non@alpherr#1{\@latex@error{%
```

The command here will have a space at the end of its name, so we make sure not to insert an extra one.

```
    \string#1allowed only in math mode}\@ehd}
```

(*End of definition for* `\non@alpherr`.)

`\dorestore@version`

```
\def\dorestore@version
 {\ifmmode
    \aftergroup\dorestore@version
  \else
    \gdef\init@restore@version{%
        \global\let\init@restore@version\relax
        \xdef\restore@mathversion
             {\expandafter\noexpand\csname mv@\math@version\endcsname
              \global\csname c@mv@\math@version\endcsname
              \number\csname c@mv@\math@version\endcsname\relax}%
        \aftergroup\dorestore@version
    }%
    \begingroup
      \let\getanddefine@fonts\@gobbletwo
      \restore@mathversion
    \endgroup
  \fi}%
\@onlypreamble\dorestore@version
```

(*End of definition for* `\dorestore@version`.)

\c@localmathalphabets

To avoid hitting the "no more math fams available" limit of 16, we keep a defined number of math alphabets flexible/local. If we have to allocate any of those we roll back the allocation after the formula has ended, so the next formula can use other alphabets in the slot(s). This makes the processing a bit slower if you are working at the limit, but that is better than dying with "out of memory".

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}
⟨latexrelease⟩   {\document@select@group}{\document@select@group}%
⟨*2ekernel | latexrelease⟩
```

We don't really undo the declaration on rollback (as that would be hard to maintain), so rolling forward needs to check if the declaration was already made.

```
\ifx\c@localmathalphabets\@undefined
```

There is no need to have this counter as part of the include checkpoints, given that it makes little sense to alter its settings mid document. All we want is the ability to change it using the \setcounter interface.

By default we keep two math fams flexible.

```
  \newcount\c@localmathalphabets
  \setcounter{localmathalphabets}{2}
\fi
```

(*End of definition for* \c@localmathalphabets.)

\document@select@group

The \document@select@group command is the version of \select@group (inside math versions) that is used in the document body to set up math alphabets (if used).

```
\def\document@select@group#1#2#3#4{%
 \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
 {%
 \ifmmode
```

We first check if there is still room for allocating another mathgroup. If there is, we check if it can be globally allocated or if we have reached the limit which is given by \e@mathgroup@top with \c@localmathalphabets subtracted.

```
   \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
     \ifnum \numexpr\e@mathgroup@top-\c@localmathalphabets
              >\csname c@mv@\math@version\endcsname
     \else
```

If we are past this point we freeze the current state of the math version so that we can return to it after the formula has ended. Of course, that should be done only once, so we check if \mv@⟨*version*⟩@frozen already exists.

```
       \ifcsname mv@\math@version @frozen\endcsname \else
```

We have to pass the current value of \math@version not the macro itself, because some of the processing is delayed to a point where the value may have changed again—not doing this caused a puzzling error in one setup.

```
         \expandafter\freeze@math@version\expandafter{\math@version}%
       \fi
     \fi
     \begingroup
       \escapechar\m@ne
       \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
       \globaldefs\@ne  \math@fonts
```

```
    \endgroup
    \expandafter\extract@alph@from@version
        \csname mv@\math@version\expandafter\endcsname
        \expandafter{\number\csname
                      c@mv@\math@version\endcsname}%
         #1%
    \global\advance\csname c@mv@\math@version\endcsname\@ne
  \else
    \let#1\relax
    \@latex@error{Too many math alphabets used in
                  version \math@version}%
       \@eha
 \fi
```

Extra `\expandafter` to remove the `\expandafter` added below

```
\else \expandafter\expandafter\expandafter\non@alpherr\fi
```

We surround `\select@group` with braces so that functions using it can be used directly after `_` or `^`.

If the legacy interface is used, e.g., `$\sf -1$` the math alphabet `#1` does not take an argument so we better do not surround `#4` with braces, because then we get `{\relax}` into the formula and introduce an extra Ord atom. The two different cases can be distinguished by looking at the current value of `\math@bgroup`.

```
 \expandafter#1\ifx\math@bgroup\bgroup{#4}\else#4\fi
 }%
}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}
⟨latexrelease⟩  {\document@select@group}{\document@select@group}%
⟨latexrelease⟩
⟨latexrelease⟩\def\document@select@group#1#2#3#4{%
⟨latexrelease⟩ \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
⟨latexrelease⟩ {%
⟨latexrelease⟩ \ifmmode
⟨latexrelease⟩   \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
⟨latexrelease⟩     \begingroup
⟨latexrelease⟩       \escapechar\m@ne
⟨latexrelease⟩       \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
⟨latexrelease⟩       \globaldefs\@ne  \math@fonts
⟨latexrelease⟩     \endgroup
⟨latexrelease⟩     \expandafter\extract@alph@from@version
⟨latexrelease⟩         \csname mv@\math@version\expandafter\endcsname
⟨latexrelease⟩         \expandafter{\number\csname
⟨latexrelease⟩                       c@mv@\math@version\endcsname}%
⟨latexrelease⟩          #1%
⟨latexrelease⟩     \global\advance\csname c@mv@\math@version\endcsname\@ne
⟨latexrelease⟩   \else
⟨latexrelease⟩     \let#1\relax
⟨latexrelease⟩     \@latex@error{Too many math alphabets used
⟨latexrelease⟩                   in version \math@version}%
⟨latexrelease⟩        \@eha
⟨latexrelease⟩  \fi
⟨latexrelease⟩ \else \expandafter\expandafter\expandafter\non@alpherr\fi
```

```
⟨latexrelease⟩ \expandafter#1\ifx\math@bgroup\bgroup{#4}\else#4\fi
⟨latexrelease⟩ }%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/01/01}
⟨latexrelease⟩  {\document@select@group}{\document@select@group}%
⟨latexrelease⟩
⟨latexrelease⟩\def\document@select@group#1#2#3#4{%
⟨latexrelease⟩ \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
⟨latexrelease⟩ {%
⟨latexrelease⟩ \ifmmode
⟨latexrelease⟩   \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
⟨latexrelease⟩     \begingroup
⟨latexrelease⟩       \escapechar\m@ne
⟨latexrelease⟩       \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
⟨latexrelease⟩       \globaldefs\@ne  \math@fonts
⟨latexrelease⟩     \endgroup
⟨latexrelease⟩     \expandafter\extract@alph@from@version
⟨latexrelease⟩         \csname mv@\math@version\expandafter\endcsname
⟨latexrelease⟩         \expandafter{\number\csname
⟨latexrelease⟩                       c@mv@\math@version\endcsname}%
⟨latexrelease⟩          #1%
⟨latexrelease⟩     \global\advance\csname c@mv@\math@version\endcsname\@ne
⟨latexrelease⟩   \else
⟨latexrelease⟩     \let#1\relax
⟨latexrelease⟩     \@latex@error{Too many math alphabets used
⟨latexrelease⟩                   in version \math@version}%
⟨latexrelease⟩        \@eha
⟨latexrelease⟩  \fi
⟨latexrelease⟩ \else \expandafter\non@alpherr\fi
⟨latexrelease⟩ #1{#4}%
⟨latexrelease⟩ }%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}
⟨latexrelease⟩  {\document@select@group}{\document@select@group}%
⟨latexrelease⟩
⟨latexrelease⟩\def\document@select@group#1#2#3#4{%
⟨latexrelease⟩ \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
⟨latexrelease⟩ {%
⟨latexrelease⟩ \ifmmode
⟨latexrelease⟩   \ifnum\csname c@mv@\math@version\endcsname<\sixt@@n
⟨latexrelease⟩     \begingroup
⟨latexrelease⟩       \escapechar\m@ne
⟨latexrelease⟩       \getanddefine@fonts
⟨latexrelease⟩         {\csname c@mv@\math@version\endcsname}#3%
⟨latexrelease⟩       \globaldefs\@ne  \math@fonts
⟨latexrelease⟩     \endgroup
⟨latexrelease⟩     \expandafter\extract@alph@from@version
⟨latexrelease⟩         \csname mv@\math@version\expandafter\endcsname
⟨latexrelease⟩         \expandafter{\number\csname
⟨latexrelease⟩                       c@mv@\math@version\endcsname}%
⟨latexrelease⟩          #1%
```

```
⟨latexrelease⟩     \global\advance\csname c@mv@\math@version\endcsname\@ne
⟨latexrelease⟩   \else
⟨latexrelease⟩     \let#1\relax
⟨latexrelease⟩     \@latex@error{Too many math alphabets used
⟨latexrelease⟩                   in version \math@version}%
⟨latexrelease⟩        \@eha
⟨latexrelease⟩  \fi
⟨latexrelease⟩ \else \expandafter\non@alpherr\fi
⟨latexrelease⟩ #1{#4}%
⟨latexrelease⟩ }%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\document@select@group`.)

`\freeze@math@version` This command stores the current state of the math version and sets things up to return to it after each formula from now on. We use L3 programming layer code to set it up.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/11/01}%
⟨latexrelease⟩                 {\freeze@math@version}{freeze math version}%
\ExplSyntaxOn
\cs_new_protected:Npn\freeze@math@version #1 {
```

Save the current `\mv@`⟨*version*⟩ code and the number of allocated mathgroups inside.

```
  \@font@info{Freeze~ math~ alphabet~ allocation~ in~ version~
              #1.\MessageBreak
              Allocated~math~groups:~\int_use:c{ c@mv@ #1 }~
              (local:~  \int_use:N\c@localmathalphabets)      }
  \cs_gset_eq:cc { mv@#1@frozen }{ mv@#1 }
  \tl_gset:cx { g__nfss_frozen_mv_ #1 _tl }{ \int_use:c { c@mv@#1 } }
```

Here is the definition of `\mv@`⟨*version*⟩`@reset`. If there has been no new math alphabet allocation, doing a reset would just cause a lot of unnecessary processing, so we do a quick check upfront for this.

```
  \cs_gset:cpn{mv@#1@reset}
     {
         \int_compare:nNnTF { \int_use:c{c@mv@#1} } >
                            { \tl_use:c{g__nfss_frozen_mv_ #1 _tl} }
             {
               \@font@info{Undo~ math~ alphabet~ allocation~ in~ version~ #1}
```

If the undo is necessary, we restore the `\mv@`⟨*version*⟩ code.

```
               \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
               \int_gset:cn { c@mv@#1 }{ \tl_use:c {g__nfss_frozen_mv_ #1 _tl} }
```

But we also should undo changes to the math alphabet definitions. We therefore run this code with a modified definition for `\getanddefine@fonts` because there is no need to do anything to the symbol fonts that are permanently allocated.

```
               \group_begin:
                 \cs_set_eq:NN \getanddefine@fonts \use_none:nn
                 \use:c {mv@#1}
               \group_end:
             }
             {
```

If there was no change, we report that in the log (but this branch could go completely).

```
                \@font@info{No~ math~ alphabet~ change~ to~ frozen~ version~ #1}
              }
```

If this is executed after a math display, we may have to arrange for ignoring spaces, because they are now hidden if the tokens from above intervene. This is signaled by the 2e switch `@ignore` which is set in `\frozen@everymath` and `\frozen@everydisplay`.

This is all 2e code so we use that syntax.

```
          \if@ignore \ignorespaces \fi
    }
}
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2021/11/15}
⟨latexrelease⟩                 {\freeze@math@version}{freeze math version}%
⟨latexrelease⟩
⟨latexrelease⟩\ExplSyntaxOn
⟨latexrelease⟩\cs_set_protected:Npn\freeze@math@version #1 {
⟨latexrelease⟩  \@font@info{Freeze~ math~ alphabet~ allocation~ in~ version~
⟨latexrelease⟩              #1.\MessageBreak
⟨latexrelease⟩              Allocated~math~groups:~\int_use:c{ c@mv@ #1 }~
⟨latexrelease⟩              (local:~  \int_use:N\c@localmathalphabets)      }
⟨latexrelease⟩  \cs_gset_eq:cc { mv@#1@frozen }{ mv@#1 }
⟨latexrelease⟩  \tl_gset:cx { g__nfss_frozen_mv_ #1 _tl }{ \int_use:c { c@mv@#1 } }
⟨latexrelease⟩  \group_insert_after:N \__nfss_init_mv_freeze:N
⟨latexrelease⟩  \exp_after:wN \group_insert_after:N \cs:w mv@#1@reset \cs_end:
⟨latexrelease⟩  \tl_gput_right:No \check@mathfonts
⟨latexrelease⟩       {
⟨latexrelease⟩         \exp_after:wN \group_insert_after:N \cs:w mv@#1@reset \cs_end:
⟨latexrelease⟩       }
⟨latexrelease⟩  \cs_gset:cpn{mv@#1@reset}
⟨latexrelease⟩     {
⟨latexrelease⟩         \int_compare:nNnTF { \int_use:c{c@mv@#1} } >
⟨latexrelease⟩                            { \tl_use:c{g__nfss_frozen_mv_ #1 _tl} }
⟨latexrelease⟩             {
⟨latexrelease⟩               \@font@info{Undo~ math~ alphabet~ allocation~ in~ version~ #1}
⟨latexrelease⟩               \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
⟨latexrelease⟩               \int_gset:cn { c@mv@#1 }{ \tl_use:c {g__nfss_frozen_mv_ #1 _tl} }
⟨latexrelease⟩               \group_begin:
⟨latexrelease⟩                 \cs_set_eq:NN \getanddefine@fonts \use_none:nn
⟨latexrelease⟩                 \use:c {mv@#1}
⟨latexrelease⟩               \group_end:
⟨latexrelease⟩             }
⟨latexrelease⟩             {
⟨latexrelease⟩               \@font@info{No~ math~ alphabet~ change~ to~ frozen~ version~ #1}
⟨latexrelease⟩             }
⟨latexrelease⟩         \if@ignore \ignorespaces \fi
⟨latexrelease⟩    }
⟨latexrelease⟩}
⟨latexrelease⟩\cs_set_protected:Npn \__nfss_init_mv_freeze:N #1 {%
⟨latexrelease⟩  \mode_if_math:T { \group_insert_after:N \__nfss_init_mv_freeze:N
⟨latexrelease⟩                    \group_insert_after:N } #1
⟨latexrelease⟩}
```

```
⟨latexrelease⟩\ExplSyntaxOff
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\freeze@math@version`.)

`\process@table`

```
\def\process@table{%
    \def\cdp@elt##1##2##3##4{%
        \@font@info{Checking defaults for
                  ##1/##2/##3/##4}%
        \expandafter
        \ifx\csname##1/##2/##3/##4\endcsname\relax
```

Grouping is important for two reasons, first `\cdp@elt` will get redefined if `\Declare...` functions are executed within the external `.fd` file and secondly `\try@load@fontshape` changes a lot of catcodes without surrounding itself with a group.

```
          \begingroup
           \def\f@encoding{##1}\def\f@family{##2}%
           \try@load@fontshape
          \endgroup
        \fi
        \expandafter
        \ifx\csname##1/##2/##3/##4\endcsname\relax
             \@latex@error{This NFSS system isn't set up properly}%
                       {For encoding scheme ##1 the defaults
                        ##2/##3/##4 do not form a valid font shape}%
        \else
             \@font@info{... okay}%
        \fi}%
    \cdp@list
```

Now we make sure that `\error@fontshape` is okay.

```
    \begingroup
       \escapechar\m@ne
       \error@fontshape
       \expandafter\ifx\csname \curr@fontshape\endcsname\relax
          \begingroup
            \try@load@fontshape
           \endgroup
       \fi
       \expandafter\ifx\csname \curr@fontshape\endcsname\relax
         \@latex@error{This NFSS system isn't set up properly}%
            {The system maintainer forgot to specify a suitable
             substitution
             font shape using the \noexpand\DeclareErrorFont
             command}%
       \fi
    \endgroup
```

Set `\select@group` to its meaning used within the document body.

```
    \let\select@group\document@select@group
```

Install the default font attributes as they are currently pointing to error font face. We can speed up the process by just using `\edef`, thereby avoiding all kind of extra processing. Don't use `\reset@font` since that would trigger `\selectfont`.

```
    \fontencoding\encodingdefault
    \edef\f@family{\familydefault}%
    \edef\f@series{\seriesdefault}%
    \edef\f@shape{\shapedefault}%
```

Drop stuff not longer needed. We need to add many more!!!!!!

```
 \everyjob{}%
}
\@onlypreamble\process@table
```

(*End of definition for* `\process@table`.)

```
%\@onlypreamble\set@mathradical
```

`\DeclareMathVersion`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/11/01}%
⟨latexrelease⟩                  {\DeclareMathVersion}{local alphabets}%
\def\DeclareMathVersion#1{%
```

When declaring a new math version we need to instantiate an L3 variable that is used when we freeze the version, because too many alphabets got allocated. If we don't do this, L3 programming layer complains if it is run in checking mode.

```
  \@namedef{g__nfss_frozen_mv_#1_tl}{}%
```

We also extend `\check@mathfonts` to call a version reset (once frozen) after a formula has finished.

```
  \expandafter\ifx\csname mv@#1\endcsname \relax
    \expandafter \g@addto@macro \expandafter \check@mathfonts
         \expandafter {\expandafter \aftergroup \csname mv@#1@reset\endcsname}%
```

Initially this macro does nothing. It is, however, important that it doesn't stop any `\ignorespaces`, so we make it expandable and not `\relax`.

```
    \@namedef{mv@#1@reset}{}%
  \fi
```

```
  \expandafter\new@mathversion\csname mv@#1\endcsname}
\@onlypreamble\DeclareMathVersion
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                  {\DeclareMathVersion}{local alphabets}%
⟨latexrelease⟩\def\DeclareMathVersion#1{%
⟨latexrelease⟩  \@namedef{g__nfss_frozen_mv_#1_tl}{}%
⟨latexrelease⟩  \expandafter\new@mathversion\csname mv@#1\endcsname}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\DeclareMathVersion}{local alphabets}%
⟨latexrelease⟩\def\DeclareMathVersion#1{%
⟨latexrelease⟩  \expandafter\new@mathversion\csname mv@#1\endcsname}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\DeclareMathVersion`.)

`\new@mathversion`

```
\def\new@mathversion#1{%
  \expandafter\in@\expandafter#1\expandafter{\version@list}%
  \ifin@
    \@font@info{Redeclaring math version
                `\expandafter\@gobblefour\string#1'}%
  \else
    \expandafter\newcount\csname c@\expandafter
                                \@gobble\string#1\endcsname
    \def\version@elt{\noexpand\version@elt\noexpand}%
    \edef\version@list{\version@list\version@elt#1}%
  \fi
```

`\toks@` is used to gather all tokens for the math version. `\count@` will be used to count the math groups we add to this version.

```
  \toks@{}%
  \count@\z@
```

Now we loop over `\group@list` to add all math groups defined so far to the version and at the same time to count them.

```
  \def\group@elt##1##2{%
       \advance\count@\@ne
       \addto@hook\toks@{\getanddefine@fonts##1##2}%
       }%
  \group@list
```

We set the counter for this math version to the number of math groups found in `\group@list`.

```
  \global\csname c@\expandafter\@gobble\string#1\endcsname\count@
```

Now we loop over `\alpha@list` to add all math alphabets known so far. We have to distinguish the case that an alphabet by default should produce an error in new versions.

```
  \def\alpha@elt##1##2##3{%
       \ifx##2\no@alphabet@error
         \toks@\expandafter{\the\toks@\install@mathalphabet##1%
             {\no@alphabet@error##1}}%
       \else
         \toks@\expandafter{\the\toks@\install@mathalphabet##1%
             {\select@group##1##2##3}}%
       \fi
          }%
  \alpha@list
```

Finally we define the math version to expand to the contents of `\toks@`.

```
  \xdef#1{\the\toks@}%
}
\@onlypreamble\new@mathversion
```

(*End of definition for* `\new@mathversion`.)

`\DeclareSymbolFont` First drop any surplus `m` from the series argument then do what has been done since 1994.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\IncludeInRelease{2022/11/01}%
⟨latexrelease⟩                  {\DeclareSymbolFont}{maybe drop m}%
\def\DeclareSymbolFont #1#2#3#4#5{%
 \def\reserved@a{\DeclareSymbolFont@m@dropped{#1}{#2}{#3}}%
 \edef\reserved@b{#4}%
 \series@maybe@drop@one@m\reserved@b\reserved@b
 \expandafter\reserved@a\expandafter{\reserved@b}{#5}%
}
\def\DeclareSymbolFont@m@dropped #1#2#3#4#5{%
 \@tempswafalse
 \edef\reserved@b{#2}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
   \@ifundefined{sym#1}{%

      \ifnum\count18<15 %
        \expandafter\new@mathgroup\csname sym#1\endcsname
        \expandafter\new@symbolfont\csname sym#1\endcsname
                        {#2}{#3}{#4}{#5}%
      \else
         \@latex@error{Too many symbol fonts declared}\@eha
      \fi
     }%
     {%
      \@font@info{Redeclaring symbol font ‘#1’}%
```

Update the group list.

```
      \def\group@elt##1##2{%
           \noexpand\group@elt\noexpand##1%
           \expandafter\ifx\csname sym#1\endcsname##1%
             \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
           \else
               \noexpand##2%
           \fi}%
      \xdef\group@list{\group@list}%
```

Update the version list.

```
      \def\version@elt##1{%
          \expandafter
          \SetSymbolFont@\expandafter##1\csname#2/#3/#4/#5\expandafter
              \endcsname \csname sym#1\endcsname
          }%
      \version@list
     }%
  \else
    \@latex@error{Encoding scheme  ‘#2’ unknown}\@eha
  \fi
  }
\@onlypreamble\DeclareSymbolFont
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
```

```
⟨latexrelease⟩                  {\DeclareSymbolFont}{maybe drop m}%
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareSymbolFont\DeclareSymbolFont@m@dropped
⟨latexrelease⟩\let\DeclareSymbolFont@m@dropped\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\DeclareSymbolFont`.)

`\group@list`

```
\let\group@list\@empty
\@onlypreamble\group@list
```

(*End of definition for* `\group@list`.)

`\group@elt`

```
\let\group@elt\relax
\@onlypreamble\group@elt
```

(*End of definition for* `\group@elt`.)

`\new@symbolfont`

```
\def\new@symbolfont#1#2#3#4#5{%
    \toks@\expandafter{\group@list}%
    \edef\group@list{\the\toks@\noexpand\group@elt\noexpand#1%
                       \expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
    \def\version@elt##1{\toks@\expandafter{##1}%
                     \edef##1{\the\toks@\noexpand\getanddefine@fonts
                     #1\expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
                    \global\advance\csname c@\expandafter
                                  \@gobble\string##1\endcsname\@ne
                  }%
    \version@list
}
\@onlypreamble\new@symbolfont
```

(*End of definition for* `\new@symbolfont`.)

`\SetSymbolFont` First drop any surplus `m` from the series argument then do what has been done since 1994.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/11/01}%
⟨latexrelease⟩                  {\SetSymbolFont}{maybe drop m}%
\def\SetSymbolFont #1#2#3#4#5#6{%
 \def\reserved@a{\SetSymbolFont@m@dropped{#1}{#2}{#3}{#4}}%
 \edef\reserved@b{#5}%
 \series@maybe@drop@one@m\reserved@b\reserved@b
 \expandafter\reserved@a\expandafter{\reserved@b}{#6}%
}
```

```
\def\SetSymbolFont@m@dropped#1#2#3#4#5#6{%
 \@tempswafalse
 \edef\reserved@b{#3}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
  \expandafter\SetSymbolFont@
    \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
    \endcsname \csname sym#1\endcsname
 \else
  \@latex@error{Encoding scheme  `#3' unknown}\@eha
 \fi
}
\@onlypreamble\SetSymbolFont
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\SetSymbolFont}{maybe drop m}%
⟨latexrelease⟩
⟨latexrelease⟩\let\SetSymbolFont\SetSymbolFont@m@dropped
⟨latexrelease⟩\let\SetSymbolFont@m@dropped\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\SetSymbolFont`.)

`\SetSymbolFont@`

```
\def\SetSymbolFont@#1#2#3{%
  \expandafter\in@\expandafter#1\expandafter{\version@list}%
  \ifin@
    \expandafter\in@\expandafter#3\expandafter{\group@list}%
    \ifin@
      \begingroup
        \expandafter\get@cdp\string#2\@nil\reserved@a
        \toks@{}%
        \def\install@mathalphabet##1##2{%
             \addto@hook\toks@{\install@mathalphabet##1{##2}}%
            }%
        \def\getanddefine@fonts##1##2{%
          \ifnum##1=#3%
             \addto@hook\toks@{\getanddefine@fonts#3#2}%
             \expandafter\get@cdp\string##2\@nil\reserved@b
             \ifx\reserved@a\reserved@b\else
                \@font@info{Encoding `\reserved@b' has changed
                    to `\reserved@a' for symbol font\MessageBreak
                   `\expandafter\@gobblefour\string#3' in the
                    math version `\expandafter
                    \@gobblefour\string#1'}%
             \fi
             \@font@info{%
                Overwriting symbol font
                `\expandafter\@gobblefour\string#3' in
```

```
                 version `\expandafter
                \@gobblefour\string#1'\MessageBreak
                \@spaces \expandafter\@gobble\string##2 -->
                        \expandafter\@gobble\string#2}%
          \else
             \addto@hook\toks@{\getanddefine@fonts##1##2}%
          \fi}%
         #1%
         \xdef#1{\the\toks@}%
      \endgroup
    \else
       \@latex@error{Symbol font `\expandafter\@gobblefour\string#3'
                  not defined}\@eha
    \fi
  \else
    \@latex@error{Math version `\expandafter\@gobblefour\string#1'
       is not
       defined}{You probably misspelled the name of the math
       version.^^JOr you have to specify an additional package.}%
  \fi
}
\@onlypreamble\SetSymbolFont@
```

(*End of definition for* `\SetSymbolFont@`.)

`\get@cdp`

```
\def\get@cdp#1#2/#3\@nil#4{\def#4{#2}}
\@onlypreamble\get@cdp
```

(*End of definition for* `\get@cdp`.)

`\DeclareMathAlphabet`

```
\def\DeclareMathAlphabet#1#2#3#4#5{%
 \@tempswafalse
 \edef\reserved@b{#2}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
   \expandafter\ifx
   \csname\expandafter\@gobble\string#1\endcsname
   \relax
      \new@mathalphabet#1{#2}{#3}{#4}{#5}%
   \else
```

Check if it is already a math alphabet.

```
      \edef\reserved@a{\noexpand\in@{\string\select@group}%
          {\expandafter\meaning\csname \expandafter
           \@gobble\string#1\space\endcsname}}%
      \reserved@a
      \ifin@
        \@font@info{Redeclaring math alphabet \string#1}%
        \def\version@elt##1{%
          \expandafter\SetMathAlphabet@\expandafter
             ##1\csname#2/#3/#4/#5\expandafter\endcsname
```

```
            \csname M@#2\expandafter\endcsname
            \csname \expandafter\@gobble\string#1\space\endcsname#1}%
        \version@list
      \else
```

Check if it is a math alphabet defined via `\DeclareSymbolFontAlphabet`.

```
        \edef\reserved@a{\noexpand\in@{\string\use@mathgroup}%
          {\expandafter\meaning\csname \expandafter
           \@gobble\string#1\space\endcsname}}%
        \reserved@a
        \ifin@
```

In that case overwriting is simple since there is nothing inserted in the math version macros.

```
          \@font@info{Redeclaring math alphabet \string#1}%
          \new@mathalphabet#1{#2}{#3}{#4}{#5}%
```

Otherwise panic.

```
        \else
          \@latex@error{Command `\string#1' already defined}\@eha
        \fi
      \fi
  \fi
 \else
  \@latex@error{Encoding scheme  `#2' unknown}\@eha
 \fi
  }
\@onlypreamble\DeclareMathAlphabet
```

(*End of definition for* `\DeclareMathAlphabet`.)

`\new@mathalphabet`

```
\def\new@mathalphabet#1#2#3#4#5{%
    \toks@\expandafter{\alpha@list}%
    \edef#1{\expandafter\noexpand\csname \expandafter
            \@gobble\string#1\space\endcsname
            \if/#5/%
               \noexpand\no@alphabet@error
               \noexpand\no@alphabet@error
            \else
               \expandafter\noexpand\csname M@#2\endcsname
               \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
            \fi
           }%
    \toks2\expandafter{#1}%
    \edef\alpha@list{\the\toks@\noexpand\alpha@elt\the\toks2}%
    \def\version@elt##1{\toks@\expandafter{##1}%
                   \edef##1{\the\toks@\install@mathalphabet
                            \expandafter\noexpand
                            \csname \expandafter\@gobble
                               \string#1\space\endcsname
                           {\if/#5/%
                             \noexpand\no@alphabet@error
                             \noexpand#1%
                            \else
```

```
                              \noexpand\select@group\the\toks2
                             \fi}}%
                  }%
    \version@list
    \expandafter\edef\csname \expandafter\@gobble
                 \string#1\space\endcsname{\if/#5/%
               \noexpand\no@alphabet@error
               \noexpand#1%
             \else
               \noexpand\select@group\the\toks2
             \fi}%
    \edef#1{\noexpand\protect
             \expandafter\noexpand\csname \expandafter
             \@gobble\string#1\space\endcsname}%
}
\@onlypreamble\new@mathalphabet
```

(*End of definition for* `\new@mathalphabet`.)

`\SetMathAlphabet`

```
\def\SetMathAlphabet#1#2#3#4#5#6{%
 \@tempswafalse
 \edef\reserved@b{#3}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
  \expandafter\SetMathAlphabet@
    \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
    \endcsname \csname M@#3\expandafter\endcsname
    \csname \expandafter\@gobble\string#1\space\endcsname#1%
 \else
  \@latex@error{Encoding scheme  `#3' unknown}\@eha
 \fi
}
\@onlypreamble\SetMathAlphabet
```

(*End of definition for* `\SetMathAlphabet`.)

`\SetMathAlphabet@`

```
\def\SetMathAlphabet@#1#2#3#4#5{%
  \expandafter\in@\expandafter#1\expandafter{\version@list}%
  \ifin@
    \expandafter\in@\expandafter#4\expandafter{\alpha@list}%
    \ifin@
      \begingroup
        \toks@{}%
        \def\getanddefine@fonts##1##2{%
             \addto@hook\toks@{\getanddefine@fonts##1##2}%
            }%
        \def\reserved@c##1##2##3##4{%                % for message below
            \expandafter\@gobble\string##4}%
        \def\install@mathalphabet##1##2{%
          \ifx##1#4%
             \addto@hook\toks@
```

```
                {\install@mathalphabet#4{\select@group#4#3#2}}%
            \@font@info{Overwriting math alphabet
                `\string#5' in version `\expandafter
                 \@gobblefour\string#1'\MessageBreak
                 \@spaces \reserved@c##2 -->
                        \expandafter\@gobble\string#2}%
          \else
            \addto@hook\toks@{\install@mathalphabet##1{##2}}%
          \fi
          }%
        #1%
        \xdef#1{\the\toks@}%
      \endgroup
    \else
```

If the math alphabet was defined via `\DeclareSymbolFontAlphabet` we have remove its external definition and add it as a normal math alphabet to every version before trying to change it in one version.

```
      \edef\reserved@a{%
        \noexpand\in@{\string\use@mathgroup}{\meaning#4}}%
      \reserved@a
      \ifin@
        \def\reserved@b##1\use@mathgroup##2##3{%
            \def\reserved@b{##3}\def\reserved@c{##2}}%
        \expandafter\reserved@b#4%
        \begingroup
          \def\install@mathalphabet##1##2{%
              \addto@hook\toks@{\install@mathalphabet##1{##2}}%
              }%
           \def\getanddefine@fonts##1##2{%
             \addto@hook\toks@{\getanddefine@fonts##1##2}%
             \ifnum##1=\reserved@b
                \expandafter
                \addto@hook\expandafter\toks@
                \expandafter{\expandafter\install@mathalphabet
                \expandafter#4\expandafter
                      {\expandafter\select@group\expandafter
                        #4\reserved@c##2}}%
             \fi
                     }%
          \def\version@elt##1{%
              \toks@{}%
              ##1%
              \xdef##1{\the\toks@}%
             }%
          \version@list
        \endgroup
```

Put it into the `\alpha@list` with default 'error'

```
        \expandafter\gdef\expandafter\alpha@list\expandafter
            {\alpha@list
             \alpha@elt #4\no@alphabet@error \no@alphabet@error}%
        \gdef#4{\no@alphabet@error #5}% fake things :-)
```

Then call the internal setting routine again:

```
        \SetMathAlphabet@{#1}{#2}{#3}#4#5%
      \else
        \@latex@error{Command `\string#5' not defined as a
                      math alphabet}%
           {Use \noexpand\DeclareMathAlphabet to define it.}%
      \fi
    \fi
  \else
    \@latex@error{Math version `\expandafter\@gobblefour\string#1'
      is not
      defined}{You probably misspelled the name of the math
      version.^^JOr you have to specify an additional package.}%
  \fi
}
\@onlypreamble\SetMathAlphabet@
```

(*End of definition for* `\SetMathAlphabet@`.)

`\DeclareMathAccent` Could do with more checks like allowing single number in `#4` lowercase in `#4` etc

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {DeclareMathAccent}{Make math accents robust}%
\def\DeclareMathAccent#1#2#3#4{%
  \expandafter\in@\csname sym#3\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
    \begingroup
      \count\z@=#4\relax
      \count\tw@\count\z@
      \divide\count\z@\sixt@@n
      \count@\count\z@
      \multiply\count@\sixt@@n
      \advance\count\tw@-\count@
      \if\relax\noexpand#1% is command?
        \edef\reserved@a{\noexpand\in@
          {\expandafter\@gobble\string\mathaccent}
          {\expandafter\meaning
           \csname\expandafter\@gobble\string#1\space\endcsname}}%
        \reserved@a
        \ifin@
          \expandafter\let
            \csname\expandafter\@gobble\string#1\space\endcsname
            \@undefined
          \expandafter\set@mathaccent
             \csname sym#3\endcsname#1#2%
             {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \@font@info{Redeclaring math accent \string#1}%
        \else
          \expandafter\ifx
          \csname\expandafter\@gobble\string#1\endcsname
          \relax
            \expandafter\set@mathaccent
               \csname sym#3\endcsname#1#2%
```

```
                {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \else
            \@latex@error{Command `\string#1' already defined}\@eha
          \fi
        \fi
      \else
       \@latex@error{Not a command name: `\noexpand#1'}\@eha
      \fi
    \endgroup
  \else
    \@latex@error{Symbol font `#3' is not defined}\@eha
  \fi
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {DeclareMathAccent}{Make math accents robust}%
⟨latexrelease⟩\def\DeclareMathAccent#1#2#3#4{%
⟨latexrelease⟩  \expandafter\in@\csname sym#3\expandafter\endcsname
⟨latexrelease⟩     \expandafter{\group@list}%
⟨latexrelease⟩  \ifin@
⟨latexrelease⟩    \begingroup
⟨latexrelease⟩      \count\z@=#4\relax
⟨latexrelease⟩      \count\tw@\count\z@
⟨latexrelease⟩      \divide\count\z@\sixt@@n
⟨latexrelease⟩      \count@\count\z@
⟨latexrelease⟩      \multiply\count@\sixt@@n
⟨latexrelease⟩      \advance\count\tw@-\count@
⟨latexrelease⟩      \if\relax\noexpand#1% is command?
⟨latexrelease⟩        \edef\reserved@a{\noexpand\in@
⟨latexrelease⟩           {\expandafter\@gobble\string\mathaccent}{\meaning#1}}%
⟨latexrelease⟩        \reserved@a
⟨latexrelease⟩        \ifin@
⟨latexrelease⟩          \expandafter\set@mathaccent
⟨latexrelease⟩             \csname sym#3\endcsname#1#2%
⟨latexrelease⟩             {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
⟨latexrelease⟩          \@font@info{Redeclaring math accent \string#1}%
⟨latexrelease⟩        \else
⟨latexrelease⟩          \expandafter\ifx
⟨latexrelease⟩          \csname\expandafter\@gobble\string#1\endcsname
⟨latexrelease⟩          \relax
⟨latexrelease⟩            \expandafter\set@mathaccent
⟨latexrelease⟩               \csname sym#3\endcsname#1#2%
⟨latexrelease⟩               {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
⟨latexrelease⟩          \else
⟨latexrelease⟩            \@latex@error{Command `\string#1' already defined}\@eha
⟨latexrelease⟩          \fi
⟨latexrelease⟩        \fi
⟨latexrelease⟩      \else
⟨latexrelease⟩       \@latex@error{Not a command name: `\noexpand#1'}\@eha
⟨latexrelease⟩      \fi
⟨latexrelease⟩    \endgroup
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@latex@error{Symbol font `#3' is not defined}\@eha
```

```
⟨latexrelease⟩   \fi
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
\@onlypreamble\DeclareMathAccent
```

(*End of definition for* `\DeclareMathAccent`.)

`\set@mathaccent`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\set@mathaccent}{makemath accents robust}%
\def\set@mathaccent#1#2#3#4{%
  \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}%
  \MakeRobust#2%
}
\@onlypreamble\set@mathaccent
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\set@mathaccent}{makemath accents robust}%
⟨latexrelease⟩
⟨latexrelease⟩\def\set@mathaccent#1#2#3#4{%
⟨latexrelease⟩   \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\set@mathaccent`.)

`\DeclareMathSymbol`

```
\def\DeclareMathSymbol#1#2#3#4{%
  \expandafter\in@\csname sym#3\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
    \begingroup
      \count\z@=#4\relax
      \count\tw@\count\z@
      \divide\count\z@\sixt@@n
      \count@\count\z@
      \multiply\count@\sixt@@n
      \advance\count\tw@-\count@
      \if\relax\noexpand#1% is command?
```

Store the command name with a space attached inside `\reserved@@b` in case we look at a robust definition.

```
        \edef\reserved@b{\expandafter\noexpand
                          \csname\expandafter\@gobble\string#1\space\endcsname}%
```

Test both `#1` and `#1␣` for containing `mathchar`.

```
        \edef\reserved@a
          {\noexpand\in@{\expandafter\@gobble\string\mathchar}%
                         {\meaning#1\expandafter\meaning\reserved@b}}%
        \reserved@a
```

Drop #1␣ in case it was defined before.

```
        \global\expandafter\let\reserved@b\@undefined
        \ifin@
          \expandafter\set@mathsymbol
             \csname sym#3\endcsname#1#2%
             {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \@font@info{Redeclaring math symbol \string#1}%
        \else
          \expandafter\ifx
            \csname\expandafter\@gobble\string#1\endcsname
            \relax
            \expandafter\set@mathsymbol
               \csname sym#3\endcsname#1#2%
               {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \else
            \@latex@error{Command `\string#1' already defined}\@eha
          \fi
        \fi
      \else
        \expandafter\set@mathchar
          \csname sym#3\endcsname#1#2
          {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      \fi
    \endgroup
  \else
    \@latex@error{Symbol font `#3' is not defined}\@eha
  \fi
}
\@onlypreamble\DeclareMathSymbol
```

(*End of definition for* `\DeclareMathSymbol`.)

`\set@mathchar`

```
\def\set@mathchar#1#2#3#4{%
  \global\mathcode`#2="\mathchar@type#3\hexnumber@#1#4\relax}
\@onlypreamble\set@mathchar
```

(*End of definition for* `\set@mathchar`.)

`\set@mathsymbol`

```
\def\set@mathsymbol#1#2#3#4{%
  \global\mathchardef#2"\mathchar@type#3\hexnumber@#1#4\relax}
\@onlypreamble\set@mathsymbol
```

(*End of definition for* `\set@mathsymbol`.)

```
%\def\mathsymbol#1#2#3{%
%  \@tempcnta=#3\relax
%  \@tempcntb\@tempcnta
%  \divide\@tempcnta\sixt@@n
%  \count@\@tempcnta
%  \multiply\count@\sixt@@n
%  \advance\@tempcntb-\count@
%  \mathchar"\mathchar@type#1\hexnumber@#2%
%             \hexnumber@\@tempcnta\hexnumber@\@tempcntb\relax}
```

```
%
%\def\DeclareMathAlphabetCharacter#1#2#3{%
%  \DeclareMathSymbol{#1}7{#2}{#3}}
```

\DeclareMathDelimiter

```
\def\DeclareMathDelimiter#1{%
  \if\relax\noexpand#1%
    \expandafter\@DeclareMathDelimiter
  \else
    \expandafter\@xxDeclareMathDelimiter
  \fi
  #1}
\@onlypreamble\DeclareMathDelimiter
```

(*End of definition for* `\DeclareMathDelimiter`.)

\@xxDeclareMathDelimiter

This macro checks if the second arg is a "math type" such as `\mathopen`. The undocumented original code didn't use math types when the delimiter was a single letter. For this reason the coding is a bit strange as it tries to support the undocumented syntax for compatibility reasons.

```
\def\@xxDeclareMathDelimiter#1#2#3#4{%
```

7 is the default value returned in the case that `\mathchar@type` is passed something unexpected, like a math symbol font name. We locally move `\mathalpha` out of the way so if you use that the right branch is taken. This will still fail if an explicit number 7 is used!

```
   \begingroup
    \let\mathalpha\mathord
    \ifnum7=\mathchar@type{#2}%
      \endgroup
```

If this branch is taken we have old syntax (5 arguments).

```
      \expandafter\@firstofone
    \else
```

If this branch is taken `\mathchar@type` is different from 7 so we assume new syntax. In this case we also use the arguments to set up the letter as a math symbol for the case where it is not used as a delimiter.

```
      \endgroup
      \DeclareMathSymbol#1{#2}{#3}{#4}%
```

Then we arrange that `\@xDeclareMathDelimiter` only gets `#1`, `#3`, `#4` ... as it does not expect a math type as argument.

```
      \expandafter\@firstoftwo
    \fi
    {\@xDeclareMathDelimiter#1}{#2}{#3}{#4}}
\@onlypreamble\@xxDeclareMathDelimiter
```

(*End of definition for* `\@xxDeclareMathDelimiter`.)

\@DeclareMathDelimiter

```
\def\@DeclareMathDelimiter#1#2#3#4#5#6{%
  \expandafter\in@\csname sym#3\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
```

```
    \expandafter\in@\csname sym#5\expandafter\endcsname
       \expandafter{\group@list}%
    \ifin@
      \begingroup
        \count\z@=#4\relax
        \count\tw@\count\z@
        \divide\count\z@\sixt@@n
        \count@\count\z@
        \multiply\count@\sixt@@n
        \advance\count\tw@-\count@
        \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      %
        \count\z@=#6\relax
        \count\tw@\count\z@
        \divide\count\z@\sixt@@n
        \count@\count\z@
        \multiply\count@\sixt@@n
        \advance\count\tw@-\count@
        \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      %
        \edef\reserved@a{\noexpand\in@
            {\expandafter\@gobble\string\delimiter}{\meaning#1}}%
        \reserved@a
        \ifin@
          \expandafter\set@mathdelimiter
             \csname sym#3\expandafter\endcsname
             \csname sym#5\endcsname#1#2%
             \reserved@c\reserved@d
          \@font@info{Redeclaring math delimiter \string#1}%
        \else
            \expandafter\ifx
            \csname\expandafter\@gobble\string#1\endcsname
            \relax
            \expandafter\set@mathdelimiter
              \csname sym#3\expandafter\endcsname
              \csname sym#5\endcsname#1#2%
              \reserved@c\reserved@d
          \else
            \@latex@error{Command `\string#1' already defined}\@eha
          \fi
        \fi
      \endgroup
    \else
      \@latex@error{Symbol font `#5' is not defined}\@eha
    \fi
  \else
    \@latex@error{Symbol font `#3' is not defined}\@eha
  \fi
}
\@onlypreamble\@DeclareMathDelimiter
```

(*End of definition for* `\@DeclareMathDelimiter`.)

`\@xDeclareMathDelimiter`

```
\def\@xDeclareMathDelimiter#1#2#3#4#5{%
  \expandafter\in@\csname sym#2\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
    \expandafter\in@\csname sym#4\expandafter\endcsname
       \expandafter{\group@list}%
    \ifin@
      \begingroup
        \count\z@=#3\relax
        \count\tw@\count\z@
        \divide\count\z@\sixt@@n
        \count@\count\z@
        \multiply\count@\sixt@@n
        \advance\count\tw@-\count@
        \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      %
        \count\z@=#5\relax
        \count\tw@\count\z@
        \divide\count\z@\sixt@@n
        \count@\count\z@
        \multiply\count@\sixt@@n
        \advance\count\tw@-\count@
        \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
        \expandafter\set@@mathdelimiter
           \csname sym#2\expandafter\endcsname\csname sym#4\endcsname#1%
           \reserved@c\reserved@d
      \endgroup
    \else
      \@latex@error{Symbol font ‘#4’ is not defined}\@eha
    \fi
  \else
    \@latex@error{Symbol font ‘#2’ is not defined}\@eha
  \fi
}
\@onlypreamble\@xDeclareMathDelimiter
```

(*End of definition for* `\@xDeclareMathDelimiter`.)

`\set@mathdelimiter` We have to end the definition of a math delimiter like `\lfloor` with a space and not with `\relax` as we did before, because otherwise constructs involving `\abovewithdelims` will prematurely end (pr/1329)

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\set@mathdelimiter}{make delimiters robust}%
\def\set@mathdelimiter#1#2#3#4#5#6{%
```

We use `\protected` not `\MakeRobust` so that `\bigl\lfloor` etc. works inside the argument of `\protected@edef`.

```
  \protected
  \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
    \hexnumber@#2#6 }%
%  \MakeRobust#3%
}
```

```
\@onlypreamble\set@mathdelimiter
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\set@mathdelimiter}{make delimiters robust}%
⟨latexrelease⟩
⟨latexrelease⟩\def\set@mathdelimiter#1#2#3#4#5#6{%
⟨latexrelease⟩   \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
⟨latexrelease⟩     \hexnumber@#2#6 }}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\set@mathdelimiter`.)

`\set@@mathdelimiter`

```
\def\set@@mathdelimiter#1#2#3#4#5{%
  \global\delcode`#3="\hexnumber@#1#4\hexnumber@#2#5\relax}
\@onlypreamble\set@@mathdelimiter
```

(*End of definition for* `\set@@mathdelimiter`.)

`\DeclareMathRadical`

```
\def\DeclareMathRadical#1#2#3#4#5{%
```

Below is a crude fix to make this macro work if `#1` is undefined or `\relax`. Should be improved!

```
  \expandafter\ifx
       \csname\expandafter\@gobble\string#1\endcsname
       \relax
     \let#1\radical
  \fi
  \edef\reserved@a{\noexpand\in@
       {\expandafter\@gobble\string\radical}{\meaning#1}}%
  \reserved@a
  \ifin@
    \expandafter\in@\csname sym#2\expandafter\endcsname
       \expandafter{\group@list}%
    \ifin@
      \expandafter\in@\csname sym#4\expandafter\endcsname
         \expandafter{\group@list}%
      \ifin@
        \begingroup
          \count\z@=#3\relax
          \count\tw@\count\z@
          \divide\count\z@\sixt@@n
          \count@\count\z@
          \multiply\count@\sixt@@n
          \advance\count\tw@-\count@
          \edef\reserved@c{%
            \hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \count\z@=#5\relax
          \count\tw@\count\z@
          \divide\count\z@\sixt@@n
          \count@\count\z@
```

```
          \multiply\count@\sixt@@n
          \advance\count\tw@-\count@
          \edef\reserved@d{%
            \hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
```

Coded inline instead of using `\set@mathradical`

```
%          \expandafter\set@mathradical
%             \csname sym#2\expandafter\endcsname
%             \csname sym#4\endcsname#1%
%             \reserved@c\reserved@d
          \xdef#1{\radical"\expandafter\hexnumber@
                                 \csname sym#2\endcsname\reserved@c
                              \expandafter\hexnumber@
                                 \csname sym#4\endcsname\reserved@d
                   \relax}%
        \endgroup
      \else
        \@latex@error{Symbol font ‘#4’ is not defined}\@eha
      \fi
    \else
      \@latex@error{Symbol font ‘#2’ is not defined}\@eha
    \fi
  \else
    \@latex@error{Command ‘\string#1’ already defined}\@eha
  \fi
}
\@onlypreamble\DeclareMathRadical
```

(*End of definition for* `\DeclareMathRadical`.)

Definition below was wrong it contained `\delimiter` !

```
def\set@mathradical#1#2#3#4#5{%
 \xdef#3{\radical"\hexnumber@#1#4\hexnumber@#2#5\relax}}
```

`\mathalpha` just a dummy currently

```
\let\mathalpha\relax
```

(*End of definition for* `\mathalpha`.)

`\mathchar@type`

```
\def\mathchar@type#1{%
  \ifodd 2#11 #1\else            % is this non-negative number?
    \ifx#1\mathord 0\else
     \ifx#1\mathop   1\else
       \ifx#1\mathbin 2\else
         \ifx#1\mathrel 3\else
           \ifx#1\mathopen 4\else
             \ifx#1\mathclose 5\else
               \ifx#1\mathpunct 6\else
                   7%            % anything else is variable ord
               \fi
             \fi
           \fi
         \fi
       \fi
```

```
      \fi
    \fi
  \fi}
\@onlypreamble\mathchar@type
```

(*End of definition for* `\mathchar@type`.)

`\DeclareSymbolFontAlphabet`

```
\def\DeclareSymbolFontAlphabet#1#2{%
   \expandafter\DeclareSymbolFontAlphabet@
     \csname \expandafter\@gobble\string#1\space\endcsname{#2}#1}
\@onlypreamble\DeclareSymbolFontAlphabet
```

(*End of definition for* `\DeclareSymbolFontAlphabet`.)

`\DeclareSymbolFontAlphabet@`

```
\def\DeclareSymbolFontAlphabet@#1#2#3{%
```

We use the switch `\if@tempswa` to decide if we can declare this symbol font alphabet.

```
    \@tempswatrue
```

First check if `#2` is known to be a symbol font

```
  \expandafter\in@\csname sym#2\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
```

Check if `#1` is defined as a math alphabet defined via `\DeclareMathAlphabet`:

```
    \expandafter\in@\expandafter#1\expandafter{\alpha@list}%
    \ifin@
```

If so remove it from the `\alpha@list` and from all math version macros.

```
      \@font@info{Redeclaring math alphabet \string#3}%
      \toks@{}%
      \def\alpha@elt##1##2##3{%
          \ifx##1#1\else\addto@hook\toks@{\alpha@elt##1##2##3}\fi}%
      \alpha@list
      \xdef\alpha@list{\the\toks@}%
```

Now we loop over all versions and remove the math alphabet:

```
      \def\version@elt##1{%
          \begingroup
            \toks@{}%
            \def\getanddefine@fonts####1####2{%
               \addto@hook\toks@{\getanddefine@fonts####1####2}}%
            \def\install@mathalphabet####1####2{%
               \ifx####1#1\else
                 \addto@hook\toks@{\install@mathalphabet
                                   ####1{####2}}\fi}%
            ##1%
            \xdef##1{\the\toks@}%
          \endgroup
          }%
      \version@list
    \else
```

If #3 is not defined as a math alphabet check if it is defined at all:

```
      \expandafter\ifx
      \csname\expandafter\@gobble\string#1\space\endcsname
      \relax
```

If it is undefined, fine otherwise check if it is a math alphabet defined via \DeclareSymbolFontAlphabet:

```
      \else
        \edef\reserved@a{%
          \noexpand\in@{\string\use@mathgroup}{\meaning#1}}%
        \reserved@a
        \ifin@
          \@font@info{Redeclaring math alphabet \string#3}%
        \else
```

Since the command #3 is defined to be something which is not a math alphabet we have to skip redefining it.

```
          \@tempswafalse
          \@latex@error{Command ‘\string#3’ already defined}\@eha
        \fi
      \fi
    \fi
  \else
```

Since the symbol font is not known we better skip defining this alphabet.

```
    \@tempswafalse
    \@latex@error{Unknown symbol font ‘#2’}\@eha
  \fi
  \if@tempswa
```

When we reach this point we are allowed to define #1 to be a symbol font math alphabet. This means that we have to set it to

\use@mathgroup ⟨*math-settings*⟩ \sym⟨*name*⟩

The ⟨*math-settings*⟩ are the one for the encoding that is used in the font shape where \sym⟨*name*⟩ is pointing to. This means that we have to get it from the information stored in \group@list. Thus we loop through that list after defining \group@elt in a suitable way.

```
    \def\group@elt##1##2{%
       \expandafter\ifx\csname sym#2\endcsname##1%
       \expandafter\reserved@a\string##2\@nil
       \fi}%
    \def\reserved@a##1##2/##3\@nil{%
       \def\reserved@a{##2}}%
    \group@list
    \toks@{\relax\ifmmode \else \non@alpherr#1\fi}%
    \edef#1{\the\toks@
            \noexpand\use@mathgroup
            \expandafter\noexpand\csname M@\reserved@a\endcsname
            \csname sym#2\endcsname}%
    \def#3{\protect#1}%
  \fi
}
\@onlypreamble\DeclareSymbolFontAlphabet@
⟨/2ekernel⟩
```

(*End of definition for* `\DeclareSymbolFontAlphabet@`.)

File 29
ltfssini.dtx

This file contains the top level LaTeX interface to the font selection scheme commands. See other parts of the LaTeX distribution, or *The LaTeX Companion* for higher level documentation of these commands.

1 NFSS Initialization

Finally, there are six commands that are to be used in LaTeX and that we will therefore protect against expansion at the wrong point: `\fontfamily`, `\fontseries`, `\fontshape`, `\fontsize`, `\selectfont`, and `\mathversion`.

```
⟨*2ekernel⟩
```

1.1 Providing math *versions*

LaTeX provides two *versions*. We call them normal and bold, respectively.

```
\DeclareMathVersion{normal}
\DeclareMathVersion{bold}
```

Now we define the standard font change commands. We don't allow the use of `\rmfamily` etc. in math mode.

(Actually most are now defined further down in the file.)

First the changes to another *family*:

```
%\DeclareRobustCommand\rmfamily
%        {\not@math@alphabet\rmfamily\mathrm
%         \fontfamily\rmdefault\selectfont}
%\DeclareRobustCommand\sffamily
%        {\not@math@alphabet\sffamily\mathsf
%         \fontfamily\sfdefault\selectfont}
%\DeclareRobustCommand\ttfamily
%        {\not@math@alphabet\ttfamily\mathtt
%         \fontfamily\ttdefault\selectfont}
```

Then the commands changing the *series*:

```
%\DeclareRobustCommand\bfseries
%        {\not@math@alphabet\bfseries\mathbf
%         \fontseries\bfdefault\selectfont}
%\DeclareRobustCommand\mdseries
%        {\not@math@alphabet\mdseries\relax
%         \fontseries\mddefault\selectfont}
\DeclareRobustCommand\upshape
        {\not@math@alphabet\upshape\relax
         \fontshape\updefault\selectfont}
```

Then the commands changing the *shape*:

```
\DeclareRobustCommand\slshape
        {\not@math@alphabet\slshape\relax
         \fontshape\sldefault\selectfont}
\DeclareRobustCommand\scshape
        {\not@math@alphabet\scshape\relax
```

```
         \fontshape\scdefault\selectfont}
\DeclareRobustCommand\itshape
        {\not@math@alphabet\itshape\mathit
         \fontshape\itdefault\selectfont}
```

2 Custom series settings for main document families

This section was introduced 2020/02/02 and for now we support a full rollback (may need splitting later).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                    {\DeclareFontSeriesDefault}{Custom series}%
```

One problem with the NFSS approach of handling the series axis turned out to be that (especially with respect to "boldness") different font families implemented different strategies. For example, with Computer Modern fonts you normally only have `bx` whereas most PostScript fonts offered only `b` but not `bx`. As a result LaTeX's standard setting for `\bfdefault` didn't work with such fonts, but if it got changed to produce `b`, then that didn't work with Computer Modern if the fonts got combined (e.g., using Computer Modern Typewriter with such fonts).

The solution back then was to provide substitution rules in the font `.fd` such that if a `bx` series got requested the `b` series got used. While this works in that particular case, it isn't a very general solution. For example, if you happen to have a font family that has several weights you may want to typeset the whole document in a somewhat lighter or darker font but if you then modify `\mddefault` to allow for this, then of course your change only works with that particular family but not with the typewriter or sans serif family you also want to use.

A better solution was provided by the `mweights` package by Bob Tennent that offers defaults on the level of the three main font families in the document: for "rm", "sf" and "tt" so that font packages could define defaults for the sans serif document font by providing `\bfseries@sf` which then was used when `\bfseries` got executed and the current family was the `\sffamily`.

`\DeclareFontSeriesDefault`

We now support this concept directly from within LaTeX and for use in font packages (or the document preamble) we offer `\DeclareFontSeriesDefault`. This declaration takes three arguments:

document family interface: Can either be `rm`, `sf` or `tt`. This is optional and if not given the overall default.

document series interface: Can be `md` or `bf`.

series value: This is the value that is going to be used with the combination is requested.

For example, `\DeclareFontSeriesDefault[rm]{bf}{sb}` would use `sb` (semi-bold) when `\rmfamily` `\bfseries` is asked for.

If used without the optional argument, e.g., `\DeclareFontSeriesDefault{bf}{b}` then this is like redefining `\bfdefault` or `\mddefault`.

If some family specify defaults aren't given, e.g. if there are no declarations for, say, `tt` then the format defaults of `\mddefault` and `\bfdefault` are assumed. If those are later changed this is *not* reflected![36]

`\DeclareFontSeriesDefault` The command to declare font series defaults for the "rm", "sf" or "tt" family.

```
\let\DeclareFontSeriesDefault\@undefined      % for rollback
\newcommand\DeclareFontSeriesDefault[3][]{%
  \expand@font@defaults
  \maybe@update@bfseries@defaults
  \maybe@update@mdseries@defaults
  \def\reserved@a{#1}%
```

No optional argument: set up general default.

```
  \ifx\reserved@a\@empty
    \ifcsname #2series\endcsname           % supported are
                                           % \[md/bf]default
```

Adding `\@empty` allows us to detect if the default gets redefined with `\renewcommand` or `\def` by the user.

```
      \expandafter\def
        \csname #2default\endcsname{#3\@empty}%
      \expandafter\def
        \csname #2default@previous\endcsname{#3\@empty}%
    \else
       \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
          {Mandatory first argument must be 'md'  or 'bf'.}
    \fi
```

Optional argument given, set up specific default.

```
  \else
    \ifcsname #2series@#1\endcsname          % supported are
                                             % \[md/bf]series@[rm/sf/tt]
      \expandafter\edef
         \csname #2series@#1\endcsname{#3}%
```

If the interface is used we remove the frozen kernel default. This way, we know that something was explicitly set up (even if the setup has the same value as the default).

```
      \expandafter\let
         \csname #2series@#1@kernel\endcsname\@undefined
    \else
       \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
          {Optional argument must be 'rm', 'sf', or 'tt'. \MessageBreak
           Mandatory first argument must be 'md'  or 'bf'.}
    \fi
  \fi
}
```

[36] I see no easy way to achieve this without compromising compatibility with existing packages that currently use mweights and directly define (some) of the `\mdseries@..` commands but not others.

(*End of definition for* `\DeclareFontSeriesDefault`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\DeclareFontSeriesDefault}{Custom series}%
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareFontSeriesDefault\@undefined      % for rollback
⟨latexrelease⟩\newcommand\DeclareFontSeriesDefault[3][]{%
⟨latexrelease⟩  \def\reserved@a{#1}%
⟨latexrelease⟩  \ifx\reserved@a\@empty
⟨latexrelease⟩    \ifcsname #2series\endcsname           % supported are
⟨latexrelease⟩                                           % \[md/bf]default
⟨latexrelease⟩      \expandafter\def
⟨latexrelease⟩        \csname #2default\endcsname{#3\@empty}%
⟨latexrelease⟩      \expandafter\def
⟨latexrelease⟩        \csname #2default@previous\endcsname{#3\@empty}%
⟨latexrelease⟩    \else
⟨latexrelease⟩       \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
⟨latexrelease⟩          {Mandatory first argument must be 'md'  or 'bf'.}
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \ifcsname #2series@#1\endcsname          % supported are
⟨latexrelease⟩                                             % \[md/bf]series@[rm/sf/tt]
⟨latexrelease⟩      \expandafter\edef
⟨latexrelease⟩         \csname #2series@#1\endcsname{#3}%
⟨latexrelease⟩      \expandafter\let
⟨latexrelease⟩         \csname #2series@#1@kernel\endcsname\@undefined
⟨latexrelease⟩    \else
⟨latexrelease⟩       \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
⟨latexrelease⟩          {Optional argument must be 'rm', 'sf', or 'tt'. \MessageBreak
⟨latexrelease⟩           Mandatory first argument must be 'md'  or 'bf'.}
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\DeclareFontSeriesDefault}{Custom series}%
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareFontSeriesDefault\@undefined
⟨latexrelease⟩\let\bfseries@rm\@undefined
⟨latexrelease⟩\let\bfseries@sf\@undefined
⟨latexrelease⟩\let\bfseries@tt\@undefined
⟨latexrelease⟩\let\bfseries@rm@kernel\@undefined
⟨latexrelease⟩\let\bfseries@sf@kernel\@undefined
⟨latexrelease⟩\let\bfseries@tt@kernel\@undefined
⟨latexrelease⟩\let\mdseries@rm\@undefined
⟨latexrelease⟩\let\mdseries@sf\@undefined
⟨latexrelease⟩\let\mdseries@tt\@undefined
⟨latexrelease⟩\expandafter\let\csname ver@mweights.sty\endcsname\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\@meta@family@list\@undefined
⟨latexrelease⟩\let\prepare@family@series@update\@undefined
⟨latexrelease⟩\let\update@series@target@value\@undefined
```

```
⟨latexrelease⟩
```

This is always called in `\document` so don't make it undefined.

```
⟨latexrelease⟩\let\init@series@setup\relax
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                    {\mdseries@rm}{Custom series}%
```

`\mdseries@rm` `\mdseries@sf` `\mdseries@tt` `\bfseries@rm` `\bfseries@sf` `\bfseries@tt`

We initialize the family specific default at the end of the format generation. Later on they may get overwritten in the preamble or a package via `\DeclareFontSeriesDefault` (or possibly directly).

Conceptual change: The `\bfdefault` will be `b` not `bx` because that is what it should be really for nearly every font except Computer/Latin Modern.

To account for the fact that by default we typeset in CM or LM we set up the `\bfseries@..` defaults to use `bx` instead.

This means that it behaves like before because if the default fonts are used then `\bfseries@rm` etc kick in and make `\textbf` use `bx`. However, if the font gets changed then `\bfdefault` will get used.

```
\def\bfseries@rm{bx}
\def\bfseries@sf{bx}
\def\bfseries@tt{bx}
```

Frozen version of the kernel defaults so we can see if they have changed.

```
\let\bfseries@rm@kernel\bfseries@rm
\let\bfseries@sf@kernel\bfseries@sf
\let\bfseries@tt@kernel\bfseries@tt
```

The default for the medium series is `m` and this will be interpreted as resetting both weight and width. To reset only one of them the virtual value `?m` and `m?` are available.

```
\def\mdseries@rm{m}
\def\mdseries@sf{m}
\def\mdseries@tt{m}
```

(*End of definition for* `\mdseries@rm` *and others.*)

`\series@change@debug`

For debugging, but right now none of this code is extracted. The idea is to have a separate package with debugging code one day.

```
⟨*debug⟩
\let\series@change@debug\typeout
\let\series@change@debug\@gobble
⟨/debug⟩
```

(*End of definition for* `\series@change@debug`*.*)

`\prepare@family@series@update`

This is core command that prepares for the family update. The big difference to the documented code above is that the nested `\ifx` statements seem to be missing. Instead we loop through an internal list that holds the names of the three meta families. This approach allows us to extend the mechanism at a later stage to allow for additional named meta families.

Here is the current definition of that list:

\@meta@family@list

```
\def\@meta@family@list{\@elt{rm}\@elt{sf}\@elt{tt}}
```

```
\def\prepare@family@series@update#1#2{%
 \if@forced@series
⟨+debug⟩ \series@change@debug{No series preparation (forced \f@series)\on@line}%
   \fontfamily#2%
 \else
⟨+debug⟩ \series@change@debug{Preparing for switching to #1 (#2)\on@line}%
   \expand@font@defaults
```

We prepare for changing the current series. We have to find it before changing the family as discussed above.

```
   \let\target@series@value\@empty
   \def\target@meta@family@value{#1}%
```

As the very last item in the meta family list we add `\@elt{??}` and define this pseudo meta family to be the current font family. So if none of the real meta families matched then this will match. This will cover the following case:

- `\bfseries` is called for a family using `bx` (e.g., CMR)
- Switch to a font family that is none of the meta families, e.g., via `\fontfamily{ptm}\allowbreak\v`
- Then none of the real meta families, match but the final `\@elt{??}` will.
- Therefore if the current series is `\mddefault` or `\bfdefault` it will be detected and the corresponding target series selected.

```
   \expandafter\edef\csname ??def@ult\endcsname{\f@family}%
```

To find it we loop over the meta family list with a suitable definition of `\@elt`.

```
   \let\@elt\update@series@target@value
      \@meta@family@list
```

Last resort pseudo meta family. Will only be looked at f none of the real ones have matched.

```
      \@elt{??}%
   \let\@elt\relax
```

That will figure out the correct series value to use without updating it. Now we can change the family.

```
   \fontfamily#2%
```

After that we update the series. That code is again like the one above.

```
   \ifx\target@series@value\@empty
⟨+debug⟩ \series@change@debug{Target series still empty ...}%
   \else
     \ifx \f@series\target@series@value
⟨+debug⟩ \series@change@debug{Target series unchanged:
⟨+debug⟩                      \f@series \space = \target@series@value}%
     \else
       \maybe@load@fontshape
⟨+debug⟩ \series@change@debug{Target series:
⟨+debug⟩                      \f@series \space -> \target@series@value}%
```

The \target@series@value may contain something like cm (coming from a default) and so we can't directly assign it to \f@series be have to drop any surplus m first.

```
%       \let\f@series\target@series@value
        \series@maybe@drop@one@m\target@series@value\f@series
      \fi
    \fi
  \fi
}
```

(*End of definition for* \prepare@family@series@update *and* \@meta@family@list.)

\update@series@target@value

In this macro used in the look you basically find the nested \ifxs from the outline above. The only difference is that is it is parameterized instead of being written out and only for one block of tests because the code is called repeatedly when looping over the meta family list. From the list we get each meta family name in turn.

```
\def\update@series@target@value#1{%
```

There is one additional test at the beginning, because the list contains all meta families and we need to ignore the case where current one from the list and target one are identical.

```
  \def\reserved@a{#1}%
  \ifx\target@meta@family@value\reserved@a   % rm -> rm do nothing
  \else
⟨+debug⟩ \series@change@debug{Trying to match #1: \csname#1def@ult\endcsname
⟨+debug⟩                      \space = \f@family\space ?}%
```

We only "do" something if the current font family matches the current meta family.

```
    \expandafter\ifx\csname#1def@ult\endcsname\f@family
```

If that's the case we know that this is the block that applies (only one meta family can match). So to speed things up we change \@elt so that the rest of the loop gets gobbled.

```
      \let\@elt\@gobble
```

Then we try to find the right new value for the series (as explained above). The two macros defined first are only there because we now need to use \csname and this way the code will be a little faster.

```
      \expandafter\let\expandafter\reserved@b
                      \csname mdseries@\target@meta@family@value\endcsname
      \expandafter\let\expandafter\reserved@c
                      \csname bfseries@\target@meta@family@value\endcsname
⟨+debug⟩\series@change@debug{Targets for mdseries and bfseries:
⟨+debug⟩                     \reserved@b\space and \reserved@c}%
```

This here is now identical to the nested \ifx block from the outline, except that it there appeared twice in \rmfamily. This is now covered by looping and stopping the loop when a match was found.

We have to sanitize the default value first because it may contain something like mc and that would never match \f@series because there it would be called c with the m dropped. It would be probably better to do that differently these days, but it is hard to adjust without causing a lot of issues, so we do the dropping in various places instead.

```
      \expandafter\series@maybe@drop@one@m
          \csname mdseries@#1\endcsname\reserved@d
      \ifx\reserved@d\f@series
⟨+debug⟩   \series@change@debug{mdseries@#1 matched -> \reserved@b}%
                                      \let\target@series@value\reserved@b
      \else
```

Again do some sanitizing.

```
        \expandafter\series@maybe@drop@one@m
           \csname bfseries@#1\endcsname\reserved@d
        \ifx\reserved@d\f@series
⟨+debug⟩  \series@change@debug{bfseries@#1 matched -> \reserved@c}%
                                        \let\target@series@value\reserved@c
      \else\ifx\f@series\mddef@ult    \let\target@series@value\reserved@b
⟨+debug⟩  \series@change@debug{mddef@ult matched -> \reserved@b}%
      \else\ifx\f@series\bfdef@ult    \let\target@series@value\reserved@c
⟨+debug⟩  \series@change@debug{bfdef@ult matched -> \reserved@c}%
      \fi\fi\fi\fi
    \fi
  \fi
}
```

(*End of definition for* `\update@series@target@value`.)

`\init@series@setup` This is code to be run at begin document ...

```
\def\init@series@setup{%
```

We only want `bx` in `\bfseries@rm` if the roman font is Computer Modern or Latin Modern, otherwise it should be `b`. It was set to `bx` in the kernel so that any font use with the default families in the preamble get this value. Now at the real document start we check if the fonts have been changed. If there was a `\DeclareFontSeriesDefault` declaration or `\bfseries@rm` was directly altered then it differs from `\bfseries@rm@kernel` and we do nothing. Otherwise we check if `\rmdefault` is one of the CM/LM font families and if so we keep `bx` otherwise we change it to `b`.

This approach doesn't cover one case: CM/LM got changed to a different family that supports `bx`, but the support package for that family used `\def\bfseries@rm{bx}` instead of using `\DeclareFontSeriesDefault`. In that case the code here changes it to `b`. Solution: use the `\DeclareFontSeriesDefault` interface.

```
  \ifx\bfseries@rm@kernel\bfseries@rm
     \expandafter\in@\expandafter{\rmdefault}%
                      {cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmtt}%
    \ifin@ \else \def\bfseries@rm{b}\fi\fi
```

Same approach for `\bfseries@sf` and `\bfseries@tt`:

```
  \ifx\bfseries@sf@kernel\bfseries@sf
    \expandafter\in@\expandafter{\sfdefault}%
                     {cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmtt}%
    \ifin@ \else \def\bfseries@sf{b}\fi\fi
  \ifx\bfseries@tt@kernel\bfseries@tt
    \expandafter\in@\expandafter{\ttdefault}%
                     {cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmtt}%
    \ifin@ \else \def\bfseries@tt{b}\fi\fi
```

If the document preamble has changed the `\familydefault` or if the if the `\rmdefault` contains a new font family, we may have to adjust the series defaults accordingly, before starting typesetting.

Similarly, if the user has changed the `\mddefault` or the medium series for the family selected as document font we may also have to adjust the `\seriesdefault`.

On the other hand if the document font is still CM or LM then `\bfdefault` is wrong, because it is now saying `b` and not `bx` as it should for such fonts.

To fix all this we first run \reset@font (the internal kernel name for \normalfont). This will set up the document encoding, family, series and shape based on the current values of \encodingdefault, \familydefault, \seriesdefault and \shapedefault. However, if the family (from \familydefault) has special medium default we should switch to that (and not use what is current value from \seriesdefault). This can be achieved by afterwards calling \mediumseries and then changing \seriesdefault to the now current series value (in \f@series).

But what should happen if \seriesdefault got explicitly changed? In that case the explicit change should survive and we should not alter \seriesdefault. This is solved by comparing the current value of \seriesdefault with a kernel version saved in the format and if they differ we do not call \mdseries or change \seriesdefault.

```
  \reset@font
  \ifx\seriesdefault\seriesdefault@kernel
    \mdseries
    \let\seriesdefault\f@series
  \fi
}%
```

(*End of definition for* \init@series@setup.)

As the kernel code now implements the same functionality as mweights, albeit internally coded slightly differently, that package shouldn't be loaded any more. We therefore pretend that it already got loaded. Thus, a font package that tries to load it and then sets \mdseries@.., etc. will continue to work but will now use the kernel code.

Of course, mid-term such package should probably use \DeclareFontSeriesDefault instead of making using low-level definitions.

```
\expandafter\let\csname ver@mweights.sty\endcsname\fmtversion
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\mdseries@rm}{Custom series}%
⟨latexrelease⟩
⟨latexrelease⟩\let\bfseries@rm\@undefined
⟨latexrelease⟩\let\bfseries@sf\@undefined
⟨latexrelease⟩\let\bfseries@tt\@undefined
⟨latexrelease⟩\let\bfseries@rm@kernel\@undefined
⟨latexrelease⟩\let\bfseries@sf@kernel\@undefined
⟨latexrelease⟩\let\bfseries@tt@kernel\@undefined
⟨latexrelease⟩\let\mdseries@rm\@undefined
⟨latexrelease⟩\let\mdseries@sf\@undefined
⟨latexrelease⟩\let\mdseries@tt\@undefined
⟨latexrelease⟩\expandafter\let\csname ver@mweights.sty\endcsname\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\@meta@family@list\@undefined
⟨latexrelease⟩\let\prepare@family@series@update\@undefined
⟨latexrelease⟩\let\update@series@target@value\@undefined
⟨latexrelease⟩
```

This is always called in \document so don't make it undefined.

```
⟨latexrelease⟩\let\init@series@setup\relax
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                  {\bfseries}{Custom series with hooks}%
```

\bfseries This document command switches to the bold series.

```
\DeclareRobustCommand\bfseries{%
  \not@math@alphabet\bfseries\mathbf
```

In the original NFSS definition it then called \fontseries with the value \bfdefault. In the new scheme we have more alternatives and therefore check if the current family (\f@family) is the current \rmdef@ult, \sfdef@ult or \ttdef@ult and the select the correct family default in that case.

```
  \expand@font@defaults
  \maybe@update@bfseries@defaults
```

```
  \ifx\f@family\rmdef@ult      \fontseries\bfseries@rm
  \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
  \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt
```

If not \bfdefault is used.

```
  \else                        \fontseries\bfdefault
  \fi\fi\fi
```

This hook in contrast is always executed.

```
  \UseHook{bfseries}%
  \selectfont
}
```

(*End of definition for* \bfseries.)

\maybe@update@bfseries@defaults If \bfdefault and \bfdefault@previous are different then the default got changed directly through the legacy interface (i.e., via \def or \renewcommand). In that case we reset all meta family defaults so that the document behaves like it was the case before the new mechanism was introduced.

```
\def\maybe@update@bfseries@defaults{%
  \ifx\bfdefault\bfdefault@previous\else
```

We add \@empty and then let \bfdefault@previous to \bfdefault so that we can detect any further change.

```
    \expandafter\def\expandafter\bfdefault
                    \expandafter{\bfdefault\@empty}%
    \let\bfdefault@previous\bfdefault
```

And we reset the meta family defaults (\bfdef@ult is an expanded version of \bfdefault).

```
    \let\bfseries@rm\bfdef@ult
    \let\bfseries@sf\bfdef@ult
    \let\bfseries@tt\bfdef@ult
```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add resets here. Not that this hook is only run when resets are necessary.

```
    \UseHook{bfseries/defaults}%
  \fi
}
```

(*End of definition for* \maybe@update@bfseries@defaults.)

\mdseries This document command switches to the medium series.

```
\DeclareRobustCommand\mdseries{%
  \not@math@alphabet\mdseries\relax
  \expand@font@defaults
  \maybe@update@mdseries@defaults
  \ifx\f@family\rmdef@ult      \fontseries\mdseries@rm
    \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
    \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
    \else                        \fontseries\mddefault
  \fi\fi\fi
  \UseHook{mdseries}%
  \selectfont
}
```

(*End of definition for* \mdseries.)

\maybe@update@mdseries@defaults

```
\def\maybe@update@mdseries@defaults{%
  \ifx\mddefault\mddefault@previous\else
    \expandafter\def\expandafter\mddefault\expandafter{\mddefault\@empty}%
    \let\mddefault@previous\mddefault
    \let\mdseries@rm\mddef@ult
    \let\mdseries@sf\mddef@ult
    \let\mdseries@tt\mddef@ult
```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add resets here.

```
    \UseHook{mdseries/defaults}%
  \fi
}
```

(*End of definition for* \maybe@update@mdseries@defaults.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                 {\bfseries}{Custom series with hooks}%
⟨latexrelease⟩
⟨latexrelease⟩\let\maybe@update@bfseries@defaults\@undefined
⟨latexrelease⟩\let\maybe@update@mdseries@defaults\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\bfseries{%
⟨latexrelease⟩  \not@math@alphabet\bfseries\mathbf
⟨latexrelease⟩  \expand@font@defaults
⟨latexrelease⟩  \ifx\bfdefault\bfdefault@previous\else
⟨latexrelease⟩    \expandafter\def\expandafter\bfdefault
⟨latexrelease⟩                     \expandafter{\bfdefault\@empty}%
⟨latexrelease⟩    \let\bfdefault@previous\bfdefault
⟨latexrelease⟩    \let\bfseries@rm\bfdef@ult
⟨latexrelease⟩    \let\bfseries@sf\bfdef@ult
⟨latexrelease⟩    \let\bfseries@tt\bfdef@ult
⟨latexrelease⟩    \UseHook{bfseries/defaults}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩    \ifx\f@family\rmdef@ult      \fontseries\bfseries@rm
⟨latexrelease⟩    \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
```

```
⟨latexrelease⟩    \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt
⟨latexrelease⟩    \else                        \fontseries\bfdefault
⟨latexrelease⟩    \fi\fi\fi
⟨latexrelease⟩  \UseHook{bfseries}%
⟨latexrelease⟩  \selectfont
⟨latexrelease⟩}
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\mdseries{%
⟨latexrelease⟩  \not@math@alphabet\mdseries\relax
⟨latexrelease⟩  \expand@font@defaults
⟨latexrelease⟩  \ifx\mddefault\mddefault@previous\else
⟨latexrelease⟩    \expandafter\def\expandafter\mddefault\expandafter{\mddefault\@empty}%
⟨latexrelease⟩    \let\mddefault@previous\mddefault
⟨latexrelease⟩    \let\mdseries@rm\mddef@ult
⟨latexrelease⟩    \let\mdseries@sf\mddef@ult
⟨latexrelease⟩    \let\mdseries@tt\mddef@ult
⟨latexrelease⟩    \UseHook{mdseries/defaults}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩    \ifx\f@family\rmdef@ult      \fontseries\mdseries@rm
⟨latexrelease⟩    \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
⟨latexrelease⟩    \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
⟨latexrelease⟩    \else                        \fontseries\mddefault
⟨latexrelease⟩    \fi\fi\fi
⟨latexrelease⟩  \UseHook{mdseries}%
⟨latexrelease⟩  \selectfont
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\bfseries}{Custom series with hooks}%
⟨latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\bfseries{%
⟨latexrelease⟩  \not@math@alphabet\bfseries\mathbf
⟨latexrelease⟩  \expand@font@defaults
⟨latexrelease⟩    \ifx\f@family\rmdef@ult      \fontseries\bfseries@rm
⟨latexrelease⟩    \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
⟨latexrelease⟩    \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt
⟨latexrelease⟩    \else                        \fontseries\bfdefault
⟨latexrelease⟩    \fi\fi\fi
⟨latexrelease⟩  \selectfont
⟨latexrelease⟩}
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\mdseries{%
⟨latexrelease⟩  \not@math@alphabet\mdseries\relax
⟨latexrelease⟩  \expand@font@defaults
⟨latexrelease⟩    \ifx\f@family\rmdef@ult      \fontseries\mdseries@rm
⟨latexrelease⟩    \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
⟨latexrelease⟩    \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
⟨latexrelease⟩    \else                        \fontseries\mddefault
⟨latexrelease⟩    \fi\fi\fi
⟨latexrelease⟩  \selectfont
⟨latexrelease⟩}
⟨latexrelease⟩
⟨latexrelease⟩
```

```
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\bfseries}{Custom series with hooks}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\bfseries
⟨latexrelease⟩        {\not@math@alphabet\bfseries\mathbf
⟨latexrelease⟩         \fontseries\bfdefault\selectfont}
⟨latexrelease⟩\DeclareRobustCommand\mdseries
⟨latexrelease⟩        {\not@math@alphabet\mdseries\relax
⟨latexrelease⟩         \fontseries\mddefault\selectfont}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩



⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\expand@font@defaults}{Custom series with hooks}%
```

`\expand@font@defaults` `\rm@def@ult` `\sf@def@ult` `\tt@def@ult` `\md@def@ult` `\bf@def@ult`

The family specific defaults are fully expanded, i.e., they are defined via `\edef` inside `\DeclareFontSeriesDefault`. However, the overall defaults, e.g., `\bfdefault` may have been redefined by the user and thus may not be fully expanded. So to enable reliable comparison we make expanded versions of them. That we rerun each time. The alternative would be to only allow for changes before begin document.

```
\def\expand@font@defaults{%
  \edef\rmdef@ult{\rmdefault}%
  \edef\sfdef@ult{\sfdefault}%
  \edef\ttdef@ult{\ttdefault}%
```

The series defaults may contain some surplus `m` that we need to drop here.

```
  \series@maybe@drop@one@m\bfdefault\bfdef@ult
  \series@maybe@drop@one@m\mddefault\mddef@ult
```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add additional code here.

```
  \UseHook{expand@font@defaults}%
}
```

(*End of definition for* `\expand@font@defaults` *and others.*)

`\rmfamily` Here are the document level commands for changing the main font families, or rather, here is a documented outline of the code, the actual code is then streamlined and somewhat generalized.

```
DeclareRobustCommand\rmfamily{%
 \not@math@alphabet\rmfamily\mathrm
```

If families are changed then we have to do a bit more work. In the original NFSS implementation a family change kept encoding, series shape and size unchanged but now we can't any longer simply reuse the current series value. Instead we may have to change it from one family default to the next.

```
\expand@font@defaults
```

We have to do the testing while the current family is still unchanged but we have to do the adjustment of the series after it got changed (because the new family might have different sets of shapes available and we certainly don't want to see substitution going on. So we use `\target@series@value` to hold the target series (if any).

```
\let\target@series@value\@empty
```

Thus, if the current family is the sans family

```
\ifx\f@family\sfdef@ult
```

and if we using the medium series of the sans family

```
      \ifx\f@series\mdseries@sf
```

then lets switch to the medium series for the serif family

```
                                        \let\target@series@value\mdseries@rm
```

and if we use the bold series of the sans family switch to the bold default of the serif family:

```
      \else\ifx\f@series\bfseries@sf  \let\target@series@value\bfseries@rm
```

However, the sans family may not have any specific defaults set, so we also compare with the overall defaults.

```
      \else\ifx\f@series\mddef@ult    \let\target@series@value\mdseries@rm
      \else\ifx\f@series\bfdef@ult    \let\target@series@value\bfseries@rm
```

If neither test was true we leave the series alone. This way a special manual setting such as `\fontseries{lc}` is not undone if the family changes (of course there may not be any support for it in the new family but then the NFSS substitution kicks in and sorts it out).

```
      \fi\fi\fi\fi
```

We need to do the same if the current family is the typewriter family:

```
\else\ifx\f@family\ttdef@ult
      \ifx\f@series\mdseries@tt       \let\target@series@value\mdseries@rm
      \else\ifx\f@series\bfseries@tt  \let\target@series@value\bfseries@rm
      \else\ifx\f@series\mddef@ult    \let\target@series@value\mdseries@rm
      \else\ifx\f@series\bfdef@ult    \let\target@series@value\bfseries@rm
      \fi\fi\fi\fi
\fi\fi
```

With these preparations for series out of the way we can now change the font family to `\rmdefault`.

```
\fontfamily\rmdefault
```

If \target@series@value is still empty there is nothing more to do other than selecting the new family. However, if not then we should update the font series now as well. But there is one further subtle issue. We may not have loaded an .fd file for our target font family yet. In the past that was done in \selectfont if necessary but since we are now doing all the comparisons in \fontseries we need to make sure that the font family specifications are already loaded prior to calling \fontseries.

```
\ifx\target@series@value\@empty \else
  \maybe@load@fontshape
```

Updating the series in this case means directly changing \f@series to the target value. We don't want to go through \fontseries because that would apply the mappings and then bx + b would keep bx instead of changing to b as desired. as

```
  \let\f@series\target@series@value
\fi
\selectfont}
```

So now for the real definition: most of the code above gets delegated to a helper command \prepare@family@series@update so that the definition becomes again fairly short. In addition we add a hook, mainly for our Japanese friends so that the code can be extended prior to the call to \selectfont.

```
\DeclareRobustCommand\rmfamily{%
   \not@math@alphabet\rmfamily\mathrm
```

This holds all the code discussed above, first argument is the meta family, i.e., rm in this case, and second argument is the default family name, e.g., cmr indirectly accessed via \rmdefault. This is calling \fontfamily and if necessary \fontseries as outline above.

```
   \prepare@family@series@update{rm}\rmdefault
```

Then comes the hook code (by default a no-op) and finally the call to \selectfont.

```
   \UseHook{rmfamily}%
   \selectfont}
```

The definitions for \sffamily and \ttfamily are similar, the differences are only in what font families get checked.

\sffamily
\ttfamily

```
\DeclareRobustCommand\sffamily{%
   \not@math@alphabet\sffamily\mathsf
   \prepare@family@series@update{sf}\sfdefault
   \UseHook{sffamily}%
   \selectfont}

\DeclareRobustCommand\ttfamily{%
   \not@math@alphabet\ttfamily\mathtt
   \prepare@family@series@update{tt}\ttdefault
   \UseHook{ttfamily}%
   \selectfont}
```

(*End of definition for* \rmfamily*,* \sffamily*, and* \ttfamily*.*)

rmfamily
sffamily
ttfamily
normalfont
expand@font@defaults
bfseries
bfseries/defaults
mdseries
mdseries/defaults

Declare the hooks used above.

```
\NewHook{rmfamily}
\NewHook{sffamily}
\NewHook{ttfamily}
\NewHook{normalfont}
\NewHook{expand@font@defaults}
\NewHook{bfseries}
\NewHook{bfseries/defaults}
\NewHook{mdseries}
\NewHook{mdseries/defaults}
```

(*End of definition for* `rmfamily` *and others.*)

\@rmfamilyhook
\@sffamilyhook
\@ttfamilyhook
\@defaultfamilyhook

These four hooks have legacy versions used in 2020/02/02 so we should support them until they aren't any longer used.

By default the hooks do nothing.

```
\let\@rmfamilyhook\@empty
\let\@sffamilyhook\@empty
\let\@ttfamilyhook\@empty
\let\@defaultfamilyhook\@empty   %FMi sort out
```

(*End of definition for* `\@rmfamilyhook` *and others.*)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {\expand@font@defaults}{Custom series with hooks}%
⟨latexrelease⟩
⟨latexrelease⟩\def\expand@font@defaults{%
⟨latexrelease⟩   \edef\rmdef@ult{\rmdefault}%
⟨latexrelease⟩   \edef\sfdef@ult{\sfdefault}%
⟨latexrelease⟩   \edef\ttdef@ult{\ttdefault}%
⟨latexrelease⟩   \edef\bfdef@ult{\bfdefault}%
⟨latexrelease⟩   \edef\mddef@ult{\mddefault}%
⟨latexrelease⟩   \edef\famdef@ult{\familydefault}%
⟨latexrelease⟩}
⟨latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\rmfamily{%
⟨latexrelease⟩    \not@math@alphabet\rmfamily\mathrm
⟨latexrelease⟩    \prepare@family@series@update{rm}\rmdefault
⟨latexrelease⟩    \@rmfamilyhook
⟨latexrelease⟩    \selectfont}
⟨latexrelease⟩\DeclareRobustCommand\sffamily{%
⟨latexrelease⟩    \not@math@alphabet\sffamily\mathsf
⟨latexrelease⟩    \prepare@family@series@update{sf}\sfdefault
⟨latexrelease⟩    \@sffamilyhook
⟨latexrelease⟩    \selectfont}
⟨latexrelease⟩\DeclareRobustCommand\ttfamily{%
⟨latexrelease⟩    \not@math@alphabet\ttfamily\mathtt
⟨latexrelease⟩    \prepare@family@series@update{tt}\ttdefault
⟨latexrelease⟩    \@ttfamilyhook
⟨latexrelease⟩    \selectfont}
⟨latexrelease⟩\let\@rmfamilyhook\@empty
⟨latexrelease⟩\let\@sffamilyhook\@empty
```

```
⟨latexrelease⟩\let\@ttfamilyhook\@empty
⟨latexrelease⟩
```

```
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\expand@font@defaults}{Custom series with hooks}%
⟨latexrelease⟩
⟨latexrelease⟩\let\expand@font@defaults\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\bfseries
⟨latexrelease⟩        {\not@math@alphabet\bfseries\mathbf
⟨latexrelease⟩         \fontseries\bfdefault\selectfont}
⟨latexrelease⟩\DeclareRobustCommand\mdseries
⟨latexrelease⟩        {\not@math@alphabet\mdseries\relax
⟨latexrelease⟩         \fontseries\mddefault\selectfont}
⟨latexrelease⟩\DeclareRobustCommand\rmfamily
⟨latexrelease⟩        {\not@math@alphabet\rmfamily\mathrm
⟨latexrelease⟩         \fontfamily\rmdefault\selectfont}
⟨latexrelease⟩\DeclareRobustCommand\sffamily
⟨latexrelease⟩        {\not@math@alphabet\sffamily\mathsf
⟨latexrelease⟩         \fontfamily\sfdefault\selectfont}
⟨latexrelease⟩\DeclareRobustCommand\ttfamily
⟨latexrelease⟩        {\not@math@alphabet\ttfamily\mathtt
⟨latexrelease⟩         \fontfamily\ttdefault\selectfont}
⟨latexrelease⟩
⟨latexrelease⟩\let\@rmfamilyhook\@undefined
⟨latexrelease⟩\let\@sffamilyhook\@undefined
⟨latexrelease⟩\let\@ttfamilyhook\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨∗2ekernel⟩
```

`\IfFontSeriesContextTF` With the ability for `\bfseries` or `\mdseries` to be mapped to different NFSS axis values it becomes important to have the ability to determine the current context as we can no longer look at `\f@series` to answer a question such as "am I currently typesetting in a bold typeface?"

This is provided by the test `\IfFontSeriesContextTF`. It takes three arguments:

- The context we try to check (either `bf` for bold or `md` for medium, i.e., the same that can go into the first mandatory argument of `\DeclareFontSeriesDefault`),
- what to do if we are in this context (true case) and
- what to do if we are not (false case).

This allows you to define commands like `\IfBold`, e.g.,

```
\NewDocumentCommand\IfBold{mm}{\IfFontSeriesContextTF{bf}{#1}{#2}}
```

and then do

```
This is \IfBold{bold}{non-bold} text.
```

and get the appropriate result.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                    {\IfFontSeriesContextTF}{Font series context}%
\DeclareRobustCommand\IfFontSeriesContextTF[1]{%
  \expand@font@defaults
```

In the beginning we haven't found the context we are looking for.

```
  \@font@series@contextfalse
```

We store the requested context away for use in the tests.

```
  \def\requested@test@context{#1}%
```

The next definition is there to ensure that get a final match during testing even if the current family is non of the meta families (`rm`, `sf` or `tt`). This will then basically tests if the current font family matches the overall default.

```
  \expandafter\edef\csname ??def@ult\endcsname{\f@family}%
```

Then we run through the meta family list (currently containing just the three values) followed by the artificial meta family `??` and test each of them in turn using `\test@font@series@context` as the testing command.

```
  \let\@elt\test@font@series@context
      \@meta@family@list
      \@elt{??}%
  \let\@elt\relax
```

Following that we evaluate the status of `\if@font@series@context` to determine which of the remaining arguments (true/false case) we have to execute.

```
  \if@font@series@context
  \expandafter\@firstoftwo
  \else
  \expandafter\@secondoftwo
  \fi
}
```

(*End of definition for* `\IfFontSeriesContextTF`.)

`\test@font@series@context` This tests the context (stored in `\requested@test@context`) and updates the boolean if the right context is found.

```
\def\test@font@series@context#1{%
```

First task is to figure out whether the current family matches `\rmfamily`, `\sffamily`, etc. so in `\reserved@a` we store the value of `\rmdef@ult` (or whatever the given meta family is) and compare that to `\f@family`.

```
  \edef\reserved@a{\csname #1def@ult\endcsname}%
  \ifx\f@family\reserved@a
```

If they match we have found the right meta family so we don't need to test any of the remaining meta family and therefore change `\@elt` to `\@gobble`.

```
    \let\@elt\@gobble
```

Now we have to test if \f@series matches the requested context (e.g., whether \bfseries@rm has that value if the current meta family is rm and we are looking for the bf context).

```
    \expandafter\ifx
                \csname\requested@test@context series@#1\endcsname\f@series
```

If yes we change the boolean and are done.

```
      \@font@series@contexttrue
```

If not then maybe the reason is that nothing special was set up for that meta family so we also check now check if \f@series matches the overall default (e.g., \bfdef@ult if we are looking for the bold context). If that matches we change the boolean.

```
    \else
      \expandafter\ifx
                  \csname\requested@test@context def@ult\endcsname\f@series
        \@font@series@contexttrue
  \fi\fi\fi
}
```

(*End of definition for* \test@font@series@context.)

\if@font@series@context The boolean to signal if we found the requested font series context.

```
\newif\if@font@series@context
```

(*End of definition for* \if@font@series@context.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\IfFontSeriesContextTF}{Font series context}%
⟨latexrelease⟩
⟨latexrelease⟩\let\IfFontSeriesContextTF\@undefined
⟨latexrelease⟩\let\test@font@series@context\@undefined
⟨latexrelease⟩\let\if@font@series@context\@undefined
⟨latexrelease⟩\let\@font@series@contexttrue\@undefined
⟨latexrelease⟩\let\@font@series@contextfalse\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

3 Supporting nested emphasis

By default LaTeX 2_ε supports two levels of nested emphasis: if the current font has an upright shape then it switches to \itshape otherwise to \eminnershape (which defaults to \upshape). This means nested emphasis will oscillate between italic and upright shapes.

Sometimes it would be nice to allow for a more lengthy sequence, but instead of providing a fixed one LaTeX now offers a general mechanism that allows to define arbitrary sequences.

\DeclareEmphSequence
\emforce

This declaration expects a comma separated list of (font) change declarations corresponding to increasing levels of emphasis. The mechanism tries to be "smart" and verifies that the declarations actually alter the font. If not it will ignore this level and tries the next one—the assumption being that there was a manual font change in the document

to the font that is now supposed to be used for emphasis. Of course, this only works if the declarations in the list actually change the font and not, say, just the color. In such a case one has to use `\emforce` to which directs the mechanism to use the level even if the font attributes haven't changed.

`\emreset` If the nesting is so deep, that the specified levels are exhausted then `\emreset` is used as a final set of declarations (which by default returns back to the upright shape). Any additional nesting levels will then reuse the list from its beginning.

`\DeclareEmphSequence` `\DeclareEmphSequence` expects a clist of declaration. Spaces in the argument are dropped to avoid spurious spaces in the output. The declarations are additive. At the very end the shape is reset using `\emreset` and `\emforce` so that this case is never skipped.[37] Further nested calls restart at the beginning.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                    {\DeclareEmphSequence}{Nested emph}%
\def\DeclareEmphSequence#1{%
  \protected@edef\emfontdeclare@clist{\zap@space#1, \@empty\emforce\emreset}%
}
```

By default the it is empty, in which case `\eminnershape` is used by LaTeX.

```
\let\emfontdeclare@clist\@empty
```

(*End of definition for* `\DeclareEmphSequence`.)

`\emrest` Reset the font to upright and upper/lower case. With the default rules using `\shapedefault` does that for us but to be on the safe side we do it like this:

```
\DeclareRobustCommand\emreset{\upshape\ulcshape}
```

(*End of definition for* `\emrest`.)

`\em` The new definition for `\em` (and implicitly `\emph`) is the same as before as long as `\emfontdeclare@clist` is empty.

```
\DeclareRobustCommand\em{%
  \@nomath\em
  \ifx\emfontdeclare@clist\@empty
    \ifdim \fontdimen\@ne\font >\z@
      \eminnershape \else \itshape \fi
  \else
```

But if not we use the list to decide how to do emphasis.

We use the current font to check if the declarations have any effect, so even a size change is allowed and identified as a modification (but a color change, for example, isn't). So first we save the current status.

```
    \edef\em@currfont{\csname\curr@fontshape/\f@size\endcsname}%
```

Then we grab the next element from the list and check if it can be used.

```
    \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
  \fi
}
```

[37] Maybe we should not add `\emforce` but allow that case to be skipped as well. Of course, that might result in an endless loop if somebody defines a sequence without any font change and without `\emforce` but ...

```
\def\eminnershape{\upshape}
```

(*End of definition for* `\em`.)

`\do@emfont@update` We know that the list (if not empty) has at least 2 elements separated by a comma, so we pick up the first in `#1` and the rest in `#2`.

```
\def\do@emfont@update#1,#2\do@emfont@update{%
```

First action is to alter the list and move the first entry to the end

```
  \def\emfontdeclare@clist{#2,#1}%
```

Then we execute the current declaration. Appending `\selectfont` means one can write just `\fontshape{it}` and that works then too.

```
%  \typeout{Use: \detokenize{#1}}%
  #1\selectfont
```

We then compare the current font with our saved version, but with a slight twist: we add `\em@force` at the end of the name. Normally this is empty so has no effect but if there was an `\emforce` as part of `#1` it will append a `/` to the font name (making it invalid) thus this will then always fail the test.

If the test fails we are done and the declarations will be used. Otherwise we will try the next declaration in the sequence.

```
  \expandafter\ifx\csname \curr@fontshape/\f@size\em@force
```

For the comparison with `\ifx` we have to expand `\em@currfont` once as the relevant info is inside.

```
                          \expandafter\endcsname
                    \em@currfont
  \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
```

If `\emforce` was used, we have to undo its effect:

```
  \else
    \let\em@force\@empty
  \fi
}
```

(*End of definition for* `\do@emfont@update`.)

`\emforce` `\em@force` The definition of `\emforce` is simple: change `\em@force` to make the above test always invalid.

```
\protected\def\emforce{\def\em@force{/}}
\let\em@force\@empty
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\emforce` *and* `\em@force`.)

`\em` `\eminnershape` These are the older definitions for `\em`, prior to 2020.

We also have to define the *emphasize* font change command (i.e. `\em`). This command will look is the current font is sloped (i.e. has a positive `\fontdimen1`) and will then select either `\upshape` or `\itshape`.

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\DeclareEmphSequence}{Nested emph}%
⟨latexrelease⟩\let\DeclareEmphSequence\@undefined
⟨latexrelease⟩\let\emfontdeclare@clist\@undefined
⟨latexrelease⟩\let\emreset\@undefined
⟨latexrelease⟩\let\do@emfont@update\@undefined
```

```
⟨latexrelease⟩\let\emforce\@undefined
⟨latexrelease⟩\let\em@force\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\em
⟨latexrelease⟩        {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
⟨latexrelease⟩                       \eminnershape \else \itshape \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\DeclareEmphSequence}{Nested emph}%
⟨latexrelease⟩\DeclareRobustCommand\em
⟨latexrelease⟩        {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
⟨latexrelease⟩                       \upshape \else \itshape \fi}%
⟨latexrelease⟩\let\eminnershape\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\em` *and* `\eminnershape`.)

`\not@math@alphabet` This function generates an error message when it is called in math mode. The same function should be defined in `newlfont.sty`.

```
\def\not@math@alphabet#1#2{%
   \relax
   \ifmmode
     \@latex@error{Command \noexpand#1invalid in math mode}%
        {%
         Please
         \ifx#2\relax
            define a new math alphabet^^J%
            if you want to use a special font in math mode%
          \else
```

We have to a `\noexpand` below to prevent expansion of `#2`. In case of `#1` we can omit this (due to the current definition of robust commands since they do come out right there :-).

```
            use the math alphabet \noexpand#2instead of
            the #1command%
         \fi
         .
        }%
   \fi}
```

(*End of definition for* `\not@math@alphabet`.)

Finally we provide two abbreviations to switch to the LaTeX *versions*.

```
\DeclareRobustCommand\boldmath{\@nomath\boldmath
              \mathversion{bold}}
\DeclareRobustCommand\unboldmath{\@nomath\unboldmath
              \mathversion{normal}}
```

Here we switch to the default math version by defining the internal macro `\math@version`. We dare not to call `\mathversion` at this place because this would call `\glb@settings`.

```
\def\math@version{normal}
```

3.1 Legacy

We start by defining a few macros that are part of standard LaTeX's user interface. The use of these functions is not encouraged, but they will allow to process older documents without changes to the source.

\newfont

```
\def\newfont#1#2{\@ifdefinable#1{\font#1=#2\relax}}
```

(*End of definition for* \newfont.)

\symbol

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\symbol}{XeTeX change for math}%
\ifdefined\XeTeXversion
  \DeclareRobustCommand\symbol[1]{\Ucharcat#1 12\relax}
\else
  \DeclareRobustCommand\symbol[1]{\char#1\relax}
\fi
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\symbol}{XeTeX change for math}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\symbol[1]{\char#1\relax}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \symbol.)

3.2 Miscellaneous

\@setfontsize
\@setsize

This abbreviation is used by LaTeX's user level size changing commands, such as \large.

```
\def\@setfontsize#1#2#3{\@nomath#1%
```

For the benefit of people relying on keeping the name of the current font command saved in \@currsize we define it. To ensure that \@setfontsize keeps being robust we omit this assignment during times where \protect differs from \@typeset@protect.

```
    \ifx\protect\@typeset@protect
      \let\@currsize#1%
    \fi
    \fontsize{#2}{#3}\selectfont}
```

For compatibility we also define \@setsize the 209 command

```
⟨*compat⟩
\def\@setsize#1#2#3#4{\@setfontsize#1{#4}{#2}}
⟨/compat⟩
```

(*End of definition for* \@setfontsize *and* \@setsize.)

\hexnumber@ To set up LaTeX's special math character definitions we first provide a macro to generate hexadecimal numbers. It is a rather simple \ifcase.

```
\def\hexnumber@#1{\ifcase\number#1
 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or
 9\or A\or B\or C\or D\or E\or F\fi}
```

(*End of definition for* \hexnumber@.)

\nfss@text In it simplest form \nfss@text is an \mbox. This will produce unbreakable text outside math and inside math you will get text with the same fonts as outside. The only drawback is that such item won't change sizes in subscripts. But this behavior can be easily changed. With the amstex style option one will get a sub style called amstext which will redefine the \nfss@text macro to produce correct text in all sizes.

We have to use \def instead of the shorter \let since \mbox is undefined when we reach this point.

```
\def\nfss@text#1{{\mbox{#1}}}
```

(*End of definition for* \nfss@text.)

\copyright The definition of \copyright was changed so that it works in other type styles, and to make it robust. We leave the family untouched so that the copyright notice will come out differently if a different font family is in use. This command is commented out, since it is now defined in ltoutenc.dtx.

```
%\DeclareRobustCommand\copyright
%    {{\ooalign{\hfil
%     \raise.07ex\hbox{\mdseries\upshape c}\hfil\crcr
%     \mathhexbox20D}}}
```

(*End of definition for* \copyright.)

\normalfont \reset@font The macro \reset@font is used in LaTeX to switch to a standard font, in order to initialize the current font in situations where typesetting is done in a new visual context (e.g. in a footnote). We define it here to allow the test for the new LaTeX version above but nevertheless are able to run all kind of mixtures.

The user interface name for \reset@font is \normalfont:

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                    {\normalfont}{Add hook to \normalfont}%
\DeclareRobustCommand\normalfont{%
```

Instead of calling \usefont, as it was done in the past, we inline the code from \usefont as we want to add the hook before \selectfont, but after all the font attributes are set.

```
    \fontencoding\encodingdefault
    \edef\f@family{\familydefault}%
    \edef\f@series{\seriesdefault}%
    \edef\f@shape{\shapedefault}%
```

Any earlier \fontseries, etc. should be canceled and we should switch unconditionally to the requested font face so we drop any code that may have been stored in \delayed@f@adjustment.

```
    \let\delayed@f@adjustment\@empty

    \UseHook{normalfont}%
```

This is the old name for the hook introduced in 2020/02/02. It will be removed in one of the future releases!

```
   \@defaultfamilyhook        % hookname from 2020/02 will vanish
   \selectfont}
\let\reset@font\normalfont
```

(*End of definition for* `\normalfont` *and* `\reset@font`.)

```
% \changes{v3.2g}{2021/03/18}
%         {Add missing 2020/02/02 latexrelease entry.}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                 {\normalfont}{Add hook to \normalfont}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\normalfont{%
⟨latexrelease⟩   \fontencoding\encodingdefault
⟨latexrelease⟩   \edef\f@family{\familydefault}%
⟨latexrelease⟩   \edef\f@series{\seriesdefault}%
⟨latexrelease⟩   \edef\f@shape{\shapedefault}%
⟨latexrelease⟩   \UseHook{normalfont}%
⟨latexrelease⟩   \@defaultfamilyhook        % hookname from 2020/02 will vanish
⟨latexrelease⟩   \selectfont}
⟨latexrelease⟩
⟨latexrelease⟩\let\reset@font\normalfont
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\normalfont}{Add hook to \normalfont}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\normalfont{%
⟨latexrelease⟩   \fontencoding\encodingdefault
⟨latexrelease⟩   \edef\f@family{\familydefault}%
⟨latexrelease⟩   \edef\f@series{\seriesdefault}%
⟨latexrelease⟩   \edef\f@shape{\shapedefault}%
⟨latexrelease⟩   \@defaultfamilyhook
⟨latexrelease⟩   \selectfont}
⟨latexrelease⟩
⟨latexrelease⟩\let\reset@font\normalfont
⟨latexrelease⟩
⟨latexrelease⟩\let\@defaultfamilyhook\@empty
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\normalfont}{Add hook to \normalfont}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\normalfont
⟨latexrelease⟩         {\usefont\encodingdefault
⟨latexrelease⟩                  \familydefault
⟨latexrelease⟩                  \seriesdefault
⟨latexrelease⟩                  \shapedefault
```

```
⟨latexrelease⟩                    \relax}
⟨latexrelease⟩\let\reset@font\normalfont
⟨latexrelease⟩
⟨latexrelease⟩\let\@defaultfamilyhook\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

We left out the special LaTeX fonts which are not automatically included in the base version of the font selection since these fonts contain only a few characters which are also included in the AMS fonts so anybody who is using these fonts doesn't need them. But for compatibility reasons we will define these symbols.

```
\def\not@base#1{\@latex@error
  {Command \noexpand#1not provided in base LaTeX2e}%
  {Load the latexsym or the amsfonts package to
   define this symbol}}
\def\mho{\not@base\mho}
\def\Join{\not@base\Join}
\def\Box{\not@base\Box}
\def\Diamond{\not@base\Diamond}
\def\leadsto{\not@base\leadsto}
\def\sqsubset{\not@base\sqsubset}
\def\sqsupset{\not@base\sqsupset}
\def\lhd{\not@base\lhd}
\def\unlhd{\not@base\unlhd}
\def\rhd{\not@base\rhd}
\def\unrhd{\not@base\unrhd}
```

We now initialize all variables set by `\DeclareErrorFont`. These values are not really important since they will be overwritten later on by the definition in `fontdef.ltx`.

However, if `fontdef.cfg` is corrupted then at least a hopefully suitable error font is present.

```
\DeclareErrorFont{OT1}{cmr}{m}{n}{10}  %% don't modify this setting
                                       %% overwrite it in fontdef.cfg
                                       %% if necessary
```

We also set some default values for `\f@family` etc. Note that we don't yet have any encodings that comes later. In the past this was implicitly done by `\DeclareErrorFont`.

```
\fontfamily{cmr}
```

Previously the default values for series and shape were set by calling `\fontseries` and `\fontshape`, but their action is now delayed until `\selectfont` which isn't called inside the format (to avoid unnecessarily loading a font that may never get used). We therefore have to set `\f@series` and `\f@shape` directly instead.

```
\def\f@series{m}           % \fontseries{m}
\def\f@shape{n}            % \fontshape{n}
```

```
\fontsize{10}{10}
```

The initial `fontenc` package load list. This will get overwritten in `fonttext` and is only provided in case an old `fonttext.cfg` does not define the command:

```
\def\@fontenc@load@list{\@elt{T1,OT1}}
```

We now load the customizable parts of NFSS.

```
\InputIfFileExists{fonttext.cfg}
           {\typeout{=====================================^^J%
                     ^^J%
                      Local config file fonttext.cfg used^^J%
                     ^^J%
                     =====================================}%
             \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
            }
           {\input{fonttext.ltx}}
\let\@addtofilelist\@gobble
```

Ditto for math although I don't think that we will get a lot of customisation :-)

```
\InputIfFileExists{fontmath.cfg}
           {\typeout{=====================================^^J%
                     ^^J%
                      Local config file fontmath.cfg used^^J%
                     ^^J%
                     =====================================}%
             \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
            }
           {\input{fontmath.ltx}}
\let\@addtofilelist\@gobble
```

Then we preload several fonts. This file might be customized *without* changing the behavior of the format (i.e. necessary font definitions will be loaded at runtime if they are not preloaded). This is done in the file `preload.ltx`.

```
\InputIfFileExists{preload.cfg}
           {\typeout{=====================================^^J%
                     ^^J%
                      Local config file preload.cfg used^^J%
                     ^^J%
                     =====================================}%
             \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
            }
           {\input{preload.ltx}}
\let\@addtofilelist\@gobble
```

`\seriesdefault`
`\seriesdefault@kernel`

After `\seriesdefault` got defined inside `fonttext.ltx` or a `.cfg` file overwriting it, we alter its value by appending `\@empty` to it. This will vanish if expanded but allows us to check if the default gets altered (even to the same value) in the document preamble. All we have to do is to save the current value somewhere and later compare the two. For this we use `\seriesdefault@kernel`.

```
\expandafter\def\expandafter\seriesdefault\expandafter{\seriesdefault\@empty}
\let\seriesdefault@kernel\seriesdefault
```

(*End of definition for* `\seriesdefault` *and* `\seriesdefault@kernel`.)

`\@acci`
`\@accii`
`\@acciii`

We also save the values of some accents in `\@acci`, `\@accii` and `\@acciii` so they can be restored by a `minipage` inside a `tabbing` environment.

```
\let\@acci\' \let\@accii\` \let\@acciii\=
```

(*End of definition for* `\@acci`, `\@accii`, *and* `\@acciii`.)

`\cal`
`\mit`

Here were the two old ⟨*alphabet identifiers*⟩.

(*End of definition for* `\cal` *and* `\mit`.)

⟨/2ekernel⟩

File 30
fontdef.dtx

1 Introduction

This file is used to generate the files `fonttext.ltx` (text font declarations) and `fontmath.ltx` (math font declarations), which are used during the format generation. It contains the declaration of the standard text encodings used at the site as well as a minimal subset of font shape groups that NFSS will look at to ensure that the specified encodings are valid.

The math part contains the setup for math encodings as well as the default math symbol declarations that belong to the encoding.

It is possible to change this setup (by using other fonts, or defaults) without losing the ability to process documents written at other sites. Portability in this sense means that a document will compile without errors. It does not mean, however, that identical output will be produced. For this it is necessary that the distributed setup is used at both installations.

2 Customization

You are not allowed to change this source file! If you want to change the default encodings and/or the font shape groups preloaded you should create a copy of `fonttext.ltx` under the name `fonttext.cfg` and change this copy. If LaTeX 2_ε finds a file of this name it will use it, otherwise it uses the standard file which is `fontdef.ltx`.

If you don't plan to use Computer Modern much or at all, it might (!) be a good idea to make your own `fonttext.cfg`. Look at the comments below (docstrip module 'text') to see what should go into such a file.

To change the math font setup use a copy of `fontmath.ltx` under the name `fontmath.cfg` and change this copy. However, dealing with this interface is even more a job for an expert than changing the text font setup — in short, we don't encourage either.

> **Warning:** please note that we don't support customised LaTeX versions. Thus, before sending in a bug report please try your test file with a LaTeX format which is not customised and send in the log from that version (unless the problem goes away).

Please note: the following standard encodings have to be defined in all local variants of `font....cfg` to guarantee that all LaTeX installations behave in the same way.

`T1`	Cork TeX text encoding
`OT1`	old TeX text encoding
`U`	unknown encoding
`OML`	old TeX math letters encoding
`OMS`	old TeX math symbols encoding
`OMX`	old TeX math extension symbols encoding
`TU`	Unicode

Notice that some of these encodings are 'old' in the sense that we hope that they will be superseded soon by encoding standards defined by the TEX user community. Therefore this set of default encodings may change in the future.

The first candidate is `OT1` which will soon be replaced by `T1`, the official TEX text encoding.

> **Warning:** If you add additional encodings to this file there is no guarantee any longer that files processable at your installation will also be processable at other installations. Thus, if you make use of such an encoding in your document, e.g. if you intend to typeset in Cyrillic (`OT2` encoding), you need to specify this encoding in the preamble of your document prior to sending it to another installation. Once the encoding is specified in that place in your document, the document is processable at all LATEX installations (provided they have suitable fonts installed).
>
> For this reason we suggest that you define a short package file that sets up an additional encoding used at your site (rather than putting the encoding into this file) since this package can easily be shipped with your document.

3 The `docstrip` modules

The following modules are used to direct `docstrip` in generating external files:

driver	produce a documentation driver file
text	produce the file `fonttext.ltx`
math	produce the file `fontmath.ltx`
cfgtext	produce a dummy `fonttext.cfg` file
cfgmath	produce a dummy `fontmath.cfg` file

A typical `docstrip` command file would then have entries like:

```
generateFile{fonttext.ltx}{t}{\from{fontdef.dtx}{text}}
```

4 A driver for this document

The next bit of code contains the documentation driver file for TEX, i.e. the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```
⟨*driver⟩
\documentclass{ltxdoc}
\GetFileInfo{fontdef.dtx}
\begin{document}
   \DocInput{fontdef.dtx}
\end{document}
⟨/driver⟩
```

5 The `fonttext.ltx` file

The identification is done earlier on with a `\ProvidesFile` declaration.

```
⟨*text⟩
\typeout{=== Don't modify this file, use a .cfg file instead ===^^J}
```

5.1 Encodings

This file declares the standard encodings for text and math fonts. All others should be declared in packages or in the documents directly.

For every text encoding there are normally a number of encoding specific commands, e.g. accents, special characters, etc. (The definition for such a command might have to change when the encoding is changed, because the character is in a different position, or not available at all, or the accent is produced in a different way.) This is handled by a general mechanism which is described in `ltoutenc.dtx`.

By convention, text encoding specific declarations, including the `\DeclareFontEncoding` declaration, are kept in separate file of the form ⟨*enc*⟩`enc.def`, e.g. `ot1enc.def`. This allows other applications to make use of the declarations as well.

Similar to the default encoding, the loading of the encoding files for the two major text encodings shouldn't be changed. In particular, the `inputenc` package depends on this.

```
\input {omlenc.def}
\input {omsenc.def}
```

Documents containing a lot of accented characters should really be using T1 fonts. We therefore load this last so that T1 encoding specific commands are executed as fast as possible (encoding files are no longer reloaded in `fontenc`.

```
\input {ot1enc.def}
\input  {t1enc.def}
\input{ts1enc.def}
```

```
\ifx\Umathcode\@undefined
```

We then set the default text font encoding. This will hopefully change some day to `T1`. This setting should *not* be changed to produce a portable format.

```
\fontencoding{OT1}
```

The initial `fontenc` package load list if an 8-bit TEX engine is used:

```
\def\@fontenc@load@list{\@elt{T1,OT1}}
```

```
\def\rmsubstdefault{cmr}
\def\sfsubstdefault{cmss}
\def\ttsubstdefault{cmtt}
\LoadFontDefinitionFile{TS1}{cmr}
```

```
\else
```

Unicode.

```
\input {tuenc.def}
\fontencoding{TU}
```

The initial `fontenc` package load list if a Unicode engine is used:

```
\def\@fontenc@load@list{\@elt{TU}}
```

```
\DeclareFontSubstitution{TU}{lmr}{m}{n}
\LoadFontDefinitionFile{TU}{lmr}
\LoadFontDefinitionFile{TU}{lmss}
\LoadFontDefinitionFile{TU}{lmtt}
```

```
\def\rmsubstdefault{lmr}
\def\sfsubstdefault{lmss}
\def\ttsubstdefault{lmtt}
\LoadFontDefinitionFile{TS1}{lmr}
```

```
\DeclareFontSubstitution{TU}{lmr}{m}{n}
```

End of Unicode branch.

```
\fi
```

If different encodings for text fonts are in use one could put the common setup into `\DeclareFontEncodingDefaults`. There is now a better mechanism so using this interface is discouraged!

```
\DeclareFontEncodingDefaults{}{}
```

Then we define the default substitution for every encoding. This release of LaTeX 2_ε assumes that the ec fonts are available. It is possible to change this to point to some other font family (e.g., Times with the appropriate encoding if it is available) without making documents non-portable. However, in such a case documents will produce different page breaks at other sites. The substitution defaults can all be changed without losing portability as long as there are font shape definitions for the selected substitutions.

```
\DeclareFontSubstitution{T1}{cmr}{m}{n}
\DeclareFontSubstitution{OT1}{cmr}{m}{n}
```

For every encoding declaration, LaTeX 2_ε will try to verify that the given substitution information makes sense, i.e. that it is impossible to go into an endless loop if font substitution happens. This is done at the moment the `\begin{document}` is encountered. LaTeX 2_ε will then check that for every encoding the substitution defaults form a valid font shape group, which means that it will check if there is a `\DeclareFontShape` declaration for this combination. We will therefore load the corresponding `.fd` files now. If we don't do this they would be loaded at verification time (i.e. at `\begin{document}` which would delay processing unnecessarily.

> **Warning:** Please note that this means that you have to regenerate the format whenever you change any of these `.fd` files since LaTeX 2_ε will not read `.fd` files if it already knows about the encoding/family combination.

The `\nfss@catcodes` ensures that white space is ignored in any definitions made in the fd files.

```
\begingroup
\nfss@catcodes
\input  {t1cmr.fd}
\input  {ot1cmr.fd}
\endgroup
```

We also load some other font definition files which are normally needed in a document. This is only done for processing speed and you can comment the next two lines out to save some memory. If necessary these files are then loaded when your document is processed. (Loading `.fd` files is a less drastic step compared to preloading fonts because the number of fonts is limited 255 at (nearly) every TeX installation, while the amount of main memory is not a limiting factor at most installations.)

```
\begingroup
\nfss@catcodes
\input {t1cmss.fd}
\input {t1cmtt.fd}
\input {ot1cmss.fd}
\input {ot1cmtt.fd}
\endgroup
```

Even though Unicode engines default to `lm` load `ts1cmr` as this may be used for fallback for TS1 encoding.

```
\ifx\Umathcode\@undefined\else
\begingroup
\nfss@catcodes
\input {ts1cmr.fd}
\endgroup
\fi
```

Even with all the precautions it is still possible that NFSS will run into problems, for example, when a `.fd` file contains corrupted data. To guard against such cases NFSS has a very low-level fallback font that is installed with the following line.

```
\DeclareErrorFont{OT1}{cmr}{m}{n}{10}
```

This means, "if everything else fails use Computer Modern Roman normal shape at 10pt in the old text encoding". You can change the font used but the encoding should be the same as the one specified with `\fontencoding` above.

5.2 Defaults

To allow the use of `\rmfamily`, `\sffamily`, etc. in documents even if non-standard families are used we provide nine macros which hold the name of the corresponding families, series, and so on. This makes it easy to use other font families (like Times Roman, etc.). One simply has to redefine these defaults.

All these hooks have to be defined in this file but you can change their meaning (except for `\encodingdefault`) without making documents non-portable.

`\encodingdefault`
`\rmdefault`
`\sfdefault`
`\ttdefault`

The following three definitions set up the meaning for `\rmfamily`, `\sffamily`, and `\ttfamily`.

```
\ifx\Umathcode\@undefined
\newcommand\encodingdefault{OT1}
\newcommand\rmdefault{cmr}
\newcommand\sfdefault{cmss}
\newcommand\ttdefault{cmtt}
\else
\newcommand\encodingdefault{TU}
\newcommand\rmdefault{lmr}
\fontfamily{\rmdefault}
\newcommand\sfdefault{lmss}
\newcommand\ttdefault{lmtt}
\fi
⟨/text⟩
⟨latexrelease⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease⟩                  {\encodingdefault}{TU encoding default}%
⟨latexrelease⟩\ifx\Umathcode\@undefined
⟨latexrelease⟩\renewcommand\encodingdefault{OT1}
⟨latexrelease⟩\fontencoding{\encodingdefault}
⟨latexrelease⟩\renewcommand\rmdefault{cmr}
⟨latexrelease⟩\fontfamily{\rmdefault}
⟨latexrelease⟩\renewcommand\sfdefault{cmss}
⟨latexrelease⟩\renewcommand\ttdefault{cmtt}
⟨latexrelease⟩\else
⟨latexrelease⟩\renewcommand\encodingdefault{TU}
```

```
⟨latexrelease⟩%done in everyjob\fontencoding{\encodingdefault}
⟨latexrelease⟩\renewcommand\rmdefault{lmr}
⟨latexrelease⟩\fontfamily{\rmdefault}
⟨latexrelease⟩\renewcommand\sfdefault{lmss}
⟨latexrelease⟩\renewcommand\ttdefault{lmtt}
⟨latexrelease⟩\fi
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\encodingdefault}{TU encoding default}%
⟨latexrelease⟩\fontencoding{OT1}
⟨latexrelease⟩\renewcommand\encodingdefault{OT1}
⟨latexrelease⟩\fontencoding{\encodingdefault}
⟨latexrelease⟩\renewcommand\rmdefault{cmr}
⟨latexrelease⟩\fontfamily{\rmdefault}
⟨latexrelease⟩\renewcommand\sfdefault{cmss}
⟨latexrelease⟩\renewcommand\ttdefault{cmtt}
⟨latexrelease⟩\EndIncludeInRelease
⟨*text⟩
```

(*End of definition for* `\encodingdefault` *and others.*)

`\bfdefault`
`\mddefault`

Series changing commands are influenced by the following hooks.

```
\newcommand\bfdefault{b}  % overwritten below (for rollback)
\newcommand\mddefault{m}  % overwritten below (for rollback)
```

(*End of definition for* `\bfdefault` *and* `\mddefault`.)

`\itdefault`
`\sldefault`
`\scdefault`
`\updefault`

Shape changing commands use the following hooks.

```
\newcommand\itdefault{it}
\newcommand\sldefault{sl}
\newcommand\scdefault{sc}
\newcommand\updefault{up}  % overwritten below (for rollback)
```

(*End of definition for* `\itdefault` *and others.*)

```
⟨/text⟩
⟨*text | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\updefault}{font defaults change}%
\renewcommand\updefault{up}
```

We append `\@empty` to the series value so that we can detect if it got changed via `\def` or `\renewcommand` later.

```
\renewcommand\bfdefault{b\@empty}
\renewcommand\mddefault{m\@empty}
```

```
\let\bfdefault@previous\bfdefault
\let\mddefault@previous\mddefault
⟨/text | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\updefault}{font defaults change}%
⟨latexrelease⟩
⟨latexrelease⟩\renewcommand\updefault{n}
⟨latexrelease⟩\renewcommand\bfdefault{bx}
⟨latexrelease⟩
```

```
⟨latexrelease⟩\let\bfdefault@previous\undefined
⟨latexrelease⟩\let\mddefault@previous\undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*text⟩
```

\familydefault
\seriesdefault
\shapedefault

Finally we have the hooks that describe the behaviour of the `\normalfont` command. To stay portable, the definition of `\encodingdefault` should *not* be changed and should match the setting above for `\fontencoding`. All other values can be set according to your taste.

```
\newcommand\familydefault{\rmdefault}
\newcommand\seriesdefault{\mddefault}
```

In previous releases `\shapedefault` pointed to `\updefault` which resolved to `n`, but these days that is no longer the case (and `up` is wrong when you want to do a reset. So we now use `n` explicitly.

```
\newcommand\shapedefault{n}
```

(*End of definition for* `\familydefault`*,* `\seriesdefault`*, and* `\shapedefault`*.*)

This finishes the low-level setup in `fonttext.ltx`.

```
⟨/text⟩
```

6 The `fontmath.ltx` file

The identification is done earlier on with a `\ProvidesFile` declaration.

```
⟨*math⟩
\typeout{=== Don't modify this file, use a .cfg file instead ===^^J}
```

6.1 The font encodings used

```
\DeclareFontEncoding{OML}{}{}
\DeclareFontEncoding{OMS}{}{}
\DeclareFontEncoding{OMX}{}{}
```

Finally a declaration for `U` encoding which serves for all fonts that do not fit standard encodings. For math this sets up `\noaccents@` providing for AMS-LaTeX. This macro is used therein to handle accented characters if they are not supported by the font. In other words, if fonts with `U` encoding are used in math, all accents (like from `\breve`) are obtained from some other font that has them.

```
\DeclareFontEncoding{U}{}{\noaccents@}
```

The encodings for math are next:

```
\DeclareFontSubstitution{OML}{cmm}{m}{it}
\DeclareFontSubstitution{OMS}{cmsy}{m}{n}
\DeclareFontSubstitution{OMX}{cmex}{m}{n}
\DeclareFontSubstitution{U}{cmr}{m}{n}
```

```
\begingroup
\nfss@catcodes
\input  {omlcmm.fd}
\input  {omscmsy.fd}
\input  {omxcmex.fd}
\input  {ucmr.fd}
\endgroup
```

6.1.1 Symbolfont and Alphabet declarations

We now define the basic symbol fonts used by LaTeX. These four symbol fonts must be defined by this file.

It is possible to make the symbol fonts point to other external fonts without losing the ability to process documents written at other sites, as long as one defines the same symbol font names with the same encodings, e.g. `operators` with `OT1` etc. If other encodings are used documents become non-portable. Such a change should therefore be done in a package file.

```
\DeclareSymbolFont{operators}   {OT1}{cmr} {m}{n}
\DeclareSymbolFont{letters}     {OML}{cmm} {m}{it}
\DeclareSymbolFont{symbols}     {OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
```

```
\SetSymbolFont{operators}{bold}{OT1}{cmr} {bx}{n}
\SetSymbolFont{letters}  {bold}{OML}{cmm} {b}{it}
\SetSymbolFont{symbols}  {bold}{OMS}{cmsy}{b}{n}
```

Below are the seven math alphabets which are defined by NFSS. Again they must be defined by this file. However, as before you can change the fonts used without losing portability, but you should be careful when changing the encoding since that may make documents come out wrong.

```
\DeclareSymbolFontAlphabet{\mathrm}    {operators}
\DeclareSymbolFontAlphabet{\mathnormal}{letters}
\DeclareSymbolFontAlphabet{\mathcal}   {symbols}
\DeclareMathAlphabet      {\mathbf}{OT1}{cmr}{bx}{n}
\DeclareMathAlphabet      {\mathsf}{OT1}{cmss}{m}{n}
\DeclareMathAlphabet      {\mathit}{OT1}{cmr}{m}{it}
\DeclareMathAlphabet      {\mathtt}{OT1}{cmtt}{m}{n}
```

Given the currently available fonts we cannot bold-en `\mathbf` and `\mathtt` but in principle one could use 'ultra bold' or something. The alphabets defined via `\DeclareSymbolFontAlphabet` will change automatically in a new math version if the corresponding symbol font changes.

```
\SetMathAlphabet\mathsf{bold}{OT1}{cmss}{bx}{n}
\SetMathAlphabet\mathit{bold}{OT1}{cmr}{bx}{it}
```

6.2 Math font sizes

The declarations below declare the text, script and scriptscript size to be used for each text font size.

All occurrences of sizes longer than a single character are replaced with the macro name that holds them, saving a number of tokens (but losing a bit of speed, so this may not stay this way).

```
\DeclareMathSizes{5}{5}{5}{5}
\DeclareMathSizes{6}{6}{5}{5}
\DeclareMathSizes{7}{7}{5}{5}
\DeclareMathSizes{8}{8}{6}{5}
\DeclareMathSizes{9}{9}{6}{5}
\DeclareMathSizes{\@xpt}{\@xpt}{7}{5}
\DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
\DeclareMathSizes{\@xiipt}{\@xiipt}{8}{6}
\DeclareMathSizes{\@xivpt}{\@xivpt}{\@xpt}{7}
```

```
\DeclareMathSizes{\@xviipt}{\@xviipt}{\@xiipt}{\@xpt}
\DeclareMathSizes{\@xxpt}{\@xxpt}{\@xivpt}{\@xiipt}
\DeclareMathSizes{\@xxvpt}{\@xxvpt}{\@xxpt}{\@xviipt}
```

6.3 The math symbol assignments

We start by setting up math codes for most of the characters typed in directly from the keyboard. Most of them are normally already setup up in the same way by IniTEX. However, we repeat them here to have a complete setup which can be exchanged with another if desired.

6.3.1 The letters

```
\DeclareMathSymbol{a}{\mathalpha}{letters}{`a}
\DeclareMathSymbol{b}{\mathalpha}{letters}{`b}
\DeclareMathSymbol{c}{\mathalpha}{letters}{`c}
\DeclareMathSymbol{d}{\mathalpha}{letters}{`d}
\DeclareMathSymbol{e}{\mathalpha}{letters}{`e}
\DeclareMathSymbol{f}{\mathalpha}{letters}{`f}
\DeclareMathSymbol{g}{\mathalpha}{letters}{`g}
\DeclareMathSymbol{h}{\mathalpha}{letters}{`h}
\DeclareMathSymbol{i}{\mathalpha}{letters}{`i}
\DeclareMathSymbol{j}{\mathalpha}{letters}{`j}
\DeclareMathSymbol{k}{\mathalpha}{letters}{`k}
\DeclareMathSymbol{l}{\mathalpha}{letters}{`l}
\DeclareMathSymbol{m}{\mathalpha}{letters}{`m}
\DeclareMathSymbol{n}{\mathalpha}{letters}{`n}
\DeclareMathSymbol{o}{\mathalpha}{letters}{`o}
\DeclareMathSymbol{p}{\mathalpha}{letters}{`p}
\DeclareMathSymbol{q}{\mathalpha}{letters}{`q}
\DeclareMathSymbol{r}{\mathalpha}{letters}{`r}
\DeclareMathSymbol{s}{\mathalpha}{letters}{`s}
\DeclareMathSymbol{t}{\mathalpha}{letters}{`t}
\DeclareMathSymbol{u}{\mathalpha}{letters}{`u}
\DeclareMathSymbol{v}{\mathalpha}{letters}{`v}
\DeclareMathSymbol{w}{\mathalpha}{letters}{`w}
\DeclareMathSymbol{x}{\mathalpha}{letters}{`x}
\DeclareMathSymbol{y}{\mathalpha}{letters}{`y}
\DeclareMathSymbol{z}{\mathalpha}{letters}{`z}
```

```
\DeclareMathSymbol{A}{\mathalpha}{letters}{`A}
\DeclareMathSymbol{B}{\mathalpha}{letters}{`B}
\DeclareMathSymbol{C}{\mathalpha}{letters}{`C}
\DeclareMathSymbol{D}{\mathalpha}{letters}{`D}
\DeclareMathSymbol{E}{\mathalpha}{letters}{`E}
\DeclareMathSymbol{F}{\mathalpha}{letters}{`F}
\DeclareMathSymbol{G}{\mathalpha}{letters}{`G}
\DeclareMathSymbol{H}{\mathalpha}{letters}{`H}
\DeclareMathSymbol{I}{\mathalpha}{letters}{`I}
\DeclareMathSymbol{J}{\mathalpha}{letters}{`J}
\DeclareMathSymbol{K}{\mathalpha}{letters}{`K}
\DeclareMathSymbol{L}{\mathalpha}{letters}{`L}
\DeclareMathSymbol{M}{\mathalpha}{letters}{`M}
\DeclareMathSymbol{N}{\mathalpha}{letters}{`N}
```

```
\DeclareMathSymbol{O}{\mathalpha}{letters}{`O}
\DeclareMathSymbol{P}{\mathalpha}{letters}{`P}
\DeclareMathSymbol{Q}{\mathalpha}{letters}{`Q}
\DeclareMathSymbol{R}{\mathalpha}{letters}{`R}
\DeclareMathSymbol{S}{\mathalpha}{letters}{`S}
\DeclareMathSymbol{T}{\mathalpha}{letters}{`T}
\DeclareMathSymbol{U}{\mathalpha}{letters}{`U}
\DeclareMathSymbol{V}{\mathalpha}{letters}{`V}
\DeclareMathSymbol{W}{\mathalpha}{letters}{`W}
\DeclareMathSymbol{X}{\mathalpha}{letters}{`X}
\DeclareMathSymbol{Y}{\mathalpha}{letters}{`Y}
\DeclareMathSymbol{Z}{\mathalpha}{letters}{`Z}
```

6.3.2 The digits

```
\DeclareMathSymbol{0}{\mathalpha}{operators}{`0}
\DeclareMathSymbol{1}{\mathalpha}{operators}{`1}
\DeclareMathSymbol{2}{\mathalpha}{operators}{`2}
\DeclareMathSymbol{3}{\mathalpha}{operators}{`3}
\DeclareMathSymbol{4}{\mathalpha}{operators}{`4}
\DeclareMathSymbol{5}{\mathalpha}{operators}{`5}
\DeclareMathSymbol{6}{\mathalpha}{operators}{`6}
\DeclareMathSymbol{7}{\mathalpha}{operators}{`7}
\DeclareMathSymbol{8}{\mathalpha}{operators}{`8}
\DeclareMathSymbol{9}{\mathalpha}{operators}{`9}
```

6.3.3 Punctuation, brace, etc. keys

```
\DeclareMathSymbol{!}{\mathclose}{operators}{"21}
\DeclareMathSymbol{*}{\mathbin}{symbols}{"03} % \ast
\DeclareMathSymbol{+}{\mathbin}{operators}{"2B}
\DeclareMathSymbol{,}{\mathpunct}{letters}{"3B}
\DeclareMathSymbol{-}{\mathbin}{symbols}{"00}
\DeclareMathSymbol{.}{\mathord}{letters}{"3A}
\DeclareMathSymbol{:}{\mathrel}{operators}{"3A}
\DeclareMathSymbol{;}{\mathpunct}{operators}{"3B}
\DeclareMathSymbol{=}{\mathrel}{operators}{"3D}
\DeclareMathSymbol{?}{\mathclose}{operators}{"3F}
```

The following symbols are defined as delimiters below which automatically defines them as math symbols.

```
%\DeclareMathSymbol{(}{\mathopen}{operators}{"28}
%\DeclareMathSymbol{)}{\mathclose}{operators}{"29}
%\DeclareMathSymbol{/}{\mathord}{letters}{"3D}
%\DeclareMathSymbol{[}{\mathopen}{operators}{"5B}
%\DeclareMathSymbol{]}{\mathclose}{operators}{"5D}
%\DeclareMathSymbol{|}{\mathord}{symbols}{"6A}
%\DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
%\DeclareMathSymbol{>}{\mathrel}{letters}{"3E}
```

Should all of the following being activated by default? Probably not.

```
%\DeclareMathSymbol{`\{}{\mathopen}{symbols}{"66}
%\DeclareMathSymbol{`\}}{\mathclose}{symbols}{"67}
%\DeclareMathSymbol{`\\}{\mathord}{symbols}{"6E} % \backslash
\mathcode`\ ="8000 % \space
\mathcode`\'="8000 % ^\prime
\mathcode`\_="8000 % \_
```

6.3.4 Delimitercodes for characters

[to be completed]

Finally, IniTEX sets all `\delcode` values to -1, except `\delcode‘.=0`

```
\DeclareMathDelimiter{(}{\mathopen} {operators}{"28}{largesymbols}{"00}
\DeclareMathDelimiter{)}{\mathclose}{operators}{"29}{largesymbols}{"01}
\DeclareMathDelimiter{[}{\mathopen} {operators}{"5B}{largesymbols}{"02}
\DeclareMathDelimiter{]}{\mathclose}{operators}{"5D}{largesymbols}{"03}
```

The next two are considered to be relations when not used in the context of a delimiter! And worse, they do even represent different glyphs when being used as delimiter and not as delimiter. This is a user level syntax inherited from plain TEX. Therefore we explicitly redefine the math symbol definitions for these symbols afterwards.

```
\DeclareMathDelimiter{<}{\mathopen}{symbols}{"68}{largesymbols}{"0A}
\DeclareMathDelimiter{>}{\mathclose}{symbols}{"69}{largesymbols}{"0B}
\DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
\DeclareMathSymbol{>}{\mathrel}{letters}{"3E}
```

And here is another case where the non-delimiter version produces a glyph different from the delimiter version.

```
\DeclareMathDelimiter{/}{\mathord}{operators}{"2F}{largesymbols}{"0E}
\DeclareMathSymbol{/}{\mathord}{letters}{"3D}
```

```
\DeclareMathDelimiter{|}{\mathord}{symbols}{"6A}{largesymbols}{"0C}
```

```
\expandafter\DeclareMathDelimiter\@backslashchar
                        {\mathord}{symbols}{"6E}{largesymbols}{"0F}
```

N.B. `{` and `}` should NOT get delcodes; otherwise parameter grouping fails!

6.4 Symbols accessed via control sequences

6.4.1 Greek letters

```
\DeclareMathSymbol{\alpha}{\mathord}{letters}{"0B}
\DeclareMathSymbol{\beta}{\mathord}{letters}{"0C}
\DeclareMathSymbol{\gamma}{\mathord}{letters}{"0D}
\DeclareMathSymbol{\delta}{\mathord}{letters}{"0E}
\DeclareMathSymbol{\epsilon}{\mathord}{letters}{"0F}
\DeclareMathSymbol{\zeta}{\mathord}{letters}{"10}
\DeclareMathSymbol{\eta}{\mathord}{letters}{"11}
\DeclareMathSymbol{\theta}{\mathord}{letters}{"12}
\DeclareMathSymbol{\iota}{\mathord}{letters}{"13}
\DeclareMathSymbol{\kappa}{\mathord}{letters}{"14}
\DeclareMathSymbol{\lambda}{\mathord}{letters}{"15}
\DeclareMathSymbol{\mu}{\mathord}{letters}{"16}
\DeclareMathSymbol{\nu}{\mathord}{letters}{"17}
\DeclareMathSymbol{\xi}{\mathord}{letters}{"18}
\DeclareMathSymbol{\pi}{\mathord}{letters}{"19}
\DeclareMathSymbol{\rho}{\mathord}{letters}{"1A}
\DeclareMathSymbol{\sigma}{\mathord}{letters}{"1B}
\DeclareMathSymbol{\tau}{\mathord}{letters}{"1C}
\DeclareMathSymbol{\upsilon}{\mathord}{letters}{"1D}
\DeclareMathSymbol{\phi}{\mathord}{letters}{"1E}
\DeclareMathSymbol{\chi}{\mathord}{letters}{"1F}
\DeclareMathSymbol{\psi}{\mathord}{letters}{"20}
```

```
\DeclareMathSymbol{\omega}{\mathord}{letters}{"21}
\DeclareMathSymbol{\varepsilon}{\mathord}{letters}{"22}
\DeclareMathSymbol{\vartheta}{\mathord}{letters}{"23}
\DeclareMathSymbol{\varpi}{\mathord}{letters}{"24}
\DeclareMathSymbol{\varrho}{\mathord}{letters}{"25}
\DeclareMathSymbol{\varsigma}{\mathord}{letters}{"26}
\DeclareMathSymbol{\varphi}{\mathord}{letters}{"27}
\DeclareMathSymbol{\Gamma}{\mathalpha}{operators}{"00}
\DeclareMathSymbol{\Delta}{\mathalpha}{operators}{"01}
\DeclareMathSymbol{\Theta}{\mathalpha}{operators}{"02}
\DeclareMathSymbol{\Lambda}{\mathalpha}{operators}{"03}
\DeclareMathSymbol{\Xi}{\mathalpha}{operators}{"04}
\DeclareMathSymbol{\Pi}{\mathalpha}{operators}{"05}
\DeclareMathSymbol{\Sigma}{\mathalpha}{operators}{"06}
\DeclareMathSymbol{\Upsilon}{\mathalpha}{operators}{"07}
\DeclareMathSymbol{\Phi}{\mathalpha}{operators}{"08}
\DeclareMathSymbol{\Psi}{\mathalpha}{operators}{"09}
\DeclareMathSymbol{\Omega}{\mathalpha}{operators}{"0A}
```

6.4.2 Ordinary symbols

```
\DeclareMathSymbol{\aleph}{\mathord}{symbols}{"40}
\DeclareMathSymbol{\imath}{\mathord}{letters}{"7B}
\DeclareMathSymbol{\jmath}{\mathord}{letters}{"7C}
\DeclareMathSymbol{\ell}{\mathord}{letters}{"60}
\DeclareMathSymbol{\wp}{\mathord}{letters}{"7D}
\DeclareMathSymbol{\Re}{\mathord}{symbols}{"3C}
\DeclareMathSymbol{\Im}{\mathord}{symbols}{"3D}
\DeclareMathSymbol{\partial}{\mathord}{letters}{"40}
\DeclareMathSymbol{\infty}{\mathord}{symbols}{"31}
\DeclareMathSymbol{\prime}{\mathord}{symbols}{"30}
\DeclareMathSymbol{\emptyset}{\mathord}{symbols}{"3B}
\DeclareMathSymbol{\nabla}{\mathord}{symbols}{"72}
\DeclareMathSymbol{\top}{\mathord}{symbols}{"3E}
\DeclareMathSymbol{\bot}{\mathord}{symbols}{"3F}
\DeclareMathSymbol{\triangle}{\mathord}{symbols}{"34}
\DeclareMathSymbol{\forall}{\mathord}{symbols}{"38}
\DeclareMathSymbol{\exists}{\mathord}{symbols}{"39}
\DeclareMathSymbol{\neg}{\mathord}{symbols}{"3A}
```

Alias:

```
%    \let\lnot=\neg
\DeclareMathSymbol{\lnot}{\mathord}{symbols}{"3A}
```

```
\DeclareMathSymbol{\flat}{\mathord}{letters}{"5B}
\DeclareMathSymbol{\natural}{\mathord}{letters}{"5C}
\DeclareMathSymbol{\sharp}{\mathord}{letters}{"5D}
\DeclareMathSymbol{\clubsuit}{\mathord}{symbols}{"7C}
\DeclareMathSymbol{\diamondsuit}{\mathord}{symbols}{"7D}
\DeclareMathSymbol{\heartsuit}{\mathord}{symbols}{"7E}
\DeclareMathSymbol{\spadesuit}{\mathord}{symbols}{"7F}
```

```
\DeclareRobustCommand\hbar{{\mathchar'26\mkern-9muh}}
\DeclareRobustCommand\surd{{\mathchar"1270}}
\DeclareRobustCommand\angle{{\vbox{\ialign{$\m@th\scriptstyle##$\crcr
      \not\mathrel{\mkern14mu}\crcr
      \noalign{\nointerlineskip}
```

```
      \mkern2.5mu\leaders\hrule \@height.34pt\hfill\mkern2.5mu\crcr}}}}
```

6.4.3 Large Operators

```
\DeclareMathSymbol{\coprod}{\mathop}{largesymbols}{"60}
\DeclareMathSymbol{\bigvee}{\mathop}{largesymbols}{"57}
\DeclareMathSymbol{\bigwedge}{\mathop}{largesymbols}{"56}
\DeclareMathSymbol{\biguplus}{\mathop}{largesymbols}{"55}
\DeclareMathSymbol{\bigcap}{\mathop}{largesymbols}{"54}
\DeclareMathSymbol{\bigcup}{\mathop}{largesymbols}{"53}
\DeclareMathSymbol{\intop}{\mathop}{largesymbols}{"52}
    \DeclareRobustCommand\int{\intop\nolimits}
\DeclareMathSymbol{\prod}{\mathop}{largesymbols}{"51}
\DeclareMathSymbol{\sum}{\mathop}{largesymbols}{"50}
\DeclareMathSymbol{\bigotimes}{\mathop}{largesymbols}{"4E}
\DeclareMathSymbol{\bigoplus}{\mathop}{largesymbols}{"4C}
\DeclareMathSymbol{\bigodot}{\mathop}{largesymbols}{"4A}
\DeclareMathSymbol{\ointop}{\mathop}{largesymbols}{"48}
    \DeclareRobustCommand\oint{\ointop\nolimits}
\DeclareMathSymbol{\bigsqcup}{\mathop}{largesymbols}{"46}
\DeclareMathSymbol{\smallint}{\mathop}{symbols}{"73}
```

6.4.4 Binary symbols

```
\DeclareMathSymbol{\triangleleft}{\mathbin}{letters}{"2F}
\DeclareMathSymbol{\triangleright}{\mathbin}{letters}{"2E}
\DeclareMathSymbol{\bigtriangleup}{\mathbin}{symbols}{"34}
\DeclareMathSymbol{\bigtriangledown}{\mathbin}{symbols}{"35}
```

Alias:

```
%   \let \varbigtriangledown \bigtriangledown
%   \let \varbigtriangleup \bigtriangleup
\DeclareMathSymbol{\varbigtriangleup}{\mathbin}{symbols}{"34}
\DeclareMathSymbol{\varbigtriangledown}{\mathbin}{symbols}{"35}
```

These last two synonyms are needed because the stmaryrd package redefines them as Operators.

```
\DeclareMathSymbol{\wedge}{\mathbin}{symbols}{"5E}
\DeclareMathSymbol{\vee}{\mathbin}{symbols}{"5F}
```

Alias:

```
%   \let\land=\wedge
%   \let\lor=\vee
\DeclareMathSymbol{\land}{\mathbin}{symbols}{"5E}
\DeclareMathSymbol{\lor}{\mathbin}{symbols}{"5F}
```

```
\DeclareMathSymbol{\cap}{\mathbin}{symbols}{"5C}
\DeclareMathSymbol{\cup}{\mathbin}{symbols}{"5B}
\DeclareMathSymbol{\ddagger}{\mathbin}{symbols}{"7A}
\DeclareMathSymbol{\dagger}{\mathbin}{symbols}{"79}
\DeclareMathSymbol{\sqcap}{\mathbin}{symbols}{"75}
\DeclareMathSymbol{\sqcup}{\mathbin}{symbols}{"74}
\DeclareMathSymbol{\uplus}{\mathbin}{symbols}{"5D}
\DeclareMathSymbol{\amalg}{\mathbin}{symbols}{"71}
\DeclareMathSymbol{\diamond}{\mathbin}{symbols}{"05}
\DeclareMathSymbol{\bullet}{\mathbin}{symbols}{"0F}
\DeclareMathSymbol{\wr}{\mathbin}{symbols}{"6F}
\DeclareMathSymbol{\div}{\mathbin}{symbols}{"04}
```

```
\DeclareMathSymbol{\odot}{\mathbin}{symbols}{"0C}
\DeclareMathSymbol{\oslash}{\mathbin}{symbols}{"0B}
\DeclareMathSymbol{\otimes}{\mathbin}{symbols}{"0A}
\DeclareMathSymbol{\ominus}{\mathbin}{symbols}{"09}
\DeclareMathSymbol{\oplus}{\mathbin}{symbols}{"08}
\DeclareMathSymbol{\mp}{\mathbin}{symbols}{"07}
\DeclareMathSymbol{\pm}{\mathbin}{symbols}{"06}
\DeclareMathSymbol{\circ}{\mathbin}{symbols}{"0E}
\DeclareMathSymbol{\bigcirc}{\mathbin}{symbols}{"0D}
\DeclareMathSymbol{\setminus}{\mathbin}{symbols}{"6E}
\DeclareMathSymbol{\cdot}{\mathbin}{symbols}{"01}
\DeclareMathSymbol{\ast}{\mathbin}{symbols}{"03}
\DeclareMathSymbol{\times}{\mathbin}{symbols}{"02}
\DeclareMathSymbol{\star}{\mathbin}{letters}{"3F}
```

6.4.5 Relations

```
\DeclareMathSymbol{\propto}{\mathrel}{symbols}{"2F}
\DeclareMathSymbol{\sqsubseteq}{\mathrel}{symbols}{"76}
\DeclareMathSymbol{\sqsupseteq}{\mathrel}{symbols}{"77}
\DeclareMathSymbol{\parallel}{\mathrel}{symbols}{"6B}
\DeclareMathSymbol{\mid}{\mathrel}{symbols}{"6A}
\DeclareMathSymbol{\dashv}{\mathrel}{symbols}{"61}
\DeclareMathSymbol{\vdash}{\mathrel}{symbols}{"60}
\DeclareMathSymbol{\nearrow}{\mathrel}{symbols}{"25}
\DeclareMathSymbol{\searrow}{\mathrel}{symbols}{"26}
\DeclareMathSymbol{\nwarrow}{\mathrel}{symbols}{"2D}
\DeclareMathSymbol{\swarrow}{\mathrel}{symbols}{"2E}
\DeclareMathSymbol{\Leftrightarrow}{\mathrel}{symbols}{"2C}
\DeclareMathSymbol{\Leftarrow}{\mathrel}{symbols}{"28}
\DeclareMathSymbol{\Rightarrow}{\mathrel}{symbols}{"29}
   \DeclareRobustCommand\neq{\not=}
```

As `\neq` is robust we should not use `\let` to define `\ne` as then it would change if `\neq` changes.

```
   \DeclareRobustCommand\ne{\not=}
```

It would ok to use `\let` for those declared by `\DeclareMathSymbol` but for a cleaner interface we avoid it always (just in case the internals change).

```
\DeclareMathSymbol{\leq}{\mathrel}{symbols}{"14}
\DeclareMathSymbol{\geq}{\mathrel}{symbols}{"15}
```

Alias:

```
%   \let\le=\leq
%   \let\ge=\geq
\DeclareMathSymbol{\le}{\mathrel}{symbols}{"14}
\DeclareMathSymbol{\ge}{\mathrel}{symbols}{"15}
```

```
\DeclareMathSymbol{\succ}{\mathrel}{symbols}{"1F}
\DeclareMathSymbol{\prec}{\mathrel}{symbols}{"1E}
\DeclareMathSymbol{\approx}{\mathrel}{symbols}{"19}
\DeclareMathSymbol{\succeq}{\mathrel}{symbols}{"17}
\DeclareMathSymbol{\preceq}{\mathrel}{symbols}{"16}
\DeclareMathSymbol{\supset}{\mathrel}{symbols}{"1B}
\DeclareMathSymbol{\subset}{\mathrel}{symbols}{"1A}
\DeclareMathSymbol{\supseteq}{\mathrel}{symbols}{"13}
\DeclareMathSymbol{\subseteq}{\mathrel}{symbols}{"12}
```

```
\DeclareMathSymbol{\in}{\mathrel}{symbols}{"32}
\DeclareMathSymbol{\ni}{\mathrel}{symbols}{"33}
```

Alias:

```
%    \let\owns=\ni
\DeclareMathSymbol{\owns}{\mathrel}{symbols}{"33}
```

```
\DeclareMathSymbol{\gg}{\mathrel}{symbols}{"1D}
\DeclareMathSymbol{\ll}{\mathrel}{symbols}{"1C}
\DeclareMathSymbol{\not}{\mathrel}{symbols}{"36}
\DeclareMathSymbol{\leftrightarrow}{\mathrel}{symbols}{"24}
\DeclareMathSymbol{\leftarrow}{\mathrel}{symbols}{"20}
\DeclareMathSymbol{\rightarrow}{\mathrel}{symbols}{"21}
```

Alias:

```
%   \let\gets=\leftarrow
%   \let\to=\rightarrow
\DeclareMathSymbol{\gets}{\mathrel}{symbols}{"20}
\DeclareMathSymbol{\to}{\mathrel}{symbols}{"21}
```

```
\DeclareMathSymbol{\mapstochar}{\mathrel}{symbols}{"37}
   \DeclareRobustCommand\mapsto{\mapstochar\rightarrow}
\DeclareMathSymbol{\sim}{\mathrel}{symbols}{"18}
\DeclareMathSymbol{\simeq}{\mathrel}{symbols}{"27}
\DeclareMathSymbol{\perp}{\mathrel}{symbols}{"3F}
\DeclareMathSymbol{\equiv}{\mathrel}{symbols}{"11}
\DeclareMathSymbol{\asymp}{\mathrel}{symbols}{"10}
\DeclareMathSymbol{\smile}{\mathrel}{letters}{"5E}
\DeclareMathSymbol{\frown}{\mathrel}{letters}{"5F}
\DeclareMathSymbol{\leftharpoonup}{\mathrel}{letters}{"28}
\DeclareMathSymbol{\leftharpoondown}{\mathrel}{letters}{"29}
\DeclareMathSymbol{\rightharpoonup}{\mathrel}{letters}{"2A}
\DeclareMathSymbol{\rightharpoondown}{\mathrel}{letters}{"2B}
```

Here cometh much profligate robustification of math constructs. Warning: some of these commands may become non-robust if an AMS package is loaded.

Further potential problems: some math font packages may make unfortunate assumptions about some of these definitions that are not true of the robust versions we need.

```
\DeclareRobustCommand
  \cong{\mathrel{\mathpalette\@vereq\sim}} % congruence sign
\def\@vereq#1#2{\lower.5\p@\vbox{\lineskiplimit\maxdimen\lineskip-.5\p@
    \ialign{$\m@th#1\hfil##\hfil$\crcr#2\crcr=\crcr}}}
\DeclareRobustCommand
  \notin{\mathrel{\m@th\mathpalette\c@ncel\in}}
\def\c@ncel#1#2{\m@th\ooalign{$\hfil#1\mkern1mu/\hfil$\crcr$#1#2$}}
\DeclareRobustCommand
  \rightleftharpoons{\mathrel{\mathpalette\rlh@{}}}
\def\rlh@#1{\vcenter{\m@th\hbox{\ooalign{\raise2pt
          \hbox{$#1\rightharpoonup$}\crcr
        $#1\leftharpoondown$}}}}
\DeclareRobustCommand
  \doteq{\buildrel\textstyle.\over=}
```

6.4.6 Arrows

```
\DeclareRobustCommand
  \joinrel{\mathrel{\mkern-3mu}}
```

```
\DeclareRobustCommand
  \relbar{\mathrel{\smash-}} % \smash, because -
                              % has the same height as +
```

In contrast to `plain.tex` `\Relbar` got braces around the equal sign to guard against it being "math active" expanding to `\futurelet`.... This might be the case when packages are implementing shorthands for math, e.g. `=>` meaning `\Rightarrow` etc. It would actually be better not to use `=` in such definitions but instead define something like `\mathequalsign` and use this. However we can't do this now as it would break other math layouts where characters are in different places (since those wouldn't know about the need for a new command name).

```
\DeclareRobustCommand
  \Relbar{\mathrel{=}}
\DeclareMathSymbol{\lhook}{\mathrel}{letters}{"2C}
   \DeclareRobustCommand\hookrightarrow{\lhook\joinrel\rightarrow}
\DeclareMathSymbol{\rhook}{\mathrel}{letters}{"2D}
   \DeclareRobustCommand\hookleftarrow{\leftarrow\joinrel\rhook}
\DeclareRobustCommand
  \bowtie{\mathrel\triangleright\joinrel\mathrel\triangleleft}
```

```
\DeclareRobustCommand
  \models{\mathrel{|}\joinrel\Relbar}
\DeclareRobustCommand
  \Longrightarrow{\Relbar\joinrel\Rightarrow}
```

LaTeX Change: `\longrightarrow` and `\longleftarrow` redefined to make then robust.

```
\DeclareRobustCommand\longrightarrow
     {\relbar\joinrel\rightarrow}
\DeclareRobustCommand\longleftarrow
     {\leftarrow\joinrel\relbar}
```

```
\DeclareRobustCommand
  \Longleftarrow{\Leftarrow\joinrel\Relbar}
\DeclareRobustCommand
  \longmapsto{\mapstochar\longrightarrow}
\DeclareRobustCommand
  \longleftrightarrow{\leftarrow\joinrel\rightarrow}
\DeclareRobustCommand
  \Longleftrightarrow{\Leftarrow\joinrel\Rightarrow}
\DeclareRobustCommand
  \iff{\;\Longleftrightarrow\;}
```

6.4.7 Punctuation symbols

```
\DeclareMathSymbol{\ldotp}{\mathpunct}{letters}{"3A}
\DeclareMathSymbol{\cdotp}{\mathpunct}{symbols}{"01}
\DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}
```

This is commented out, since `\ldots` is now defined in ltoutenc.dtx.

```
%\def\@ldots{\mathinner{\ldotp\ldotp\ldotp}}
%\DeclareRobustCommand\ldots
%          {\relax\ifmmode\@ldots\else\mbox{$\m@th\@ldots\,$}\fi}
```

```
\DeclareRobustCommand
  \cdots{\mathinner{\cdotp\cdotp\cdotp}}
\DeclareRobustCommand
  \vdots{\vbox{\baselineskip4\p@ \lineskiplimit\z@
```

```
    \kern6\p@\hbox{.}\hbox{.}\hbox{.}}}
\DeclareRobustCommand
  \ddots{\mathinner{\mkern1mu\raise7\p@
    \vbox{\kern7\p@\hbox{.}}\mkern2mu
    \raise4\p@\hbox{.}\mkern2mu\raise\p@\hbox{.}\mkern1mu}}
```

6.4.8 Math accents

```
\DeclareMathAccent{\acute}{\mathalpha}{operators}{"13}
\DeclareMathAccent{\grave}{\mathalpha}{operators}{"12}
\DeclareMathAccent{\ddot}{\mathalpha}{operators}{"7F}
\DeclareMathAccent{\tilde}{\mathalpha}{operators}{"7E}
\DeclareMathAccent{\bar}{\mathalpha}{operators}{"16}
\DeclareMathAccent{\breve}{\mathalpha}{operators}{"15}
\DeclareMathAccent{\check}{\mathalpha}{operators}{"14}
\DeclareMathAccent{\hat}{\mathalpha}{operators}{"5E}
\DeclareMathAccent{\vec}{\mathord}{letters}{"7E}
\DeclareMathAccent{\dot}{\mathalpha}{operators}{"5F}
\DeclareMathAccent{\widetilde}{\mathord}{largesymbols}{"65}
\DeclareMathAccent{\widehat}{\mathord}{largesymbols}{"62}
```

For some reason plain TEX never bothered to provide a ring accent in math (although it is available in the fonts), but since we got a request for it here we go:

```
\DeclareMathAccent{\mathring}{\mathalpha}{operators}{"17}
```

6.4.9 Radicals

```
\DeclareMathRadical{\sqrtsign}{symbols}{"70}{largesymbols}{"70}
```

6.4.10 Over and under something, etc

```
\DeclareRobustCommand\overrightarrow[1]{\vbox{\m@th\ialign{##\crcr
      \rightarrowfill\crcr\noalign{\kern-\p@\nointerlineskip}
      $\hfil\displaystyle{#1}\hfil$\crcr}}}
\DeclareRobustCommand\overleftarrow[1]{\vbox{\m@th\ialign{##\crcr
      \leftarrowfill\crcr\noalign{\kern-\p@\nointerlineskip}%
      $\hfil\displaystyle{#1}\hfil$\crcr}}}
\DeclareRobustCommand\overbrace[1]
     {\mathop{\vbox{\m@th\ialign{##\crcr\noalign{\kern3\p@}%
      \downbracefill\crcr\noalign{\kern3\p@\nointerlineskip}%
      $\hfil\displaystyle{#1}\hfil$\crcr}}}\limits}
\DeclareRobustCommand\underbrace[1]{\mathop{\vtop{\m@th\ialign{##\crcr
   $\hfil\displaystyle{#1}\hfil$\crcr
   \noalign{\kern3\p@\nointerlineskip}%
   \upbracefill\crcr\noalign{\kern3\p@}}}}\limits}
```

(quite a waste of tokens, IMHO — Frank)

```
\DeclareRobustCommand\skew[3]
  {{\muskip\z@#1mu\divide\muskip\z@\tw@ \mkern\muskip\z@
    #2{\mkern-\muskip\z@{#3}\mkern\muskip\z@}\mkern-\muskip\z@}{}}
```

```
\DeclareRobustCommand\rightarrowfill{$\m@th\smash-\mkern-7mu%
  \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
  \mkern-7mu\mathord\rightarrow$}
\DeclareRobustCommand\leftarrowfill{$\m@th\mathord\leftarrow\mkern-7mu%
  \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
  \mkern-7mu\smash-$}
\DeclareMathSymbol{\braceld}{\mathord}{largesymbols}{"7A}
\DeclareMathSymbol{\bracerd}{\mathord}{largesymbols}{"7B}
```

```
\DeclareMathSymbol{\bracelu}{\mathord}{largesymbols}{"7C}
\DeclareMathSymbol{\braceru}{\mathord}{largesymbols}{"7D}
\DeclareRobustCommand\downbracefill{$\m@th \setbox\z@\hbox{$\braceld$}%
  \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru
  \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd$}
\DeclareRobustCommand\upbracefill{$\m@th \setbox\z@\hbox{$\braceld$}%
  \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd
  \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru$}
```

6.4.11 Delimiters

```
\DeclareMathDelimiter{\lmoustache}   % top from (, bottom from )
   {\mathopen}{largesymbols}{"7A}{largesymbols}{"40}
\DeclareMathDelimiter{\rmoustache}   % top from ), bottom from (
   {\mathclose}{largesymbols}{"7B}{largesymbols}{"41}
\DeclareMathDelimiter{\arrowvert}    % arrow without arrowheads
   {\mathord}{symbols}{"6A}{largesymbols}{"3C}
\DeclareMathDelimiter{\Arrowvert}    % double arrow without arrowheads
   {\mathord}{symbols}{"6B}{largesymbols}{"3D}
\DeclareMathDelimiter{\Vert}
   {\mathord}{symbols}{"6B}{largesymbols}{"0D}
```

`\DeclareMathDelimiter` produces a command that is robust (with an internal macro containing the payload) so we should not use `\let` for making an alias

```
%\let\|=\Vert
\DeclareMathDelimiter{\|}
   {\mathord}{symbols}{"6B}{largesymbols}{"0D}
\DeclareMathDelimiter{\vert}
   {\mathord}{symbols}{"6A}{largesymbols}{"0C}
\DeclareMathDelimiter{\uparrow}
   {\mathrel}{symbols}{"22}{largesymbols}{"78}
\DeclareMathDelimiter{\downarrow}
   {\mathrel}{symbols}{"23}{largesymbols}{"79}
\DeclareMathDelimiter{\updownarrow}
   {\mathrel}{symbols}{"6C}{largesymbols}{"3F}
\DeclareMathDelimiter{\Uparrow}
   {\mathrel}{symbols}{"2A}{largesymbols}{"7E}
\DeclareMathDelimiter{\Downarrow}
   {\mathrel}{symbols}{"2B}{largesymbols}{"7F}
\DeclareMathDelimiter{\Updownarrow}
   {\mathrel}{symbols}{"6D}{largesymbols}{"77}
\DeclareMathDelimiter{\backslash}    % for double coset G\backslash H
   {\mathord}{symbols}{"6E}{largesymbols}{"0F}
\DeclareMathDelimiter{\rangle}
   {\mathclose}{symbols}{"69}{largesymbols}{"0B}
\DeclareMathDelimiter{\langle}
   {\mathopen}{symbols}{"68}{largesymbols}{"0A}
\DeclareMathDelimiter{\rbrace}
   {\mathclose}{symbols}{"67}{largesymbols}{"09}
\DeclareMathDelimiter{\lbrace}
   {\mathopen}{symbols}{"66}{largesymbols}{"08}
\DeclareMathDelimiter{\rceil}
   {\mathclose}{symbols}{"65}{largesymbols}{"07}
\DeclareMathDelimiter{\lceil}
   {\mathopen}{symbols}{"64}{largesymbols}{"06}
```

```
\DeclareMathDelimiter{\rfloor}
   {\mathclose}{symbols}{"63}{largesymbols}{"05}
\DeclareMathDelimiter{\lfloor}
   {\mathopen}{symbols}{"62}{largesymbols}{"04}
```

\lgroup
\rgroup
\bracevert

There are three plain TEX delimiters which are not fully supported by NFSS, since they partly point into a bold cmr font. Allocating a full symbol font, just to have three delimiters seems a bit too much given the limited space available. For this reason only the extensible sizes are supported. If this is not desired one can use, without losing portability, define \mathbf and \mathtt as font symbol alphabet (setting up cmr/bx/n and cmtt/m/n as symbol fonts first) and modify the delimiter declarations to point with their small variant to those symbol fonts. (This is done in oldlfont.dtx so look there for examples.)

```
\DeclareMathDelimiter{\lgroup} % extensible ( with sharper tips
     {\mathopen}{largesymbols}{"3A}{largesymbols}{"3A}
\DeclareMathDelimiter{\rgroup} % extensible ) with sharper tips
     {\mathclose}{largesymbols}{"3B}{largesymbols}{"3B}
\DeclareMathDelimiter{\bracevert} % the vertical bar that extends braces
     {\mathord}{largesymbols}{"3E}{largesymbols}{"3E}
```

(*End of definition for* \lgroup, \rgroup, *and* \bracevert.)

6.5 Math versions of text commands

The \mathunderscore here is really a text definition, so it has been put back into ltoutenc.dtx (by Chris, 30/04/97) and should be removed from here.

These symbols are the math versions of text commands such as \P, \$, etc.

\mathparagraph
\mathsection
\mathdollar
\mathsterling
\mathunderscore

These math symbols are not in plain TEX.

```
\DeclareMathSymbol{\mathparagraph}{\mathord}{symbols}{"7B}
\DeclareMathSymbol{\mathsection}{\mathord}{symbols}{"78}
\DeclareMathSymbol{\mathdollar}{\mathord}{operators}{"24}
\DeclareRobustCommand\mathsterling{\mathit{\mathchar"7024}}
\DeclareRobustCommand\mathunderscore{\kern.06em\vbox{\hrule\@width.3em}}
```

(*End of definition for* \mathparagraph *and others.*)

\mathellipsis

This is plain TEX's \ldots.

```
\DeclareRobustCommand\mathellipsis{\mathinner{\ldotp\ldotp\ldotp}}%
```

(*End of definition for* \mathellipsis.)

6.6 Other special functions and parameters

6.6.1 Biggggg

```
⟨/math⟩
⟨*math | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                    {\Big}{Start LR-mode}%
\DeclareRobustCommand\big[1]{\leavevmode@ifvmode
   {\hbox{$\left#1\vbox to8.5\p@{}\right.\n@space$}}}
\DeclareRobustCommand\Big[1]{\leavevmode@ifvmode
   {\hbox{$\left#1\vbox to11.5\p@{}\right.\n@space$}}}
```

```
\DeclareRobustCommand\bigg[1]{\leavevmode@ifvmode
   {\hbox{$\left#1\vbox to14.5\p@{}\right.\n@space$}}}
\DeclareRobustCommand\Bigg[1]{\leavevmode@ifvmode
   {\hbox{$\left#1\vbox to17.5\p@{}\right.\n@space$}}}
⟨/math | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\Big}{Start LR-mode}%
⟨latexrelease⟩\def\big#1{{\hbox{$\left#1\vbox to8.5\p@{}\right.\n@space$}}}
⟨latexrelease⟩\def\Big#1{{\hbox{$\left#1\vbox to11.5\p@{}\right.\n@space$}}}
⟨latexrelease⟩\def\bigg#1{{\hbox{$\left#1\vbox to14.5\p@{}\right.\n@space$}}}
⟨latexrelease⟩\def\Bigg#1{{\hbox{$\left#1\vbox to17.5\p@{}\right.\n@space$}}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*math⟩
```

```
\def\n@space{\nulldelimiterspace\z@ \m@th}
```

6.6.2 The log-like functions

`\operator@font` The `\operator@font` determines the symbol font used for log-like functions.

```
\def\operator@font{\mathgroup\symoperators}
```

(*End of definition for* `\operator@font`.)

6.6.3 Parameters

```
\thinmuskip=3mu
\medmuskip=4mu plus 2mu minus 4mu
\thickmuskip=5mu plus 5mu
```

This finishes the low-level setup in `fontmath.ltx`.

```
⟨/math⟩
```

7 Default cfg files

We provide default `cfg` files here to ensure that on installations that search large file trees we do not pick up some strange customisation files from somewhere.

```
⟨*cfgtext | cfgmath | cfgprel⟩
%%
%%
%%
%% Load the standard setup:
%%
⟨+cfgtext⟩\input{fonttext.ltx}
⟨+cfgmath⟩\input{fontmath.ltx}
⟨+cfgprel⟩\input{preload.ltx}
%%
%% Small changes could go here; see documentation in cfgguide.tex for
%% allowed modifications.
%%
%% In particular it is not allowed to misuse this configuration file
%% to modify internal LaTeX commands!
%%
%% If you use this file as the basis for configuration please change
%% the \ProvidesFile lines to clearly identify your modification, e.g.,
%%
```

```
⟨+cfgtext⟩%%   \ProvidesFile{fonttext.cfg}[2001/06/01
⟨+cfgmath⟩%%   \ProvidesFile{fonttext.cfg}[2001/06/01
⟨+cfgprel⟩%%    \ProvidesFile{preload.cfg}[2001/06/01
%%                                Customised local font setup]
%%
%%
⟨/cfgtext | cfgmath | cfgprel⟩
```

File 31
preload.dtx

1 Overview

This file contains an number of possible settings for preloading fonts during installation of NFSS2 (which is used by LaTeX 2_ε). It will be used to generate the following files:

preload.min	minimal subset of fonts necessary to run NFSS2
preload.ori	preload of CM fonts similar to the old `lfonts.tex`
preload.ltx	The standard selection of preloads
cmpreloa.xpt	preload of CM fonts for 10pt document size
cmpreloa.xip	preload of CM fonts for 11pt document size
cmpreloa.xii	preload of CM fonts for 12pt document size
dcpreloa.xpt	preload of DC fonts for 10pt size
dcpreloa.xip	preload of DC fonts for 11pt size
dcpreloa.xii	preload of DC fonts for 12pt size

These files are for installations that make use of Computer Modern fonts either old encoding (OT1) or Cork encoding (T1). The Computer Modern fonts with Cork encoding are known as DC-fonts.

Most important is `preload.ltx` which is used during format generation. You are *not* allowed to change this file.

2 Customization

You can customize the preloaded fonts in your LaTeX 2_ε system by installing a file with the name `preload.cfg`. If this file exists it will be used in place of the system file `preload.ltx`. You can, for example, copy one of the files mentioned above (that can be generated from this source) to `preload.cfg`.

Or you can define completely other preloads. In that case start from `preload.min` since that contains the fonts that have to be preloaded by *all* LaTeX 2_ε systems.

Avoid using `preload.ori`, it will load so many fonts that on most installations it is nearly impossible to load other font families afterwards. This file is only generated to show what fonts have been preloaded by LaTeX 2.09.

If you normally use other fonts than Computer Modern `preload.min` might be best.

> **Warning:** If you preload fonts with encodings other than the normally supported encodings you have to declare that encoding in a `fontdef.cfg` configuration file (see the documentation in the file `fontdef.dtx`). Adding an extra encoding to the format might produce non-portable documents, thus this should be avoided if possible.

3 Module switches for the DOCSTRIP program

The DOCSTRIP will generate the above file from this source using the following module directives:

driver	produce a documentation driver file
preload	produce a preload... file
cm	for OT1 encoded Computer Modern
dc	for T1 encoded Computer Modern
min	produce minimal subset
xpt	produce 10pt preloads
xipt	produce 11pt preloads
xiipt	produce 12pt preloads
ori	produce preloads similar to old `lfonts.tex`
tex	produce preload.ltx

A typical DOCSTRIP command file would then have entries like:

```
generateFile{preload.min}{t}{\from{preload.dtx}{preload,min}}
```

for generating preload files.

4 A driver for this document

The next bit of code contains the documentation driver file for TEX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```
⟨*driver⟩
\documentclass{ltxdoc}
%\OnlyDescription  % comment out for implementation details
\begin{document}
   \DocInput{preload.dtx}
\end{document}
⟨/driver⟩
```

5 The code

We begin by loading the math extension font (cmex10) and the LATEX line and circle fonts. It is necessary to do this explicitly since these are used by the LATEX format. Since the internal font name contains `/` characters and digits we construct the name via `\csname`. These are the only fonts (!) that must be loaded in this file.

All `\DeclarePreloadSizes` can be removed or others can be added, they only influence the processing speed.

```
\expandafter\font\csname OMX/cmex/m/n/10\endcsname=cmex10\relax
\font\tenln  =line10   \font\tenlnw  =linew10\relax
\font\tencirc=lcircle10 \font\tencircw=lcirclew10\relax
```

The above fonts should not be touched but anything below this point here in the preload suggestions can be modified without any problems.

```
⟨-tex⟩%*****************************************
⟨-tex⟩% Start any modification below this point **
⟨-tex⟩%*****************************************
⟨-tex⟩
%%
%% Computer Modern Roman:
%%-----------------------
```

```
⟨*ori⟩
\DeclarePreloadSizes{OT1}{cmr}{m}{n}
        {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
\DeclarePreloadSizes{OT1}{cmr}{bx}{n}{9,10,10.95,12,14.4,17.28}
\DeclarePreloadSizes{OT1}{cmr}{m}{sl}{10,10.95,12}
\DeclarePreloadSizes{OT1}{cmr}{m}{it}{7,8,9,10,10.95,12}
⟨/ori⟩
⟨+xpt & cm⟩ \DeclarePreloadSizes{OT1}{cmr}{m}{n}{5,7,10}
⟨+xpt & dc⟩ \DeclarePreloadSizes{T1}{cmr}{m}{n}{5,7,10}
⟨+xipt & cm⟩ \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,10.95}
⟨+xipt & dc⟩ \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,10.95}
⟨+xiipt & cm⟩ \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,12}
⟨+xiipt & dc⟩ \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,12}
%%
%% Computer Modern Sans:
%%----------------------
⟨+ori⟩ \DeclarePreloadSizes{OT1}{cmss}{m}{n}{10,10.95,12}
%%
%% Computer Modern Typewriter:
%%----------------------------
⟨+ori⟩ \DeclarePreloadSizes{OT1}{cmtt}{m}{n}{9,10,10.95,12}
%%
%% Computer Modern Math:
%%----------------------
⟨*ori⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}
        {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}
        {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
⟨/ori⟩
```

The math fonts are the same for both DC and CM fonts. So far there isn't an agreed on standard.

```
⟨*xpt⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}{5,7,10}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}{5,7,10}
⟨/xpt⟩
⟨*xipt⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,10.95}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,10.95}
⟨/xipt⟩
⟨*xiipt⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,12}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,12}
⟨/xiipt⟩
%%
%% LaTeX symbol fonts:
%%-------------------
⟨*ori⟩
\DeclarePreloadSizes{U}{lasy}{m}{n}
        {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
⟨/ori⟩
⟨/preload⟩
```

File 32
ltfntcmd.dtx

Abstract

The commands defined in this file `ltfntcmd` are part of the kernel code for LaTeX 2_ε/NFSS2.
It is also meant to serve as documentation for package writers since it demonstrates how to define high-level font changing commands using a small number of creator functions.

1 Introduction

Font changes such as `\bfseries`, `\sffamily`, etc. are declarations; this means that their scope is delimited by the grouping structure, either by the next `\end` of some environment or by explicitly using a group, e.g., writing something like `{\bfseries...}` in the source. If you make the mistake of writing `\bfseries{...}` (thinking of `\bfseries` as a command with one argument) then the result is rather striking.

Font declarations are an artifact of the TeX system and for several reasons it is better to avoid them on the user level whenever possible. In LaTeX3 they will probably all be replaced by environments and by font commands taking one argument.

This file defines a creator function for such declarative font switches. This function creates commands which can be used in both math and text.

This file also defines a number of high-level commands (all starting with `\text..`) that have one argument and typeset this argument in the requested way. Thus these commands are for typesetting short pieces of text in a specific family, series or shape. These are all produced as examples of the use of a creator function which is itself also defined in this file.

Table 3 shows all these high-level commands in action. A further advantage of using these commands is that they automatically take care of any necessary italic correction on either side of their argument.

Thus, when using such commands, one does not have to worry about forgetting the italic correction when changing fonts. Only in very few situations is this additional space wrong but, for example, most typographers recommend omitting the italic correction if a small punctuation character, like a comma, directly follows the font change. Since the amount of correction required is partly a matter of taste, you can define in what situations the italic correction should be suppressed. This is done by putting the characters that should cancel a preceding italic correction in the list `\nocorrlist`.[38] The default definition for this list is produced by the following.

```
\newcommand \nocorrlist {,.}
```

It is best to declare the most often used characters first, because this will make the processing slightly faster. For example,

```
\emph{When using the \NFSS{} high-level commands,
the \emph{proper} use of italic corrections is
automatically taken care of}. Only
```

[38] Any package that changes the `\catcode` of a character inside `\nocorrlist` must then explicitly reset the list. Otherwise the changed character will no longer be recognized by the suppression algorithm.

Command	*Corresponds to*	*Action*
`\textnormal{..}`	`\normalfont`	Typeset argument in normal family
`\textrm{..}`	`\rmfamily`	Typeset argument in roman family
`\textsf{..}`	`\sffamily`	Typeset argument in sans serif family
`\texttt{..}`	`\ttfamily`	Typeset argument in `typewriter` family
`\textmd{..}`	`\mdseries`	Typeset argument in medium series
`\textbf{..}`	`\bfseries`	Typeset argument in **bold** series
`\textup{..}`	`\upshape`	Typeset argument in normal shape
`\textit{..}`	`\itshape`	Typeset argument in *italic* shape
`\textsl{..}`	`\slshape`	Typeset argument in *slanted* shape
`\textsc{..}`	`\scshape`	Typeset argument in SMALL CAPS shape
`\emph{..}`	`\em`	Typeset argument *emphasized*

Table 3: Font-change commands with arguments

The font change commands provided here all start with `\text..` to emphasize that they are for use in normal text and to be easily memorable. They automatically take care of any necessary italic correction on either side of the argument.

```
\emph{sometimes} one has to help \LaTeX{} by
adding a \verb=\nocorr= command.
```

which results in:

> *When using the NFSS high-level commands, the* proper *use of italic corrections is automatically taken care of.* Only *sometimes* one has to help LaTeX by adding a `\nocorr` command.

In contrast, the use of the declaration forms is often more appropriate when you define your own commands or environments.

```
\newenvironment{bfitemize}{\begin{itemize}\normalfont\bfseries}
                          {\end{itemize}}
\begin{bfitemize}
\item This environment produces boldface items.
\item It is defined in terms of \LaTeX's
  \texttt{itemize} environment and NFSS
  declarations.
\end{bfitemize}
```

This gives:

- **This environment produces boldface items.**
- **It is defined in terms of LaTeX's `itemize` environment and NFSS declarations.**

In addition to global customization of when to insert the italic correction, it is of course sometimes necessary to explicitly insert one with `\/`.

It is also possible to suppress the italic correction in individual instances. For this, the command `\nocorr` is provided.

The `\nocorr` must appear as the first or last token inside the braces of the argument of the `\text...` commands, at that end of the text where you wish to suppress the italic correction.

It is worth pointing out here that inserting a `\/` in places where it can have no function (i.e. anywhere except immediately after a slanted letter) is not an error—it will just be silently ignored. Unfortunately this is not true if the redefinition of `\/` in `amstex.sty` is used as this version can cause space to be removed immediately before the `\/`.

2 The implementation

`\DeclareTextFontCommand` This is the creator function for `\text..` commands. It gives a warning if `\foo` or `\fragfoo` is already defined.

In math mode it simply puts the font declaration and text into a box (possibly an automagically sized one).

Otherwise it first scans the text to see where `\nocorr` occurs within it. This sets the `\check@ic` commands to do what is necessary concerning the italic correction at both ends.

The algorithm for deciding whether to put in an italic correction is not very subtle: one is added whenever the newly current font is not itself positively sloped, unless the next token is a character in the 'nocorr' list. At the end of the text this is done after closing the group so as to check the 'outer font'. Note that this will often result in adding an italic correction token after a character in an unsloped font; we believe (in early 2003) that this is perhaps inefficient but not dangerous.

It also now checks for empty contents of the text command and optimizes this case. Some care is also taken to check that doing dangerous things in vertical mode is avoided.

The italic correction token is added to the horizontal list before (in the list) an immediately preceding non-zero glob of glue (skip) and any non-zero penalty preceding that since, in the typical case, this puts it immediately after the last character in the preceding word.

Note that it is necessary to put in the `\aftergroup\maybe@ic` at the end of the group so that it comes after any other aftergroup tokens and immediately before the following tokens. It is also necessary to remove the `\fi` from the token list before the group ends; this is done by adding an `\expandafter` just before the closing brace.

```
⟨*2ekernel⟩
\def \DeclareTextFontCommand #1#2{%
  \DeclareRobustCommand#1[1]{%
    \ifmmode
      \nfss@text{#2##1}%
    \else
      \hmode@bgroup
       \text@command{##1}%
       #2\check@icl ##1\check@icr
       \expandafter
      \egroup
    \fi
                        }%
}
```

(*End of definition for* `\DeclareTextFontCommand`.)

`\textrm` `\textsf` `\texttt` `\textnormal`

Now we define the `\text`⟨*family*⟩ commands in terms of the above; `\texttt` does not look very nice!

```
\DeclareTextFontCommand{\textrm}{\rmfamily}
\DeclareTextFontCommand{\textsf}{\sffamily}
\DeclareTextFontCommand{\texttt}{\ttfamily}
\DeclareTextFontCommand{\textnormal}{\normalfont}
```

(*End of definition for* `\textrm` *and others.*)

`\textbf` `\textmd`

For the series attribute:

```
\DeclareTextFontCommand{\textbf}{\bfseries}
\DeclareTextFontCommand{\textmd}{\mdseries}
```

(*End of definition for* `\textbf` *and* `\textmd`.)

`\textit` `\textsl` `\textsc` `\textup`

And for the shapes:

```
\DeclareTextFontCommand{\textit}{\itshape}
\DeclareTextFontCommand{\textsl}{\slshape}
\DeclareTextFontCommand{\textsc}{\scshape}
\DeclareTextFontCommand{\textup}{\upshape}
```

(*End of definition for* `\textit` *and others.*)

`\textulc` `\textsw` `\textssc`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {\textulc}{Additional text commands}%
\DeclareTextFontCommand{\textulc}{\ulcshape}
\DeclareTextFontCommand{\textsw}{\swshape}
\DeclareTextFontCommand{\textssc}{\sscshape}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\textulc}{Additional text commands}%
⟨latexrelease⟩
⟨latexrelease⟩\let\textulc\@undefined
⟨latexrelease⟩\let\textsw\@undefined
⟨latexrelease⟩\let\textssc\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\textulc`, `\textsw`, *and* `\textssc`.)

`\emph`

Finally we have the `\em` font change declaration of LaTeX. The corresponding definition with argument is

```
\DeclareTextFontCommand{\emph}{\em}
```

(*End of definition for* `\emph`.)

`\nocorr`

This is just a label, so it does nothing; it should also be unexpandable.

```
\let \nocorr \relax
```

(*End of definition for* `\nocorr`.)

`\check@icl`
`\check@icr`

We define these defaults in case some error causes them to be expanded at the wrong time.

```
\let \check@icl \@empty
\let \check@icr \@empty
```

(*End of definition for* `\check@icl` *and* `\check@icr`.)

`\text@command`
`\check@nocorr@`

This checks for a `\nocorr` as the first token in its argument and also for one in any other position not protected within braces (the latter is treated as if it were at the end of the argument).

Is this the correct action in the 'empty' case? It is efficient but typographically it is, strictly, incorrect!

```
\def \text@command #1{%
  \edef \reserved@a {\unexpanded{#1}}%
  \ifx \reserved@a \@empty
    \let \check@icl \@empty
    \let \check@icr \@empty
  \else
```

`\space` is a reserved word in LaTeX or actually already in plain TeX. If somebody really redefines it so many things will break that I don't see any reason to make this routine here slower than necessary.

```
%    \def \reserved@b { }%
%    \ifx \reserved@a \reserved@b
    \ifx \reserved@a \space
      \let \check@icl \@empty
      \let \check@icr \@empty
    \else
      \check@nocorr@ #1\nocorr\@nil
    \fi
  \fi
}
\def \check@nocorr@ #1#2\nocorr#3\@nil {%
```

The two checks are initialised here to their values in the normal case.

```
  \let \check@icl \maybe@ic
  \def \check@icr {\ifvmode \else \aftergroup \maybe@ic \fi}%
  \def \reserved@a {\nocorr}%
  \def \reserved@b {#1}%
  \def \reserved@c {#3}%
  \ifx \reserved@a \reserved@b
    \ifx \reserved@c \@empty
```

In this case there is a `\nocorr` at the start but not at the end, so `\check@icl` should be empty.

```
      \let \check@icl \@empty
    \else
```

Otherwise there is a `\nocorr` both at the start and elsewhere, so no italic corrections should be added.

```
      \let \check@icl \@empty
      \let \check@icr \@empty
    \fi
```

```
  \else
    \ifx \reserved@c \@empty
```

In this case there is no \nocorr anywhere, so we need to check for an italic correction at both the beginning and the end. This has been set up as the default so no code is needed here.

```
    \else
```

In this case there is no \nocorr at the start but there is one elsewhere, so no \aftergroup is needed.

```
      \let \check@icr \@empty
    \fi
  \fi
}
```

(*End of definition for* \text@command *and* \check@nocorr@.)

\ifmaybe@ic Switch used solely within \maybe@ic not interfering with other switches.

```
\newif\ifmaybe@ic
```

(*End of definition for* \ifmaybe@ic.)

\maybe@ic \maybe@ic@ These macros implement the italic correction.

```
\def \maybe@ic {\futurelet\@let@token\maybe@ic@}
\def \maybe@ic@ {%
```

We first check to see if the current font is positively sloped. (But do not forget the message Rainer sent about an upright font with non-zero slope! Or is this an urban myth?) It has been suggested that this should test against a small positive value, but what?

```
  \ifdim  \fontdimen\@ne\font>\z@
  \else
    \maybe@ictrue
```

It would be possible, but probably not worthwhile, to continue the forward scan beyond any closing braces.

```
    \expandafter\@tfor\expandafter\reserved@a\expandafter:\expandafter=%
        \nocorrlist
```

We have to hide the \@let@token in the macro \t@st@ic rather than testing it directly in the loop since it might be \let to a \fi or \else, which would result in chaos.

```
    \do \t@st@ic
```

Frank thinks that the next bit is inefficient if done after the second change. Chris thinks that most all of this is inefficient for the commonest cases: but that is the price of a cleverer algorithm. It is certainly needed to deal with the use of \nolinebreak.

```
    \ifmaybe@ic \sw@slant \fi
  \fi
}
```

(*End of definition for* \maybe@ic *and* \maybe@ic@.)

\t@st@ic The next token in the input stream is stored in \@let@token via a \let, the current token from \nocorrlist is stored via \def in \reserved@a. To compare them we have to fiddle around a bit.

If the only things to check were characters then this could be done via an \if thus their catcodes would not matter; but this will not work whilst \futurelet is used above.

```
\def \t@st@ic {%
  \expandafter\let\expandafter\reserved@b\expandafter=\reserved@a\relax
  \ifx\reserved@b\@let@token
```

If they are the same we record the fact and jump out of the loop.

```
    \maybe@icfalse
    \@break@tfor
  \fi
}
```

(*End of definition for* \t@st@ic.)

\sw@slant The definition of the mysterious \sw@slant command is as follows.

\fix@penalty

```
\def \sw@slant {%
```

It is surely correct to put in an italic correction when there is no skip. If the last thing on the list is actually a zero skip (including things whose dimension part is zero, such as \hfill), or anything other than a character, then the italic correction will have no effect.

In order to work correctly with unbreakable spaces from ~ (and other common forms of line-breaking control) we also move back across a penalty before the glue.

```
  \ifdim \lastskip=\z@
    \fix@penalty
  \else
    \skip@ \lastskip
    \unskip
    \fix@penalty
    \hskip \skip@
  \fi
}
```

The above code means: "If there is a non-zero space just before the current position (\ifdim...) save the amount of that space (\skip@\lastskip), remove it (\unskip), then do a similar thing if there is a penalty just before the skip, and finally put the space back in."

Since zero glue cannot be distinguished in this context from no glue, we dare not put in an \hskip in this case as this may produce an unwanted breakpoint. This is not satisfactory.

The penalty before the glue is handled similarly, with the same caveats concerning the zero case. Is this the first recorded use of \unpenalty in standard LaTeX code?

```
\def \fix@penalty {%
  \ifnum \lastpenalty=\z@
    \@@italiccorr
  \else
    \count@ \lastpenalty
    \unpenalty
    \@@italiccorr
```

```
    \penalty \count@
  \fi
}
```

(*End of definition for* `\sw@slant` *and* `\fix@penalty`.)

`\nocorrlist` This holds the list of characters that should prevent italic correction. They should be ordered by decreasing frequency of use. If any such character is made active later on one needs to redefine the list so that the active character becomes part of it.

```
\def \nocorrlist {,.}
```

(*End of definition for* `\nocorrlist`.)

`\nfss@text` This command will by default behave like a LaTeX `\mbox` but may be redefined by packages such as `amstext.sty` to be a bit cleverer.

```
\ifx \nfss@text\@undefined
  \def \nfss@text {\leavevmode\hbox}
\fi
```

(*End of definition for* `\nfss@text`.)

`\DeclareOldFontCommand` This is the function used to create declarative font-changing commands that can also be used to change alphabets in math-mode.

Usage: `\DeclareOldFontCommand \fn{`⟨*font-change decls*⟩`}` ⟨*math-alphabet*⟩

Here `\fn` is the font-declaration command being defined, ⟨*font-change decls*⟩ is the declaration it will expand to in text-mode, and ⟨*math-alphabet*⟩ is the (single) math alphabet specifier which is to be used in math-mode.

It does not care whether the command being defined already exists but it does give a warning if it redefines anything.

Here are some typical examples of its use in conjunction with more basic NFSS2 font commands.

```
\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\sf}{\normalfont\sffamily}{\mathsf}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}
```

```
\def \DeclareOldFontCommand #1#2#3{%
  \DeclareRobustCommand #1{\@fontswitch {#2}{#3}}%
}
```

(*End of definition for* `\DeclareOldFontCommand`.)

`\@fontswitch` `\@@math@egroup` `\@@math@egroup` These two commands actually do the necessary tests and declarative font- or alphabet-changing.

```
\def \@fontswitch #1#2{%
  \ifmmode
     \let \math@bgroup \relax
     \def \math@egroup {\let \math@bgroup \@@math@bgroup
                        \let \math@egroup \@@math@egroup}%
```

We need to have a \relax in the following line in case the #2 is something like \mathsf grabbing the next token as an argument. For this reason the code also uses explicit arguments again (see pr/1275).

```
      #2\relax
  \else
      #1%
  \fi
}
\let \@@math@bgroup \math@bgroup
\let \@@math@egroup \math@egroup
```

(*End of definition for* \@fontswitch*,* \@@math@egroup*, and* \@@math@egroup*.*)

These commands are available only in the preamble.

```
\@onlypreamble \DeclareTextFontCommand
\@onlypreamble \DeclareOldFontCommand
```

3 Initialization

\normalsize This is defined to produce an error.

```
\def\normalsize{%
  \@latex@error {The font size command \protect\normalsize\space
              is not defined:\MessageBreak
              there is probably something wrong with
              the class file}\@eha
}
⟨/2ekernel⟩
```

(*End of definition for* \normalsize*.*)

File 33
lttextcomp.dtx

This file contains the implementation for accessing the glyphs provided by the TS1 encoding (Text Companion Encoding). This is now offered as part of the kernel and so the textcomp package which used to provide the definitions is now mainly needed for compatibility reasons (and doesn't do much any more).

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\NewModuleRelease{2020/02/02}{lttextcomp}
⟨latexrelease⟩                  {Text Companion symbols}
```

\oldstylenums
\legacyoldstylenums

Preserve the old definition of \oldstylenums under a different name.

This macro implements old style numerals but only works if we assume that the standard math fonts are used. Thus it needs changing in case other math encodings are used.

```
\DeclareRobustCommand\legacyoldstylenums[1]{%
   \begingroup
```

Provide spacing using the interword space of the current font.

```
    \spaceskip\fontdimen\tw@\font
```

Then switch to the math italic font. We don't change the current value of \f@series which means that you can use bold numerals if \bfseries is in force. As family we use \rmdefault which means that this only works if there exist an OML encoded version of that font or rather a corresponding .fd file (which is the case for standard LaTeX fonts even though they only contain substitutions).

```
    \usefont{OML}{\rmdefault}{\f@series}{it}%
    \mathgroup\symletters #1%
   \endgroup
}
```

And here is the improved one that adjusts depending on surroundings.

```
\DeclareRobustCommand\oldstylenums[1]{%
 \begingroup
 \ifmmode
   \mathgroup\symletters #1%
  \else
```

The \CheckEncodingSubset is discussed below.

```
   \CheckEncodingSubset\@use@text@encoding{TS1}\tc@oldstylesubst2{{#1}}%
 \fi
 \endgroup
}
```

The helper to select the substitution if needed.

```
\def\tc@oldstylesubst#1{%
   \tc@errorwarn
          {Oldstyle digits unavailable for
           family \f@family.\MessageBreak
           Default oldstyle digits used instead}\@eha
  \bgroup
      \expand@font@defaults
```

The substitution defaults are provided in the file `fonttext.ltx`.

```
      \ifx\f@family\rmdef@ult
         \fontfamily\rmsubstdefault
      \else\ifx\f@family\sfdef@ult
         \fontfamily\sfsubstdefault
      \else\ifx\f@family\ttdef@ult
         \fontfamily\ttsubstdefault
      \else
        \fontfamily\textcompsubstdefault
      \fi\fi\fi
      \fontencoding{TS1}\selectfont#1%
  \egroup
}
```

(*End of definition for* `\oldstylenums` *and* `\legacyoldstylenums`.)

`\textcompsubstdefault` Here is the default for the "unknown" case:

```
\def\textcompsubstdefault{\rmsubstdefault}
```

(*End of definition for* `\textcompsubstdefault`.)

`\DeclareEncodingSubset` The declaration takes 3 mandatory arguments: an *encoding* for which a subsetting is wanted (currently always `TS1`, and most likely forever), the *font family* for which we declare the subset and finally the *subset* number, with a value between `0` (all of the encoding is supported) and `9` (many glyphs are missing).

For `TS1` the numbers have been chosen in a way that most fonts can be fairly correctly categorized, but the default settings are always conservative, that is they may claim that less glyphs are supported than there actually are.

As these days many font families are set up to end in `-LF` (lining figures), `-OsF` (oldstyle figures), etc. the declaration supports a shortcut: if the *font family* name ends in `-*` then the star gets replaced by these common ending, e.g.,

```
\DeclareEncodingSubset{TS1}{Alegreya-*}{2}
```

is the same as writing

```
\DeclareEncodingSubset{TS1}{Alegreya-LF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-OsF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-TLF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-TOsF}{2}
```

If only some are needed then one can define them individually but in many cases all four are wanted, hence the shortcut.

The coding of the declaration has no error checking as it is mostly for internal use.

```
\def\DeclareEncodingSubset#1#2{%
  \DeclareEncodingSubset@aux{#1}#2*\DeclareEncodingSubset@aux
}
```

```
\def\DeclareEncodingSubset@aux#1#2*#3\DeclareEncodingSubset@aux#4{%
```

if #3 is empty then there was no star, otherwise we define all four variants.

```
  \expandafter\ifx\expandafter X\detokenize{#3}X%
    \@DeclareEncodingSubset{#1}{#2}{#4}%
  \else
    \@DeclareEncodingSubset{#1}{#2LF}{#4}%
    \@DeclareEncodingSubset{#1}{#2TLF}{#4}%
    \@DeclareEncodingSubset{#1}{#2OsF}{#4}%
    \@DeclareEncodingSubset{#1}{#2TOsF}{#4}%
  \fi
}
```

The subset info is stored in a command with the name *family*:*subset* so if that already exists we change otherwise declare a subset.

```
\def\@DeclareEncodingSubset#1#2#3{%
   \@ifundefined{#1:#2}%
     {\@font@info{Setting #2 sub-encoding to #1/#3}}%
     {\@font@info{Changing #2 sub-encoding to #1/#3}}%
```

This declaration should be usable in .fd files and therefore has to make its definition globally, because such files can get loaded in random places.

```
     \global\@namedef{#1:#2}{#3}}
```

Any reason to allow those in the middle of documents?

```
\@onlypreamble\DeclareEncodingSubset
\@onlypreamble\DeclareEncodingSubset@aux
\@onlypreamble\@DeclareEncodingSubset
```

(*End of definition for* \DeclareEncodingSubset.)

\CheckEncodingSubset The command \CheckEncodingSubset will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either \UseTextSymbol, \UseTextAccent depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: #2 and #5 of \CheckEncodingSubset.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute #1{#2}#5 otherwise it runs #3#5, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions below.

```
\def\CheckEncodingSubset#1#2#3#4#5{%
    \ifnum #4>%
      \expandafter\ifx\csname #2:\f@family\endcsname\relax
        0\csname #2:?\endcsname
      \else
        \csname #2:\f@family\endcsname
      \fi
   \relax
   \expandafter\@firstoftwo
  \else
   \expandafter\@secondoftwo
 \fi
```

```
  {#1{#2}}{#3}%
  #5%
}
```

(*End of definition for* `\CheckEncodingSubset`.)

To set up the glyphs for the subsets we need a number helpers.

`\tc@errorwarn` To we produce errors, warnings, or only info in the transcripts if glyphs require substitutions? By default it is "info" only. With the `textcomp` package that can be changed.

```
\def\tc@errorwarn#1#2{\@latex@info{#1}}
```

(*End of definition for* `\tc@errorwarn`.)

`\tc@subst`

```
\def\tc@subst#1{%
   \tc@errorwarn
    {Symbol \string#1 not provided by\MessageBreak
     font family \f@family\space
     in TS1 encoding.\MessageBreak Default family used instead}\@eha
  \bgroup
      \expand@font@defaults
      \ifx\f@family\rmdef@ult
         \fontfamily\rmsubstdefault
      \else\ifx\f@family\sfdef@ult
         \fontfamily\sfsubstdefault
      \else\ifx\f@family\ttdef@ult
         \fontfamily\ttsubstdefault
      \else
         \fontfamily\textcompsubstdefault
      \fi\fi\fi
```

Whatever default was chosen, we claim now (locally hopefully) that it can handle all slots (even if not true) to avoid looping in certain situations, e.g., when something was set up incorrectly.

```
      \@namedef{TS1:\f@family}{0}%
      \selectfont#1%
  \egroup
}
```

(*End of definition for* `\tc@subst`.)

`\tc@fake@euro` `\tc@fake@euro` is an example of a "fake" definition to use in arg `#3` of the command `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce a poor man's Euro symbol by combining a "C" with a "=".

```
\def\tc@fake@euro#1{%
   \leavevmode
   \@font@info{Faking \noexpand#1for font family
                          \f@family\MessageBreak in TS1 encoding}%
   \valign{##\cr
      \vfil\hbox to 0.07em{\dimen@\f@size\p@
                           \math@fontsfalse
                           \fontsize{.7\dimen@}\z@\selectfont=\hss}%
      \vfil\cr%
      \hbox{C}\crcr
   }%
}
```

(*End of definition for* `\tc@fake@euro`.)

`\tc@check@symbol` `\tc@check@accent` These are two abbreviations that we use below to check symbols and accents in TS1. Only there to save some space, e.g., we can then write

```
DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}
```

to ensure that `\textcurrency` is only typeset if the current font has a `TS1` subset id of less than 3. Otherwise `\tc@error` is called telling the user that for this font family `\textcurrency` is not available.

```
\def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
```

Accents have been made an error in the `textcomp` package when not available. Now that we provide the functionality in the kernel we avoid the error by swapping in a `T1` accent if the `TS1` accent is not available.

```
%\def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}
\def\tc@check@accent#1{\CheckEncodingSubset\UseTextAccent
                                            {TS1}{\tc@swap@accent#1}}
\def\tc@swap@accent#1#2{\UseTextAccent{T1}#1}
```

(*End of definition for* `\tc@check@symbol` *and* `\tc@check@accent`.)

1 Sub-encodings

Here are the default definitions for the `TS1` symbols. First those that we assume are always available if a font implements `TS1`.

```
\DeclareTextSymbolDefault{\textdollar}{TS1}
\UndeclareTextCommand{\textdollar}{OT1}  % don't use the OT1 def any longer
\DeclareTextSymbolDefault{\textsterling}{TS1}
\UndeclareTextCommand{\textsterling}{OT1}% don't use the OT1 def any longer
\DeclareTextSymbolDefault{\textperthousand}{TS1}
\UndeclareTextCommand{\textperthousand}{T1} % don't use the T1 def
```

Using `\UndeclareTextCommand` above is enough only if the encoding definition files are not reloaded afterwards. In the past that happened if `fontenc` was used in the document preamble (not any longer). So in some sense it is better to fully remove them from the encoding files, but for rollbacks it is easier to keep them in for now.

These are the standard `itemize` and footnote symbols originally taken from `OMS` and now from `TS1`:

```
\DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
\DeclareTextSymbolDefault{\textbullet}{TS1}
\DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
\DeclareTextSymbolDefault{\textdagger}{TS1}
\DeclareTextSymbolDefault{\textparagraph}{TS1}
\DeclareTextSymbolDefault{\textperiodcentered}{TS1}
\DeclareTextSymbolDefault{\textsection}{TS1}
```

And here are the other `TS1` glyphs that are implemented by every font (or nearly every)—a few are commented out and moved to sub-encoding 9, because they aren't around in some fonts.

```
%%\DeclareTextSymbolDefault{\textbardbl}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\textbrokenbar}{TS1}
```

```
%%\DeclareTextSymbolDefault{\textcelsius}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\textcent}{TS1}
\DeclareTextSymbolDefault{\textcopyright}{TS1}
\DeclareTextSymbolDefault{\textdegree}{TS1}
\DeclareTextSymbolDefault{\textdiv}{TS1}
\DeclareTextSymbolDefault{\textlnot}{TS1}
\DeclareTextSymbolDefault{\textonehalf}{TS1}
\DeclareTextSymbolDefault{\textonequarter}{TS1}
%%\DeclareTextSymbolDefault{\textonesuperior}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\textordfeminine}{TS1}
\DeclareTextSymbolDefault{\textordmasculine}{TS1}
\DeclareTextSymbolDefault{\textpm}{TS1}
\DeclareTextSymbolDefault{\textquotesingle}{TS1}
\DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
\DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
\DeclareTextSymbolDefault{\textregistered}{TS1}
%%\DeclareTextSymbolDefault{\textthreequartersemdash}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\textthreequarters}{TS1}
%%\DeclareTextSymbolDefault{\textthreesuperior}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\texttimes}{TS1}
\DeclareTextSymbolDefault{\texttrademark}{TS1}
%%\DeclareTextSymbolDefault{\texttwelveudash}{TS1} % subst in sub-enc 9 above
%%\DeclareTextSymbolDefault{\texttwosuperior}{TS1} % subst in sub-enc 9 above
\DeclareTextSymbolDefault{\textyen}{TS1}
```

```
\DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
\DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}
```

In the following sections the remaining default definitions are ordered by sub-encoding in which they are become **unavailable**, i.e., they are not provided in the sub-encoding with that number and all sub-encodings with higher numbers.

Thus the symbols that are available in sub-encoding x are the symbols above (always available) and the symbols listed as becoming unavailable in sub-encodings $x + 1$ and higher.

1.1 Unavailable in sub-encoding 1 and higher (drop symbols not working in Latin Modern)

The `\textcircled` is available but the glyph is simply too small so we keep using the `OMS` glyph.

```
\DeclareTextCommandDefault{\textcircled}
  {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OMS}}1\textcircled}
```

1.2 Unavailable in sub-encoding 2 (majority of new OTF fonts via autoinst) and higher

```
\DeclareTextCommandDefault{\t}
  {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OML}}2\t}
```

Capital accents are really only very seldom implemented, so from sub-encoding `2` onwards we use the normal `T1` accents if they are asked for in the document.

In Unicode engines we don't implement them at all but always use the basic accents instead. whether that works or not really depends on the font, something like `\"X` usually comes out wrong in Unicode engines.

```
\ifx\Umathcode\@undefined
  \DeclareTextCommandDefault{\capitalacute}
                            {\tc@check@accent{\'}2\capitalacute}
  \DeclareTextCommandDefault{\capitalbreve}
                            {\tc@check@accent{\u}2\capitalbreve}
  \DeclareTextCommandDefault{\capitalcaron}
                            {\tc@check@accent{\v}2\capitalcaron}
  \DeclareTextCommandDefault{\capitalcedilla}
                            {\tc@check@accent{\c}2\capitalcedilla}
  \DeclareTextCommandDefault{\capitalcircumflex}
                            {\tc@check@accent{\^}2\capitalcircumflex}
  \DeclareTextCommandDefault{\capitaldieresis}
                            {\tc@check@accent{\"}2\capitaldieresis}
  \DeclareTextCommandDefault{\capitaldotaccent}
                            {\tc@check@accent{\.}2\capitaldotaccent}
  \DeclareTextCommandDefault{\capitalgrave}
                            {\tc@check@accent{\`}2\capitalgrave}
  \DeclareTextCommandDefault{\capitalhungarumlaut}
                            {\tc@check@accent{\H}2\capitalhungarumlaut}
  \DeclareTextCommandDefault{\capitalmacron}
                            {\tc@check@accent{\=}2\capitalmacron}
  \DeclareTextCommandDefault{\capitalogonek}
                            {\tc@check@accent{\k}2\capitalogonek}
  \DeclareTextCommandDefault{\capitalring}
                            {\tc@check@accent{\r}2\capitalring}
  \DeclareTextCommandDefault{\capitaltie}
                            {\tc@check@accent{\t}2\capitaltie}
  \DeclareTextCommandDefault{\capitaltilde}
                            {\tc@check@accent{\~}2\capitaltilde}
```

For `\newtie` and `\capitalnewtie` this is actually wrong, they should pick up the accent from the substitution font (not done yet).

```
  \DeclareTextCommandDefault{\newtie}
                            {\tc@check@accent{\t}2\newtie}
  \DeclareTextCommandDefault{\capitalnewtie}
                            {\tc@check@accent{\t}2\capitalnewtie}
```

In Unicode engines we just execute the simple accents:

```
\else
  \DeclareTextCommandDefault\capitalacute{\@tabacckludge'}
  \DeclareTextCommandDefault\capitalbreve{\u}
  \DeclareTextCommandDefault\capitalcaron{\v}
  \DeclareTextCommandDefault\capitalcedilla{\c}
  \DeclareTextCommandDefault\capitalcircumflex{\^}
  \DeclareTextCommandDefault\capitaldieresis{\"}
  \DeclareTextCommandDefault\capitaldotaccent{\.}
  \DeclareTextCommandDefault\capitalgrave{\@tabacckludge`}
  \DeclareTextCommandDefault\capitalhungarumlaut{\H}
  \DeclareTextCommandDefault\capitalmacron{\@tabacckludge=}
  \DeclareTextCommandDefault\capitalnewtie{\t}
  \DeclareTextCommandDefault\capitalogonek{\k}
  \DeclareTextCommandDefault\capitalring{\r}
  \DeclareTextCommandDefault\capitaltie{\t}
  \DeclareTextCommandDefault\capitaltilde{\~}
  \DeclareTextCommandDefault\newtie{\t}
```

```
\fi
```

The next two symbols exist in some fonts (faked?), but we ignore that to keep the subsets reasonable compact and most important linear.

```
\DeclareTextCommandDefault{\textlbrackdbl}
                          {\tc@check@symbol2\textlbrackdbl}
\DeclareTextCommandDefault{\textrbrackdbl}
                          {\tc@check@symbol2\textrbrackdbl}
```

Old style numerals are again in some fonts but using `-OsF`, etc. is the better approach to get them, so we claim they aren't in sub-encoding 2 as that's true for most fonts.

```
\DeclareTextCommandDefault{\texteightoldstyle}
                          {\tc@check@symbol2\texteightoldstyle}
\DeclareTextCommandDefault{\textfiveoldstyle}
                          {\tc@check@symbol2\textfiveoldstyle}
\DeclareTextCommandDefault{\textfouroldstyle}
                          {\tc@check@symbol2\textfouroldstyle}
\DeclareTextCommandDefault{\textnineoldstyle}
                          {\tc@check@symbol2\textnineoldstyle}
\DeclareTextCommandDefault{\textoneoldstyle}
                          {\tc@check@symbol2\textoneoldstyle}
\DeclareTextCommandDefault{\textsevenoldstyle}
                          {\tc@check@symbol2\textsevenoldstyle}
\DeclareTextCommandDefault{\textsixoldstyle}
                          {\tc@check@symbol2\textsixoldstyle}
\DeclareTextCommandDefault{\textthreeoldstyle}
                          {\tc@check@symbol2\textthreeoldstyle}
\DeclareTextCommandDefault{\texttwooldstyle}
                          {\tc@check@symbol2\texttwooldstyle}
\DeclareTextCommandDefault{\textzerooldstyle}
                          {\tc@check@symbol2\textzerooldstyle}
```

The next set of glyphs is special to TEX fonts (and available with a few older PS fonts supported through virtual fonts), but not any longer in the majority of fonts provided through autoinst, so we pretend there aren't available in sub-encoding 2 and below.

```
\DeclareTextCommandDefault{\textacutedbl}
                          {\tc@check@symbol2\textacutedbl}
\DeclareTextCommandDefault{\textasciiacute}
                          {\tc@check@symbol2\textasciiacute}
\DeclareTextCommandDefault{\textasciibreve}
                          {\tc@check@symbol2\textasciibreve}
\DeclareTextCommandDefault{\textasciicaron}
                          {\tc@check@symbol2\textasciicaron}
\DeclareTextCommandDefault{\textasciidieresis}
                          {\tc@check@symbol2\textasciidieresis}
\DeclareTextCommandDefault{\textasciigrave}
                          {\tc@check@symbol2\textasciigrave}
\DeclareTextCommandDefault{\textasciimacron}
                          {\tc@check@symbol2\textasciimacron}
\DeclareTextCommandDefault{\textgravedbl}
                          {\tc@check@symbol2\textgravedbl}
\DeclareTextCommandDefault{\texttildelow}
                          {\tc@check@symbol2\texttildelow}
```

Finally those below are only available in CM-based fonts but in no font that has its origin outside of the TEX world.

```
\DeclareTextCommandDefault{\textbaht}
                          {\tc@check@symbol2\textbaht}
\DeclareTextCommandDefault{\textbigcircle}
                          {\tc@check@symbol2\textbigcircle}
\DeclareTextCommandDefault{\textborn}
                          {\tc@check@symbol2\textborn}
\DeclareTextCommandDefault{\textcentoldstyle}
                          {\tc@check@symbol2\textcentoldstyle}
\DeclareTextCommandDefault{\textcircledP}
                          {\tc@check@symbol2\textcircledP}
\DeclareTextCommandDefault{\textcopyleft}
                          {\tc@check@symbol2\textcopyleft}
\DeclareTextCommandDefault{\textdblhyphenchar}
                          {\tc@check@symbol2\textdblhyphenchar}
\DeclareTextCommandDefault{\textdblhyphen}
                          {\tc@check@symbol2\textdblhyphen}
\DeclareTextCommandDefault{\textdied}
                          {\tc@check@symbol2\textdied}
\DeclareTextCommandDefault{\textdiscount}
                          {\tc@check@symbol2\textdiscount}
\DeclareTextCommandDefault{\textdivorced}
                          {\tc@check@symbol2\textdivorced}
\DeclareTextCommandDefault{\textdollaroldstyle}
                          {\tc@check@symbol2\textdollaroldstyle}
\DeclareTextCommandDefault{\textguarani}
                          {\tc@check@symbol2\textguarani}
\DeclareTextCommandDefault{\textleaf}
                          {\tc@check@symbol2\textleaf}
\DeclareTextCommandDefault{\textlquill}
                          {\tc@check@symbol2\textlquill}
\DeclareTextCommandDefault{\textmarried}
                          {\tc@check@symbol2\textmarried}
\DeclareTextCommandDefault{\textmho}
                          {\tc@check@symbol2\textmho}
\DeclareTextCommandDefault{\textmusicalnote}
                          {\tc@check@symbol2\textmusicalnote}
\DeclareTextCommandDefault{\textnaira}
                          {\tc@check@symbol2\textnaira}
\DeclareTextCommandDefault{\textopenbullet}
                          {\tc@check@symbol2\textopenbullet}
\DeclareTextCommandDefault{\textpeso}
                          {\tc@check@symbol2\textpeso}
\DeclareTextCommandDefault{\textpilcrow}
                          {\tc@check@symbol2\textpilcrow}
\DeclareTextCommandDefault{\textrecipe}
                          {\tc@check@symbol2\textrecipe}
\DeclareTextCommandDefault{\textreferencemark}
                          {\tc@check@symbol2\textreferencemark}
\DeclareTextCommandDefault{\textrquill}
                          {\tc@check@symbol2\textrquill}
\DeclareTextCommandDefault{\textservicemark}
                          {\tc@check@symbol2\textservicemark}
\DeclareTextCommandDefault{\textsurd}
                          {\tc@check@symbol2\textsurd}
```

The \textpertenthousand also belongs in this group but here we have a choice: in T1 there is a definition for \textpertenthousand making the symbol up from % and \char 24 (twice) but in many fonts that char doesn't exist and the slot is reused for random ligatures. So better not use it because often it is wrong. But pointing to TS1 is also not great as only a few fonts have it as a real symbol, so we get a substitution to CM or LM.

Alternatively we could just state that the symbol is unavailable in those fonts. For now I substitute.

```
\DeclareTextCommandDefault{\textpertenthousand}
                          {\tc@check@symbol2\textpertenthousand}
\UndeclareTextCommand{\textpertenthousand}{T1}
```

1.3 Unavailable in sub-encoding 3 and higher

Sub-encoding 2 is the one where we loose many symbols. In the higher-numbered sub-encodings we see only a few dropped additionally.

```
\DeclareTextCommandDefault{\textlangle}
                          {\tc@check@symbol3\textlangle}
\DeclareTextCommandDefault{\textrangle}
                          {\tc@check@symbol3\textrangle}
```

1.4 Unavailable in sub-encoding 4 and higher

```
\DeclareTextCommandDefault{\textcolonmonetary}
                          {\tc@check@symbol4\textcolonmonetary}
\DeclareTextCommandDefault{\textdong}
                          {\tc@check@symbol4\textdong}
\DeclareTextCommandDefault{\textdownarrow}
                          {\tc@check@symbol4\textdownarrow}
\DeclareTextCommandDefault{\textleftarrow}
                          {\tc@check@symbol4\textleftarrow}
\DeclareTextCommandDefault{\textlira}
                          {\tc@check@symbol4\textlira}
\DeclareTextCommandDefault{\textrightarrow}
                          {\tc@check@symbol4\textrightarrow}
\DeclareTextCommandDefault{\textuparrow}
                          {\tc@check@symbol4\textuparrow}
\DeclareTextCommandDefault{\textwon}
                          {\tc@check@symbol4\textwon}
```

1.5 Unavailable in sub-encoding 5 (most older PS fonts) and higher

Most older PS fonts (supported in TEX since the early nineties when virtual fonts became available) are sorted under this sub-encoding. But in reality, many of them don't have all glyphs that should be available in sub-encoding 5. Instead they show little squares, i.e., they produce "tofu" if you are unlucky.

But the coverage is so random that it is impossible to sort them properly and if we tried to ensure that they only typeset those glyphs that are really always available, we would have to put them all into sub-encoding 9; so putting them into 5 is really a compromise.

Modern fonts usually don't typeset a tofu character if a glyph is missing. They are therefore only classified as sub-encoding 5 if they really support its glyph set completely.

```
\DeclareTextCommandDefault{\textestimated}
                          {\tc@check@symbol5\textestimated}
\DeclareTextCommandDefault{\textnumero}
                          {\tc@check@symbol5\textnumero}
```

1.6 Unavailable in sub-encoding 6 and higher

```
\DeclareTextCommandDefault{\textflorin}
                          {\tc@check@symbol6\textflorin}
\DeclareTextCommandDefault{\textcurrency}
                          {\tc@check@symbol6\textcurrency}
```

1.7 Unavailable in sub-encoding 7 and higher

```
\DeclareTextCommandDefault{\textfractionsolidus}
                          {\tc@check@symbol7\textfractionsolidus}
\DeclareTextCommandDefault{\textohm}
                          {\tc@check@symbol7\textohm}
\DeclareTextCommandDefault{\textmu}
                          {\tc@check@symbol7\textmu}
\DeclareTextCommandDefault{\textminus}
                          {\tc@check@symbol7\textminus}
```

1.8 Unavailable in sub-encoding 8 and higher

```
\DeclareTextCommandDefault{\textblank}
                          {\tc@check@symbol{8}\textblank}
\DeclareTextCommandDefault{\textinterrobangdown}
                          {\tc@check@symbol{8}\textinterrobangdown}
\DeclareTextCommandDefault{\textinterrobang}
                          {\tc@check@symbol{8}\textinterrobang}
```

Fonts with this sub-encoding don't have a Euro symbol, but instead of substituting we fake it.

```
\DeclareTextCommandDefault{\texteuro}
  {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro{8}\texteuro}
```

1.9 Unavailable in Sub-encoding 9 (most missing)

```
\DeclareTextCommandDefault{\textcelsius}
                          {\tc@check@symbol{9}\textcelsius}
\DeclareTextCommandDefault{\textonesuperior}
                          {\tc@check@symbol{9}\textonesuperior}
\DeclareTextCommandDefault{\textthreequartersemdash}
                          {\tc@check@symbol{9}\textthreequartersemdash}
\DeclareTextCommandDefault{\textthreesuperior}
                          {\tc@check@symbol{9}\textthreesuperior}
\DeclareTextCommandDefault{\texttwelveudash}
                          {\tc@check@symbol{9}\texttwelveudash}
\DeclareTextCommandDefault{\texttwosuperior}
                          {\tc@check@symbol{9}\texttwosuperior}
\DeclareTextCommandDefault{\textbardbl}
                          {\tc@check@symbol{9}\textbardbl}
```

2 Unicode engine specials

If we are using a unicode engine we handle some glyphs differently, so this here are the definitions for the Unicode encoding (overwriting the defaults above).

```
\ifx \Umathcode\@undefined  \else
```

This set should be taken from TS1 encoding even if it means you get it from the default font for that encoding.

```
%\DeclareTextSymbol{\textcopyleft}{TS1}{171}
%\DeclareTextSymbol{\textdblhyphen}{TS1}{45}
%\DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
%\DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
%\DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
%\DeclareTextSymbol{\textleaf}{TS1}{108}
%\DeclareTextSymbol{\texttwelveudash}{TS1}{21}
%\DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}
```

If oldstyle numerals are asked for we just use `\oldstylenums`.

```
\DeclareTextCommand{\textzerooldstyle} \UnicodeEncodingName{\oldstylenums{0}}
\DeclareTextCommand{\textoneoldstyle}  \UnicodeEncodingName{\oldstylenums{1}}
\DeclareTextCommand{\texttwooldstyle}  \UnicodeEncodingName{\oldstylenums{2}}
\DeclareTextCommand{\textthreeoldstyle}\UnicodeEncodingName{\oldstylenums{3}}
\DeclareTextCommand{\textfouroldstyle} \UnicodeEncodingName{\oldstylenums{4}}
\DeclareTextCommand{\textfiveoldstyle} \UnicodeEncodingName{\oldstylenums{5}}
\DeclareTextCommand{\textsixoldstyle}  \UnicodeEncodingName{\oldstylenums{6}}
\DeclareTextCommand{\textsevenoldstyle}\UnicodeEncodingName{\oldstylenums{7}}
\DeclareTextCommand{\texteightoldstyle}\UnicodeEncodingName{\oldstylenums{8}}
\DeclareTextCommand{\textnineoldstyle} \UnicodeEncodingName{\oldstylenums{9}}
```

These have Unicode slots so this should be integrated into TU explicitly

```
\DeclareTextSymbol{\textpilcrow}        \UnicodeEncodingName{"00B6}
\DeclareTextSymbol{\textborn}           \UnicodeEncodingName{"002A}
\DeclareTextSymbol{\textdied}           \UnicodeEncodingName{"2020}
\DeclareTextSymbol{\textlbrackdbl}      \UnicodeEncodingName{"27E6}
\DeclareTextSymbol{\textrbrackdbl}      \UnicodeEncodingName{"27E7}
\DeclareTextSymbol{\textguarani}        \UnicodeEncodingName{"20B2}
```

We could make `\textcentoldstyle` and `\textdollaroldstyle` point to dollar and cent in the Unicode encoding

```
%\DeclareTextSymbol{\textcentoldstyle}  \UnicodeEncodingName{"00A2}
%\DeclareTextSymbol{\textdollaroldstyle}\UnicodeEncodingName{"0024}
```

but I think it is better to pick them up from TS1 even if that usually means LMR fonts

```
\DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
\DeclareTextSymbol{\textcentoldstyle}  {TS1}{139}
```

```
\fi               % --- END of Unicode engines specials
```

3 Font family sub-encodings setup

We declare the subsets for a good number of fonts in the kernel ...

But first the default for anything that is not declared. We use 9 which is most likely much too conservative, but with the advantage that we aren't getting missing glyphs (or at least that this is very unlikely). For nearly all font in the TEX Live distribution of

2019 “correct” classifications are given below, so that this default is only used for new font families, and over time the right classifications can be added here too.

```
\DeclareEncodingSubset{TS1}{?}{9}
```

This first block contains the fonts that have been already supported by the `textcomp` package way back, i.e., the font families that have TEX support since the mid-nineties.

```
\DeclareEncodingSubset{TS1}{ccr}     {0}
\DeclareEncodingSubset{TS1}{cmbr}    {0}
\DeclareEncodingSubset{TS1}{cmr}     {0}
\DeclareEncodingSubset{TS1}{cmss}    {0}
\DeclareEncodingSubset{TS1}{cmtl}    {0}
\DeclareEncodingSubset{TS1}{cmtt}    {0}
\DeclareEncodingSubset{TS1}{cmvtt}   {0}
\DeclareEncodingSubset{TS1}{pxr}     {0}
\DeclareEncodingSubset{TS1}{pxss}    {0}
\DeclareEncodingSubset{TS1}{pxtt}    {0}
\DeclareEncodingSubset{TS1}{qag}     {0}
\DeclareEncodingSubset{TS1}{qbk}     {0}
\DeclareEncodingSubset{TS1}{qcr}     {0}
\DeclareEncodingSubset{TS1}{qcs}     {0}
\DeclareEncodingSubset{TS1}{qhvc}    {0}
\DeclareEncodingSubset{TS1}{qhv}     {0}
\DeclareEncodingSubset{TS1}{qpl}     {0}
\DeclareEncodingSubset{TS1}{qtm}     {0}
\DeclareEncodingSubset{TS1}{qzc}     {0}
\DeclareEncodingSubset{TS1}{txr}     {0}
\DeclareEncodingSubset{TS1}{txss}    {0}
\DeclareEncodingSubset{TS1}{txtt}    {0}

\DeclareEncodingSubset{TS1}{lmr}     {1}
\DeclareEncodingSubset{TS1}{lmdh}    {1}
\DeclareEncodingSubset{TS1}{lmss}    {1}
\DeclareEncodingSubset{TS1}{lmssq}   {1}
\DeclareEncodingSubset{TS1}{lmvtt}   {1}
```

The lmtt family is missing TM, SM, and perthousand for some reason, so the first safe sub-encoding would be 2, but that is then missing out a huge number of glyphs that are available, so we claim it is sub-encoding 1 even if this can lead to missing glyphs.

```
\DeclareEncodingSubset{TS1}{lmtt}    {1} % missing TM, SM and pertenthousand
```

The next three families have been removed from TeX Live, but we keep the definitions

```
\DeclareEncodingSubset{TS1}{ptmx}    {2}
\DeclareEncodingSubset{TS1}{ptmj}    {2}
\DeclareEncodingSubset{TS1}{ul8}     {2}
```

The next set are the early PostScript font implementations, these days there are better alternatives, but .... Note that, their virtual fonts contain a lot of “tofu” in form of black squares, thus they don’t even give a missing character warning if you select such a glyph. This is why they are set as sub-encoding 5.

```
\DeclareEncodingSubset{TS1}{bch} {5} % tofu for blank, ohm
\DeclareEncodingSubset{TS1}{futj}{5} % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{futs}{5} % tofu for blank, ohm
\DeclareEncodingSubset{TS1}{futx}{5} % probably (currently broken distrib)
\DeclareEncodingSubset{TS1}{pag} {5} % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{pbk} {5} % tofu for blank, interrobang/down, ohm
```

```
\DeclareEncodingSubset{TS1}{pcr} {5} % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{phv} {5} % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{pnc} {5} % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{pplj}{5} % tofu for blank
\DeclareEncodingSubset{TS1}{pplx}{5} % tofu for blank
\DeclareEncodingSubset{TS1}{ppl} {5} % tofu for blank interrobang/down
\DeclareEncodingSubset{TS1}{ptm} {5} % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{pzc} {5} % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{ul9} {5} % tofu for blank, interrobang/down, ohm
```

The next set suffers from the same problem and they contain even fewer real glyphs.

```
\DeclareEncodingSubset{TS1}{dayroms}{6} % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{dayrom} {6} % tofu for blank, interrobang/down, ohm
\DeclareEncodingSubset{TS1}{augie}{8} % really only missing euro
\DeclareEncodingSubset{TS1}{put}  {8}
\DeclareEncodingSubset{TS1}{uag}  {8} % probably (currently broken distrib)
\DeclareEncodingSubset{TS1}{ugq}  {8}
\DeclareEncodingSubset{TS1}{zi4}     {9}
```

LucidaBright (sold through TUG) probably not quite correct, I guess as I have the older fonts ...

```
\DeclareEncodingSubset{TS1}{hls}     {5}
\DeclareEncodingSubset{TS1}{hlst}    {5}
\DeclareEncodingSubset{TS1}{hlct}    {5}
\DeclareEncodingSubset{TS1}{hlh}     {5}
\DeclareEncodingSubset{TS1}{hlx}     {8}
\DeclareEncodingSubset{TS1}{hlce}    {8}
\DeclareEncodingSubset{TS1}{hlcn}    {8}
\DeclareEncodingSubset{TS1}{hlcw}    {8}
\DeclareEncodingSubset{TS1}{hlcf}    {8}
```

Below are the newer fonts that have support files for LaTeX. With very few exceptions the classifications are done so that all characters are correctly produced (either being available in the font or substituted).

There are a few fonts that contain "tofu" squares in places (instead of a real glyph) and in a few cases some really seldom needed chars are unavailable, i.e., produce missing glyphs (to avoid that a large number of available chars are unnecessarily substituted).

Encoding declarations for these font families shouldn't really be in the kernel, but part of the `.fd` files for the family. When we introduced the concept in 2021 we had some hope that this would happen over time and that we could take the declarations out—after all it is nearly impossible to maintain it correctly in the kernel, given that fonts may get new glyphs added (happened for several of them in the recent year) which is something we wouldn't notice. However, so far this hasn't happened, so in 2024, I went through the current set and adjusted the declarations in several places.

Next four are wrong and still need adjustment:

```
\DeclareEncodingSubset{TS1}{lato-*}       {0}  % with a bunch of tofu inside
\DeclareEncodingSubset{TS1}{opensans-*}   {0}  % with a bunch of tofu inside
\DeclareEncodingSubset{TS1}{cantarell-*}  {0}  % with a bunch of tofu inside
\DeclareEncodingSubset{TS1}{tli}          {1}  % with lots of tofu inside
\DeclareEncodingSubset{TS1}{fbb-*}        {2}  % missing centoldstyle
```

```
\DeclareEncodingSubset{TS1}{Alegreya-*}               {2}
\DeclareEncodingSubset{TS1}{AlegreyaSans-*}           {2}
\DeclareEncodingSubset{TS1}{BaskervilleF-*}           {2}
\DeclareEncodingSubset{TS1}{DejaVuSans-TLF}           {2}
\DeclareEncodingSubset{TS1}{DejaVuSansCondensed-TLF}  {2}
```

Next one is missing `\textfractionsolidus` but is otherwise completely sub-encoding 2 so we use that sub-encoding.

```
\DeclareEncodingSubset{TS1}{DejaVuSansMono-TLF}       {2}
```

```
\DeclareEncodingSubset{TS1}{EBGaramond-*}             {2}
\DeclareEncodingSubset{TS1}{Merriwthr-OsF}            {2}
\DeclareEncodingSubset{TS1}{MerriwthrSans-OsF}        {2}
\DeclareEncodingSubset{TS1}{Montserrat-*}             {2}
\DeclareEncodingSubset{TS1}{MontserratAlternates-*}   {2}
\DeclareEncodingSubset{TS1}{NotoSansMono-TLF}         {2}
\DeclareEncodingSubset{TS1}{NotoSansMono-TOsF}        {2}
\DeclareEncodingSubset{TS1}{Tempora-TLF}              {2}
\DeclareEncodingSubset{TS1}{Tempora-TOsF}             {2}
\DeclareEncodingSubset{TS1}{XCharter-TLF}             {2}
\DeclareEncodingSubset{TS1}{XCharter-TOsF}            {2}
\DeclareEncodingSubset{TS1}{erewhon-*}                {2}
```

```
\DeclareEncodingSubset{TS1}{Arimo-TLF}                {3}
\DeclareEncodingSubset{TS1}{Crlt-*}                   {3}
\DeclareEncodingSubset{TS1}{IBMPlexMono-TLF}          {3}
\DeclareEncodingSubset{TS1}{IBMPlexSans-TLF}          {3}
\DeclareEncodingSubset{TS1}{IBMPlexSerif-TLF}         {3}
\DeclareEncodingSubset{TS1}{SourceCodePro-TLF}        {3}
\DeclareEncodingSubset{TS1}{SourceCodePro-TOsF}       {3}
\DeclareEncodingSubset{TS1}{SourceSansPro-*}          {3}
\DeclareEncodingSubset{TS1}{SourceSerifPro-*}         {3}
\DeclareEncodingSubset{TS1}{Tinos-TLF}                {3}
```

```
\DeclareEncodingSubset{TS1}{AccanthisADFStdNoThree-LF}{4}
\DeclareEncodingSubset{TS1}{Cabin-TLF}                {4}
\DeclareEncodingSubset{TS1}{Caladea-TLF}              {4}
\DeclareEncodingSubset{TS1}{Chivo-*}                  {4}
\DeclareEncodingSubset{TS1}{ClearSans-TLF}            {4}
\DeclareEncodingSubset{TS1}{Coelacanth-LF}            {4}
\DeclareEncodingSubset{TS1}{CrimsonPro-*}             {4}
\DeclareEncodingSubset{TS1}{FiraMono-TLF}             {4}
\DeclareEncodingSubset{TS1}{FiraMono-TOsF}            {4}
\DeclareEncodingSubset{TS1}{FiraSans-*}               {4}
\DeclareEncodingSubset{TS1}{Go-TLF}                   {4}
\DeclareEncodingSubset{TS1}{GoMono-TLF}               {4}
\DeclareEncodingSubset{TS1}{InriaSans-*}              {4}
\DeclareEncodingSubset{TS1}{InriaSerif-*}             {4}
\DeclareEncodingSubset{TS1}{LibertinusSans-*}         {4}
\DeclareEncodingSubset{TS1}{LibertinusSerif-*}        {4}
\DeclareEncodingSubset{TS1}{LibreBodoni-TLF}          {4}
\DeclareEncodingSubset{TS1}{LibreFranklin-TLF}        {4}
\DeclareEncodingSubset{TS1}{LinguisticsPro-LF}        {4}
\DeclareEncodingSubset{TS1}{LinguisticsPro-OsF}       {4}
\DeclareEncodingSubset{TS1}{LinuxBiolinumT-*}         {4}
\DeclareEncodingSubset{TS1}{LinuxLibertineT-*}        {4}
```

```
\DeclareEncodingSubset{TS1}{MintSpirit-*}             {4}
\DeclareEncodingSubset{TS1}{MintSpiritNoTwo-*}        {4}
\DeclareEncodingSubset{TS1}{PTMono-TLF}               {4}
\DeclareEncodingSubset{TS1}{PTSans-TLF}               {4}
\DeclareEncodingSubset{TS1}{PTSansCaption-TLF}        {4}
\DeclareEncodingSubset{TS1}{PTSansNarrow-TLF}         {4}
\DeclareEncodingSubset{TS1}{PTSerif-TLF}              {4}
\DeclareEncodingSubset{TS1}{PTSerifCaption-TLF}       {4}
\DeclareEncodingSubset{TS1}{Raleway-TLF}              {4}
\DeclareEncodingSubset{TS1}{Raleway-TOsF}             {4}
\DeclareEncodingSubset{TS1}{Roboto-*}                 {4}
\DeclareEncodingSubset{TS1}{RobotoMono-TLF}           {4}
\DeclareEncodingSubset{TS1}{RobotoSlab-TLF}           {4}
\DeclareEncodingSubset{TS1}{Rosario-*}                {4}
\DeclareEncodingSubset{TS1}{SticksTooText-*}          {4}
\DeclareEncodingSubset{TS1}{UniversalisADFStd-LF}     {4}

\DeclareEncodingSubset{TS1}{Almndr-OsF}               {5}
\DeclareEncodingSubset{TS1}{Baskervaldx-*}            {5}
\DeclareEncodingSubset{TS1}{Bttr-TLF}                 {5}
\DeclareEncodingSubset{TS1}{Cinzel-LF}                {5}
\DeclareEncodingSubset{TS1}{CinzelDecorative-LF}      {5}
\DeclareEncodingSubset{TS1}{Cochineal-*}              {5}
\DeclareEncodingSubset{TS1}{DejaVuSerif-TLF}          {5}
\DeclareEncodingSubset{TS1}{DejaVuSerifCondensed-TLF} {5}
\DeclareEncodingSubset{TS1}{GilliusADF-LF}            {5}
\DeclareEncodingSubset{TS1}{GilliusADFCond-LF}        {5}
\DeclareEncodingSubset{TS1}{GilliusADFNoTwo-LF}       {5}
\DeclareEncodingSubset{TS1}{GilliusADFNoTwoCond-LF}   {5}
\DeclareEncodingSubset{TS1}{OldStandard-TLF}          {5}
\DeclareEncodingSubset{TS1}{PlyfrDisplay-TLF}         {5}
\DeclareEncodingSubset{TS1}{PlyfrDisplay-TOsF}        {5}
\DeclareEncodingSubset{TS1}{TheanoDidot-TLF}          {5}
\DeclareEncodingSubset{TS1}{TheanoDidot-TOsF}         {5}
\DeclareEncodingSubset{TS1}{TheanoModern-TLF}         {5}
\DeclareEncodingSubset{TS1}{TheanoModern-TOsF}        {5}
\DeclareEncodingSubset{TS1}{TheanoOldStyle-TLF}       {5}
\DeclareEncodingSubset{TS1}{TheanoOldStyle-TOsF}      {5}
\DeclareEncodingSubset{TS1}{charssil-TLF}             {5}

\DeclareEncodingSubset{TS1}{Crimson-TLF}              {6}
\DeclareEncodingSubset{TS1}{LibertinusSerifDisplay-LF}{6}
\DeclareEncodingSubset{TS1}{LinuxLibertineDisplayT-*} {6}
\DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-LF}   {6}
\DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-TLF}  {6}
\DeclareEncodingSubset{TS1}{Ovrlck-LF}                {6}

\DeclareEncodingSubset{TS1}{ComicNeue-TLF}            {7}
\DeclareEncodingSubset{TS1}{ComicNeueAngular-TLF}     {7}
\DeclareEncodingSubset{TS1}{CormorantGaramond-*}      {7}
\DeclareEncodingSubset{TS1}{Heuristica-TLF}           {7}
\DeclareEncodingSubset{TS1}{Heuristica-TOsF}          {7}
\DeclareEncodingSubset{TS1}{IMFELLEnglish-TLF}        {7}
\DeclareEncodingSubset{TS1}{LibreBskrvl-TLF}          {7}
\DeclareEncodingSubset{TS1}{LibreCsln-*}              {7}
\DeclareEncodingSubset{TS1}{Lbstr-LF}                 {7}
```

```
\DeclareEncodingSubset{TS1}{Mrcls-LF}                {7}
```

Strangely enough NotoSerif and NotoSans are sub-encoding 7 as they are missing `\textminus` and several other glyphs. In contrast the NotoSansMono is far more complete.

```
\DeclareEncodingSubset{TS1}{NotoSans-*}              {7}
\DeclareEncodingSubset{TS1}{NotoSerif-*}             {7}
\DeclareEncodingSubset{TS1}{Quattro-LF}              {7}
\DeclareEncodingSubset{TS1}{QuattroSans-LF}          {7}
\DeclareEncodingSubset{TS1}{Frm-LF}                  {7}  % the superiors are missing
\DeclareEncodingSubset{TS1}{LibertinusMono-TLF}      {8}
\DeclareEncodingSubset{TS1}{AlgolRevived-TLF}        {9}
```

4 Legacy symbol support for lists and footnote symbols

`\UseLegacyTextSymbols`

```
\def\UseLegacyTextSymbols{%
  \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}%
  \DeclareTextSymbolDefault{\textbardbl}{OMS}%
  \DeclareTextSymbolDefault{\textbullet}{OMS}%
  \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}%
  \DeclareTextSymbolDefault{\textdagger}{OMS}%
  \DeclareTextSymbolDefault{\textparagraph}{OMS}%
  \DeclareTextSymbolDefault{\textperiodcentered}{OMS}%
  \DeclareTextSymbolDefault{\textsection}{OMS}%
  \UndeclareTextCommand{\textsection}{T1}%
  \expandafter\let\csname oldstylenums \expandafter\endcsname
                  \csname legacyoldstylenums \endcsname
}
```

(*End of definition for* `\UseLegacyTextSymbols`.)

`\textlegacyasteriskcentered`
`\textlegacybardbl`
`\textlegacybullet`
`\textlegacydaggerdbl`
`\textlegacydagger`
`\textlegacyparagraph`
`\textlegacyperiodcentered`
`\textlegacysection`

Here are new names for the legacy symbols that LaTeX used to pick up from the `OMS` encoded fonts (and used for itemize lists or footnote symbols).

We go the roundabout way via separate OMS declarations so that

```
\renewcommand\textbullet{\textlegacybullet}
```

doesn't produce an endless loop.

```
\DeclareTextSymbol{\textlegacyasteriskcentered}{OMS}{3}   % "03
\DeclareTextSymbol{\textlegacybardbl}{OMS}{107}           % "6B
\DeclareTextSymbol{\textlegacybullet}{OMS}{15}            % "0F
\DeclareTextSymbol{\textlegacydaggerdbl}{OMS}{122}        % "7A
\DeclareTextSymbol{\textlegacydagger}{OMS}{121}           % "79
\DeclareTextSymbol{\textlegacyparagraph}{OMS}{123}        % "7B
\DeclareTextSymbol{\textlegacyperiodcentered}{OMS}{1}     % "01
\DeclareTextSymbol{\textlegacysection}{OMS}{120}          % "78
```

```
\DeclareTextSymbolDefault{\textlegacyasteriskcentered}{OMS}
\DeclareTextSymbolDefault{\textlegacybardbl}{OMS}
\DeclareTextSymbolDefault{\textlegacybullet}{OMS}
\DeclareTextSymbolDefault{\textlegacydaggerdbl}{OMS}
\DeclareTextSymbolDefault{\textlegacydagger}{OMS}
\DeclareTextSymbolDefault{\textlegacyparagraph}{OMS}
\DeclareTextSymbolDefault{\textlegacyperiodcentered}{OMS}
\DeclareTextSymbolDefault{\textlegacysection}{OMS}
```

(*End of definition for* `\textlegacyasteriskcentered` *and others.*)

Supporting rollback ...

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩     {lttextcomp}{Undefine text companion symbols}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\oldstylenums[1]{%
⟨latexrelease⟩   \begingroup
⟨latexrelease⟩    \spaceskip\fontdimen\tw@\font
⟨latexrelease⟩    \usefont{OML}{\rmdefault}{\f@series}{it}%
⟨latexrelease⟩    \mathgroup\symletters #1%
⟨latexrelease⟩   \endgroup
⟨latexrelease⟩}
⟨latexrelease⟩\let\legacyoldstylenums\@undefined
⟨latexrelease⟩\def\textcompsubstdefault{cmr}
⟨latexrelease⟩
⟨latexrelease⟩\let\DeclareEncodingSubset\@undefined
⟨latexrelease⟩\let\CheckEncodingSubset\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareTextSymbolDefault{\textdollar}{OT1}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textsterling}{OT1}
⟨latexrelease⟩\DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
⟨latexrelease⟩   \ifdim \fontdimen\@ne\font >\z@
⟨latexrelease⟩      \slshape
⟨latexrelease⟩   \else
⟨latexrelease⟩      \upshape
⟨latexrelease⟩   \fi
⟨latexrelease⟩   \char`\$\egroup}
⟨latexrelease⟩\DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
⟨latexrelease⟩   \ifdim \fontdimen\@ne\font >\z@
⟨latexrelease⟩      \itshape
⟨latexrelease⟩   \else
⟨latexrelease⟩      \fontshape{ui}\selectfont
⟨latexrelease⟩   \fi
⟨latexrelease⟩   \char`\$\egroup}
⟨latexrelease⟩\DeclareTextCommand{\textperthousand}{T1}
⟨latexrelease⟩   {\%\char 24 }
⟨latexrelease⟩
⟨latexrelease⟩\DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textbullet}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textdagger}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textparagraph}{OMS}
⟨latexrelease⟩\DeclareTextSymbolDefault{\textperiodcentered}{OMS}
```

```
⟨latexrelease⟩\DeclareTextSymbolDefault{\textsection}{OMS}
⟨latexrelease⟩
⟨latexrelease⟩\DeclareTextSymbolDefault{\textbardbl}{OMS}
⟨latexrelease⟩\let\textbrokenbar\@undefined
⟨latexrelease⟩\let\textcelsius\@undefined
⟨latexrelease⟩\let\textcent\@undefined
⟨latexrelease⟩\DeclareTextCommandDefault{\textcopyright}
⟨latexrelease⟩                          {\textcircled{c}}
⟨latexrelease⟩\let\textdegree\@undefined
⟨latexrelease⟩\let\textdiv\@undefined
⟨latexrelease⟩\let\textlnot\@undefined
⟨latexrelease⟩\let\textonehalf\@undefined
⟨latexrelease⟩\let\textonequarter\@undefined
⟨latexrelease⟩\let\textonesuperior\@undefined
⟨latexrelease⟩\DeclareTextCommandDefault{\textordfeminine}
⟨latexrelease⟩                          {\textsuperscript{a}}
⟨latexrelease⟩\DeclareTextCommandDefault{\textordmasculine}
⟨latexrelease⟩                          {\textsuperscript{o}}
⟨latexrelease⟩\let\textpm\@undefined
⟨latexrelease⟩\let\textquotesingle\@undefined
⟨latexrelease⟩\let\textquotestraightbase\@undefined
⟨latexrelease⟩\let\textquotestraightdblbase\@undefined
⟨latexrelease⟩\DeclareTextCommandDefault{\textregistered}
⟨latexrelease⟩    {\textcircled{%
⟨latexrelease⟩        \check@mathfonts\fontsize\sf@size\z@
⟨latexrelease⟩        \math@fontsfalse\selectfont R}}
⟨latexrelease⟩\let\textthreequartersemdash\@undefined
⟨latexrelease⟩\let\textthreequarters\@undefined
⟨latexrelease⟩\let\textthreesuperior\@undefined
⟨latexrelease⟩\let\texttimes\@undefined
⟨latexrelease⟩\DeclareTextCommandDefault{\texttrademark}
⟨latexrelease⟩                          {\textsuperscript{TM}}
⟨latexrelease⟩\let\texttwelveudash\@undefined
⟨latexrelease⟩\let\texttwosuperior\@undefined
⟨latexrelease⟩\let\textyen\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textcapitalcompwordmark\@undefined
⟨latexrelease⟩\let\textascendercompwordmark\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareTextAccentDefault{\textcircled}{OMS}
⟨latexrelease⟩\DeclareTextAccentDefault{\t}{OML}
⟨latexrelease⟩
⟨latexrelease⟩\let\capitalacute\@undefined
⟨latexrelease⟩\let\capitalbreve\@undefined
⟨latexrelease⟩\let\capitalcaron\@undefined
⟨latexrelease⟩\let\capitalcedilla\@undefined
⟨latexrelease⟩\let\capitalcircumflex\@undefined
⟨latexrelease⟩\let\capitaldieresis\@undefined
⟨latexrelease⟩\let\capitaldotaccent\@undefined
⟨latexrelease⟩\let\capitalgrave\@undefined
⟨latexrelease⟩\let\capitalhungarumlaut\@undefined
⟨latexrelease⟩\let\capitalmacron\@undefined
⟨latexrelease⟩\let\capitalnewtie\@undefined
⟨latexrelease⟩\let\capitalogonek\@undefined
```

```
⟨latexrelease⟩\let\capitalring\@undefined
⟨latexrelease⟩\let\capitaltie\@undefined
⟨latexrelease⟩\let\capitaltilde\@undefined
⟨latexrelease⟩\let\newtie\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textlbrackdbl\@undefined
⟨latexrelease⟩\let\textrbrackdbl\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\texteightoldstyle\@undefined
⟨latexrelease⟩\let\textfiveoldstyle\@undefined
⟨latexrelease⟩\let\textfouroldstyle\@undefined
⟨latexrelease⟩\let\textnineoldstyle\@undefined
⟨latexrelease⟩\let\textoneoldstyle\@undefined
⟨latexrelease⟩\let\textsevenoldstyle\@undefined
⟨latexrelease⟩\let\textsixoldstyle\@undefined
⟨latexrelease⟩\let\textthreeoldstyle\@undefined
⟨latexrelease⟩\let\texttwooldstyle\@undefined
⟨latexrelease⟩\let\textzerooldstyle\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textacutedbl\@undefined
⟨latexrelease⟩\let\textasciiacute\@undefined
⟨latexrelease⟩\let\textasciibreve\@undefined
⟨latexrelease⟩\let\textasciicaron\@undefined
⟨latexrelease⟩\let\textasciidieresis\@undefined
⟨latexrelease⟩\let\textasciigrave\@undefined
⟨latexrelease⟩\let\textasciimacron\@undefined
⟨latexrelease⟩\let\textgravedbl\@undefined
⟨latexrelease⟩\let\texttildelow\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textbaht\@undefined
⟨latexrelease⟩\let\textbigcircle\@undefined
⟨latexrelease⟩\let\textborn\@undefined
⟨latexrelease⟩\let\textcentoldstyle\@undefined
⟨latexrelease⟩\let\textcircledP\@undefined
⟨latexrelease⟩\let\textcopyleft\@undefined
⟨latexrelease⟩\let\textdblhyphenchar\@undefined
⟨latexrelease⟩\let\textdblhyphen\@undefined
⟨latexrelease⟩\let\textdied\@undefined
⟨latexrelease⟩\let\textdiscount\@undefined
⟨latexrelease⟩\let\textdivorced\@undefined
⟨latexrelease⟩\let\textdollaroldstyle\@undefined
⟨latexrelease⟩\let\textguarani\@undefined
⟨latexrelease⟩\let\textleaf\@undefined
⟨latexrelease⟩\let\textlquill\@undefined
⟨latexrelease⟩\let\textmarried\@undefined
⟨latexrelease⟩\let\textmho\@undefined
⟨latexrelease⟩\let\textmusicalnote\@undefined
⟨latexrelease⟩\let\textnaira\@undefined
⟨latexrelease⟩\let\textopenbullet\@undefined
⟨latexrelease⟩\let\textpeso\@undefined
⟨latexrelease⟩\let\textpilcrow\@undefined
⟨latexrelease⟩\let\textrecipe\@undefined
⟨latexrelease⟩\let\textreferencemark\@undefined
⟨latexrelease⟩\let\textrquill\@undefined
```

```
⟨latexrelease⟩\let\textservicemark\@undefined
⟨latexrelease⟩\let\textsurd\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareTextCommand{\textpertenthousand}{T1}
⟨latexrelease⟩                   {\%\char 24\char 24 }
⟨latexrelease⟩
⟨latexrelease⟩\let\textlangle\@undefined
⟨latexrelease⟩\let\textrangle\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textcolonmonetary\@undefined
⟨latexrelease⟩\let\textdong\@undefined
⟨latexrelease⟩\let\textdownarrow\@undefined
⟨latexrelease⟩\let\textleftarrow\@undefined
⟨latexrelease⟩\let\textlira\@undefined
⟨latexrelease⟩\let\textrightarrow\@undefined
⟨latexrelease⟩\let\textuparrow\@undefined
⟨latexrelease⟩\let\textwon\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textestimated\@undefined
⟨latexrelease⟩\let\textnumero\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textflorin\@undefined
⟨latexrelease⟩\let\textcurrency\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textfractionsolidus\@undefined
⟨latexrelease⟩\let\textohm\@undefined
⟨latexrelease⟩\let\textmu\@undefined
⟨latexrelease⟩\let\textminus\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textblank\@undefined
⟨latexrelease⟩\let\textinterrobangdown\@undefined
⟨latexrelease⟩\let\textinterrobang\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\texteuro\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\textcelsius\@undefined
⟨latexrelease⟩\let\textonesuperior\@undefined
⟨latexrelease⟩\let\textthreequartersemdash\@undefined
⟨latexrelease⟩\let\textthreesuperior\@undefined
⟨latexrelease⟩\let\texttwelveudash\@undefined
⟨latexrelease⟩\let\texttwosuperior\@undefined
⟨latexrelease⟩\let\textbardbl\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\UseLegacyTextSymbols\@undefined
⟨latexrelease⟩\let\textlegacyasteriskcentered\@undefined
⟨latexrelease⟩\let\textlegacybardbl\@undefined
⟨latexrelease⟩\let\textlegacybullet\@undefined
⟨latexrelease⟩\let\textlegacydaggerdbl\@undefined
⟨latexrelease⟩\let\textlegacydagger\@undefined
⟨latexrelease⟩\let\textlegacyparagraph\@undefined
⟨latexrelease⟩\let\textlegacyperiodcentered\@undefined
⟨latexrelease⟩\let\textlegacysection\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndModuleRelease
```

5 The textcomp package

For any rollback request before 2018-08-11 we make an attempt by loading the 2018 version.

```
⟨*TS1sty⟩
\DeclareRelease{}{1997-12-01}{textcomp-2018-08-11.sty}
\DeclareRelease{}{2018-08-11}{textcomp-2018-08-11.sty}
\DeclareCurrentRelease{}{2020-02-02}

\ProvidesPackage{textcomp}
 [2024/04/24 v2.1b Standard LaTeX package]
```

A precaution in case this is used without rebuilding the format.

```
\NeedsTeXFormat{LaTeX2e}[2020/02/02]
```

This is implemented by defining the default subset:

```
\DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
\DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
\DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{8}}
\DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{9}}
```

The default is set up in the kernel is "safe" these days for unknown fonts but LaTeX has definitions for most families so it seldom applies.

If a different default is used then one needs to check the results to ensure that there aren't "missing glyphs".

The next set of options define the warning level (default in the kernel is info only). Using the package options you can change this behavior.

```
\DeclareOption{error}
              {\gdef\tc@errorwarn{\PackageError{textcomp}}}
\DeclareOption{warn}
              {\gdef\tc@errorwarn#1#2{\PackageWarning{textcomp}{#1}}}
\DeclareOption{info}
              {\gdef\tc@errorwarn#1#2{\PackageInfo{textcomp}{#1}}}
\DeclareOption{quiet}{\gdef\tc@errorwarn#1#2{}}
```

The "force" option basically changes the sub-encoding to that of the default (which, unless changes, is 9 these days), i.e., it no longer depends on the font in use. This is mainly there because it might have been used in older documents, but not something that is recommended.

```
\DeclareOption{force}{%
    \def\CheckEncodingSubset#1#2#3#4#5{%
      \ifnum #4>%
           0\csname #2:?\endcsname
           \relax
      \expandafter\@firstoftwo
     \else
      \expandafter\@secondoftwo
    \fi
     {#1{#2}}{#3}%
     #5}%
}

\ProcessOptions\relax
```

There is not much else to do nowadays, because everything is already set up in the LaTeX kernel.

```
\InputIfFileExists{textcomp.cfg}
 {\PackageInfo{textcomp}{Local configuration file used}}{}
```

⟨/TS1sty⟩

5.1 The old textcomp package code

This section contains the old code for the textcomp package and its documentation. It is only used if we roll back prior to 2020. Thus all the rest is mainly for historians. Note that the old code categorized in the sub-encodings only into 6 classes not 10.

```
⟨*TS1oldsty⟩
\ProvidesPackage{textcomp}
   [2018/08/11 v2.0j Standard LaTeX package]
```

This one is for the `TS1` encoding which contains text symbols for use with the `T1`-encoded text fonts. It therefore first inputs the file `TS1enc.def` and then sets (or resets) the defaults for the symbols it contains. The result of this is that when one of these symbols is accessed and the current encoding does not provide it, the symbol will be supplied by a silent, local change to this encoding.

Since many PostScript fonts only implement a subset of `TS1` many commands only produce black blobs of ink. To resolve the resulting problems a number of options have been introduced and some code has been developed to distinguish sub-encodings.

The sub-encodings have a numerical id and are defined as follows for `TS1`:

#5 those `TS1` symbols that are also in the ISO-Adobe character set; without `textcurrency`, which is often misused for the Euro. Older Type1 fonts from the non-TeX world provide only this subset.

#4 = #5 + `\texteuro`. Most newer fonts provide this.

#3 = #4 + `\textomega`. Can also be described as `TS1` ∩ (`ISO-Adobe` ∪ `MacRoman`). (Except for the missing "currency".)

#2 = #3 + `\textestimated` + `\textcurrency`. Can also be described as `TS1` ∩ `Adobe-Western-2`. This may be relevant for OpenType fonts, which usually show the Adobe-Western-2 character set.

#1 = `TS1` without `\textcircled` and `\t`. These two glyphs are often not implemented and if their kernel defaults are changed commands like `\copyright` unnecessarily fail.

#0 = full `TS1`

And here a summary to go in the transcript file:

```
\PackageInfo{textcomp}{Sub-encoding information:\MessageBreak
    \space\space 5 = only ISO-Adobe without
                              \string\textcurrency\MessageBreak
    \space\space 4 = 5 + \string\texteuro\MessageBreak
    \space\space 3 = 4 + \string\textohm\MessageBreak
    \space\space 2 = 3 + \noexpand\textestimated+
                                \string\textcurrency\MessageBreak
    \space\space 1 = TS1 - \noexpand\textcircled-
```

```
                                        \string\t\MessageBreak
    \space\space 0 = TS1 (full)\MessageBreak
    Font families with sub-encoding setting implement\MessageBreak
    only a restricted character set as indicated.\MessageBreak
    Family '?' is the default used for unknown fonts.\MessageBreak
    See the documentation for details\@gobble}
```

`\DeclareEncodingSubset` An encoding subset to which a font family belongs is declared by the command `\DeclareEncodingSubset` that takes the major encoding as the first argument (e.g., `TS1`), the family name as the second argument (e.g., `cmr`), and the subset encoding id as a third, (e.g., `0` for `cmr`).

The default encoding subset to use when nothing is known about the current font family is named `?`.

```
\def\DeclareEncodingSubset#1#2#3{%
   \@ifundefined{#1:#2}%
     {\PackageInfo{textcomp}{Setting #2 sub-encoding to #1/#3}}%
     {\PackageInfo{textcomp}{Changing #2 sub-encoding to #1/#3}}%
   \@namedef{#1:#2}{#3}}
\@onlypreamble\DeclareEncodingSubset
```

(*End of definition for* `\DeclareEncodingSubset`.)

The options for the package are the following:

safe for unknown font families enables only symbols that are also in the ISO-Adobe character set; without "currency", which is often misused for the Euro. Older Type1 fonts from the non-TeX world provide only this subset.

euro enables the "safe" symbols plus the `\texteuro` command. Most newer fonts provide this.

full enables all `TS1` commands; useful only with fonts like EC or CM bright.

almostfull same as "full", except that `\textcircled` and `\t` are *not* redefined from their defaults to avoid that commands like `\copyright` suddenly no longer work.

force ignore all subset encoding definitions stored in the package itself or in the configuration file and always use the default subset as specified by one of the other options (seldom useful, only dangerous).

`\iftc@forced` Switch used to implement the `force` option

```
\newif\iftc@forced   \tc@forcedfalse
```

(*End of definition for* `\iftc@forced`.)

This is implemented by defining the default subset:

```
\DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
\DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
\DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{4}}
\DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{5}}
```

The default is "almostfull" which means that old documents will work except that `\textcircled` and `\t` will use the kernel defaults (with the advantage that this also works if the current font, as often the case, doesn't implement these glyphs).

The "force" option simply sets the switch to true.

```
\DeclareOption{force}{\tc@forcedtrue}
```

The suggestions to user is to use the "safe" option always unless that balks in which case they could switch to "almostfull" but then better check their output manually.

```
\def\tc@errorwarn{\PackageError}
\DeclareOption{warn}{\gdef\tc@errorwarn#1#2#3{\PackageWarning{#1}{#2}}}
\DeclareOption{quiet}{\gdef\tc@errorwarn#1#2#3{}}
```

```
\ExecuteOptions{almostfull}
\ProcessOptions\relax
```

`\CheckEncodingSubset` The command `\CheckEncodingSubset` will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either `\UseTextSymbol`, `\UseTextAccent` depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: `#2` and `#5` of `\CheckEncodingSubset`.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute `#1{#2}#5` otherwise it runs `#3#5`, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions below.

```
\iftc@forced
```

If the "force" option was given we always use the default for testing against.

```
\def\CheckEncodingSubset#1#2#3#4#5{%
    \ifnum #4>%
        0\csname #2:?\endcsname
        \relax
   \expandafter\@firstoftwo
  \else
   \expandafter\@secondoftwo
 \fi
  {#1{#2}}{#3}%
  #5%
}
```

In normal circumstances the test is a bit more complicated: first check if there exists a macro `\`⟨*arg2*⟩`:`⟨*current-family*⟩ and if so use that value to test against, otherwise use the default to test against.

```
\else
\def\CheckEncodingSubset#1#2#3#4#5{%
    \ifnum #4>%
      \expandafter\ifx\csname #2:\f@family\endcsname\relax
        0\csname #2:?\endcsname
      \else
        \csname #2:\f@family\endcsname
      \fi
   \relax
   \expandafter\@firstoftwo
  \else
   \expandafter\@secondoftwo
 \fi
```

```
  {#1{#2}}{#3}%
  #5%
}
\fi
```

(*End of definition for* `\CheckEncodingSubset`.)

`\tc@subst`

```
\def\tc@subst#1{%
   \tc@errorwarn{textcomp}%
    {Symbol \string#1 not provided by\MessageBreak
     font family \f@family\space
     in TS1 encoding.\MessageBreak Default family used instead}\@eha
  \bgroup\fontfamily\textcompsubstdefault\selectfont#1\egroup
}
```

(*End of definition for* `\tc@subst`.)

`\tc@error` `\tc@error` is going to be used in arg `#3` of `\CheckEncodingSubset` when a symbol is not available in a certain font family. It gets pass the encoding it normally lives in (arg one) and the name of the symbol or accent that has a problem.

```
% error commands take argument:
% #1 symbol to be used
\def\tc@error#1{%
   \PackageError{textcomp}%  % should be latex error if general
    {Accent \string#1 not provided by\MessageBreak
     font family \f@family\space
     in TS1 encoding}\@eha
}
```

(*End of definition for* `\tc@error`.)

`\tc@fake@euro` `\tc@fake@euro` is an example of a "fake" definition to use in arg `#3` of `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce an Euro symbol by combining a "C" with a "=".

```
\def\tc@fake@euro#1{%
   \leavevmode
   \PackageInfo{textcomp}{Faking \noexpand#1for font family
                          \f@family\MessageBreak in TS1 encoding}%
   \valign{##\cr
      \vfil\hbox to 0.07em{\dimen@\f@size\p@
                           \math@fontsfalse
                           \fontsize{.7\dimen@}\z@\selectfont=\hss}%
      \vfil\cr%
      \hbox{C}\crcr
   }%
}
```

(*End of definition for* `\tc@fake@euro`.)

`\tc@check@symbol` `\tc@check@accent` These are two abbreviations that we use below to check symbols and accents in TS1. Only there to save some space, e.g., we can then write

```
DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}
```

to ensure that \textcurrency is only typeset if the current font has a TS1 subset id of less than 3. Otherwise \tc@error is called telling the user that for this font family \textcurrency is not available.

```
\def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
\def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}
```

(*End of definition for* \tc@check@symbol *and* \tc@check@accent.)

We start with the commands that are "safe" and which can be unconditionally set up, first the accents...

```
\DeclareTextAccentDefault{\capitalcedilla}{TS1}
\DeclareTextAccentDefault{\capitalogonek}{TS1}
\DeclareTextAccentDefault{\capitalgrave}{TS1}
\DeclareTextAccentDefault{\capitalacute}{TS1}
\DeclareTextAccentDefault{\capitalcircumflex}{TS1}
\DeclareTextAccentDefault{\capitaltilde}{TS1}
\DeclareTextAccentDefault{\capitaldieresis}{TS1}
\DeclareTextAccentDefault{\capitalhungarumlaut}{TS1}
\DeclareTextAccentDefault{\capitalring}{TS1}
\DeclareTextAccentDefault{\capitalcaron}{TS1}
\DeclareTextAccentDefault{\capitalbreve}{TS1}
\DeclareTextAccentDefault{\capitalmacron}{TS1}
\DeclareTextAccentDefault{\capitaldotaccent}{TS1}
```

...and then the other glyphs.

```
\DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
\DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}
\DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
\DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
\DeclareTextSymbolDefault{\texttwelveudash}{TS1}
\DeclareTextSymbolDefault{\textthreequartersemdash}{TS1}
\DeclareTextSymbolDefault{\textdollar}{TS1}
\DeclareTextSymbolDefault{\textquotesingle}{TS1}
\DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
\DeclareTextSymbolDefault{\textfractionsolidus}{TS1}
\DeclareTextSymbolDefault{\textminus}{TS1}
\DeclareTextSymbolDefault{\textlbrackdbl}{TS1}
\DeclareTextSymbolDefault{\textrbrackdbl}{TS1}
\DeclareTextSymbolDefault{\textasciigrave}{TS1}
\DeclareTextSymbolDefault{\texttildelow}{TS1}
\DeclareTextSymbolDefault{\textasciibreve}{TS1}
\DeclareTextSymbolDefault{\textasciicaron}{TS1}
\DeclareTextSymbolDefault{\textgravedbl}{TS1}
\DeclareTextSymbolDefault{\textacutedbl}{TS1}
\DeclareTextSymbolDefault{\textdagger}{TS1}
\DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
\DeclareTextSymbolDefault{\textbardbl}{TS1}
\DeclareTextSymbolDefault{\textperthousand}{TS1}
\DeclareTextSymbolDefault{\textbullet}{TS1}
\DeclareTextSymbolDefault{\textcelsius}{TS1}
\DeclareTextSymbolDefault{\textflorin}{TS1}
\DeclareTextSymbolDefault{\texttrademark}{TS1}
\DeclareTextSymbolDefault{\textcent}{TS1}
\DeclareTextSymbolDefault{\textsterling}{TS1}
\DeclareTextSymbolDefault{\textyen}{TS1}
```

```
\DeclareTextSymbolDefault{\textbrokenbar}{TS1}
\DeclareTextSymbolDefault{\textsection}{TS1}
\DeclareTextSymbolDefault{\textasciidieresis}{TS1}
\DeclareTextSymbolDefault{\textcopyright}{TS1}
\DeclareTextSymbolDefault{\textordfeminine}{TS1}
\DeclareTextSymbolDefault{\textlnot}{TS1}
\DeclareTextSymbolDefault{\textregistered}{TS1}
\DeclareTextSymbolDefault{\textasciimacron}{TS1}
\DeclareTextSymbolDefault{\textdegree}{TS1}
\DeclareTextSymbolDefault{\textpm}{TS1}
\DeclareTextSymbolDefault{\texttwosuperior}{TS1}
\DeclareTextSymbolDefault{\textthreesuperior}{TS1}
\DeclareTextSymbolDefault{\textasciiacute}{TS1}
\DeclareTextSymbolDefault{\textmu}{TS1}
\DeclareTextSymbolDefault{\textparagraph}{TS1}
\DeclareTextSymbolDefault{\textperiodcentered}{TS1}
\DeclareTextSymbolDefault{\textonesuperior}{TS1}
\DeclareTextSymbolDefault{\textordmasculine}{TS1}
\DeclareTextSymbolDefault{\textonequarter}{TS1}
\DeclareTextSymbolDefault{\textonehalf}{TS1}
\DeclareTextSymbolDefault{\textthreequarters}{TS1}
\DeclareTextSymbolDefault{\texttimes}{TS1}
\DeclareTextSymbolDefault{\textdiv}{TS1}
```

The `\texteuro` is only available for subsets with id 4 or less. Otherwise we fake the glyph using `\tc@fake@euro`

```
\DeclareTextCommandDefault{\texteuro}
   {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro5\texteuro}
```

The `\textohm` is only available for subsets with id 3 or less. Otherwise we produce an error.

```
\DeclareTextCommandDefault{\textohm}{\tc@check@symbol4\textohm}
```

The `\textestimated` and `\textcurrency` are only provided for fonts with subset encoding with id 2 or less.

```
\DeclareTextCommandDefault{\textestimated}%
    {\tc@check@symbol3\textestimated}
\DeclareTextCommandDefault{\textcurrency}%
    {\tc@check@symbol3\textcurrency}
```

Nearly all of the remaining glyphs are provided only with fonts with id 1 or 0, i.e., are essentially complete.

```
\DeclareTextCommandDefault{\capitaltie}%
    {\tc@check@accent2\capitaltie}
\DeclareTextCommandDefault{\newtie}%
    {\tc@check@accent2\newtie}
\DeclareTextCommandDefault{\capitalnewtie}%
    {\tc@check@accent2\capitalnewtie}
\DeclareTextCommandDefault{\textleftarrow}%
    {\tc@check@symbol2\textleftarrow}
\DeclareTextCommandDefault{\textrightarrow}%
    {\tc@check@symbol2\textrightarrow}
\DeclareTextCommandDefault{\textblank}%
    {\tc@check@symbol2\textblank}
\DeclareTextCommandDefault{\textdblhyphen}%
```

```
    {\tc@check@symbol2\textdblhyphen}
\DeclareTextCommandDefault{\textzerooldstyle}%
    {\tc@check@symbol2\textzerooldstyle}
\DeclareTextCommandDefault{\textoneoldstyle}%
    {\tc@check@symbol2\textoneoldstyle}
\DeclareTextCommandDefault{\texttwooldstyle}%
    {\tc@check@symbol2\texttwooldstyle}
\DeclareTextCommandDefault{\textthreeoldstyle}%
    {\tc@check@symbol2\textthreeoldstyle}
\DeclareTextCommandDefault{\textfouroldstyle}%
    {\tc@check@symbol2\textfouroldstyle}
\DeclareTextCommandDefault{\textfiveoldstyle}%
    {\tc@check@symbol2\textfiveoldstyle}
\DeclareTextCommandDefault{\textsixoldstyle}%
    {\tc@check@symbol2\textsixoldstyle}
\DeclareTextCommandDefault{\textsevenoldstyle}%
    {\tc@check@symbol2\textsevenoldstyle}
\DeclareTextCommandDefault{\texteightoldstyle}%
    {\tc@check@symbol2\texteightoldstyle}
\DeclareTextCommandDefault{\textnineoldstyle}%
    {\tc@check@symbol2\textnineoldstyle}
\DeclareTextCommandDefault{\textlangle}%
    {\tc@check@symbol2\textlangle}
\DeclareTextCommandDefault{\textrangle}%
    {\tc@check@symbol2\textrangle}
\DeclareTextCommandDefault{\textmho}%
    {\tc@check@symbol2\textmho}
\DeclareTextCommandDefault{\textbigcircle}%
    {\tc@check@symbol2\textbigcircle}
\DeclareTextCommandDefault{\textuparrow}%
    {\tc@check@symbol2\textuparrow}
\DeclareTextCommandDefault{\textdownarrow}%
    {\tc@check@symbol2\textdownarrow}
\DeclareTextCommandDefault{\textborn}%
    {\tc@check@symbol2\textborn}
\DeclareTextCommandDefault{\textdivorced}%
    {\tc@check@symbol2\textdivorced}
\DeclareTextCommandDefault{\textdied}%
    {\tc@check@symbol2\textdied}
\DeclareTextCommandDefault{\textleaf}%
    {\tc@check@symbol2\textleaf}
\DeclareTextCommandDefault{\textmarried}%
    {\tc@check@symbol2\textmarried}
\DeclareTextCommandDefault{\textmusicalnote}%
    {\tc@check@symbol2\textmusicalnote}
\DeclareTextCommandDefault{\textdblhyphenchar}%
    {\tc@check@symbol2\textdblhyphenchar}
\DeclareTextCommandDefault{\textdollaroldstyle}%
    {\tc@check@symbol2\textdollaroldstyle}
\DeclareTextCommandDefault{\textcentoldstyle}%
    {\tc@check@symbol2\textcentoldstyle}
\DeclareTextCommandDefault{\textcolonmonetary}%
    {\tc@check@symbol2\textcolonmonetary}
\DeclareTextCommandDefault{\textwon}%
```

```
    {\tc@check@symbol2\textwon}
\DeclareTextCommandDefault{\textnaira}%
    {\tc@check@symbol2\textnaira}
\DeclareTextCommandDefault{\textguarani}%
    {\tc@check@symbol2\textguarani}
\DeclareTextCommandDefault{\textpeso}%
    {\tc@check@symbol2\textpeso}
\DeclareTextCommandDefault{\textlira}%
    {\tc@check@symbol2\textlira}
\DeclareTextCommandDefault{\textrecipe}%
    {\tc@check@symbol2\textrecipe}
\DeclareTextCommandDefault{\textinterrobang}%
    {\tc@check@symbol2\textinterrobang}
\DeclareTextCommandDefault{\textinterrobangdown}%
    {\tc@check@symbol2\textinterrobangdown}
\DeclareTextCommandDefault{\textdong}%
    {\tc@check@symbol2\textdong}
\DeclareTextCommandDefault{\textpertenthousand}%
    {\tc@check@symbol2\textpertenthousand}
\DeclareTextCommandDefault{\textpilcrow}%
    {\tc@check@symbol2\textpilcrow}
\DeclareTextCommandDefault{\textbaht}%
    {\tc@check@symbol2\textbaht}
\DeclareTextCommandDefault{\textnumero}%
    {\tc@check@symbol2\textnumero}
\DeclareTextCommandDefault{\textdiscount}%
    {\tc@check@symbol2\textdiscount}
\DeclareTextCommandDefault{\textopenbullet}%
    {\tc@check@symbol2\textopenbullet}
\DeclareTextCommandDefault{\textservicemark}%
    {\tc@check@symbol2\textservicemark}
\DeclareTextCommandDefault{\textlquill}%
    {\tc@check@symbol2\textlquill}
\DeclareTextCommandDefault{\textrquill}%
    {\tc@check@symbol2\textrquill}
\DeclareTextCommandDefault{\textcopyleft}%
    {\tc@check@symbol2\textcopyleft}
\DeclareTextCommandDefault{\textcircledP}%
    {\tc@check@symbol2\textcircledP}
\DeclareTextCommandDefault{\textreferencemark}%
    {\tc@check@symbol2\textreferencemark}
\DeclareTextCommandDefault{\textsurd}%
    {\tc@check@symbol2\textsurd}
```

The `\textcircled` and `\t` are handled specially, unless the current font has a subset id of 0 (i.e. full `TS1`) we pick the symbols up from the math font encodings, i.e., the third argument to `\CheckEncodingSubset` uses `\UseTextAccent` to get them from there.

```
\DeclareTextCommandDefault{\textcircled}
  {\CheckEncodingSubset\UseTextAccent{TS1}%
    {\UseTextAccent{OMS}}1\textcircled}
\DeclareTextCommandDefault{\t}
  {\CheckEncodingSubset\UseTextAccent{TS1}%
    {\UseTextAccent{OML}}1\t}
```

Finally input the encoding-specific definitions for `TS1` thus making the top-level

definitions optimized for this encoding (and not for the default encoding).

```
\input{ts1enc.def}
```

Now having the new glyphs available we also want to make sure that they are used. For most cases this will automatically happen but for some glyphs there are inferior definitions already known to LaTeX which will prevent the usage of the `TS1` versions. So we better get rid of them:

```
\UndeclareTextCommand{\textsterling}{OT1}
\UndeclareTextCommand{\textdollar}  {OT1}
```

Similar declarations should probably be made for other encodings like `OT4` if they are in use.

```
%\UndeclareTextCommand{\textsterling}{OT4}
%\UndeclareTextCommand{\textdollar}  {OT4}
```

From the `T1` encoding there are two candidates for removal: ‰ and ‰ since these are both constructed from % followed by a tiny '₀' rather than being a single glyph. The problem with this approach is that in PostScript fonts this small zero is usually not available resulting in %▪ rather than ‰ while the real glyph (at least for `\textperthousand`) is available in the PostScript version of `TS1`. So for the moment we compromise by removing the `T1` declaration for `\textperthousand` but keeping the one for `\textpertenthousand`. This will have the effect that with Computer Modern fonts everything will come out (although ‰ and ‰ are not taken from the same physical font) and with PostScript fonts ‰ will come out correctly while ‰ will most likely look like %▬ — which is probably an improvement over just getting a single '▪' to indicate a completely missing glyph, which would happen if we also 'undeclared' `\textpertenthousand`.

```
\UndeclareTextCommand{\textperthousand}{T1}
%\UndeclareTextCommand{\textpertenthousand}{T1}
```

5.1.1 Supporting oldstyle digits

```
\DeclareRobustCommand\oldstylenums[1]{%
 \begingroup
  \ifmmode
   \mathgroup\symletters #1%
  \else
   \CheckEncodingSubset\@use@text@encoding{TS1}%
       {\PackageWarning{textcomp}%
          {Oldstyle digits unavailable for
           family \f@family.\MessageBreak
           Lining digits used instead}}%
       \tw@{#1}%
   \fi
 \endgroup
}
```

5.1.2 Subset encoding defaults

For many font families commonly used in the TeX world we provide the subset encoding data here. Users can add additional font families in the file `textcomp.cfg` if they own other fonts.

However, if the option "forced" was given then all subset encoding specifications are ignored, so there is no point in setting any of them up:

```
\iftc@forced \else
```

Computer modern based fonts (e.g., CM, CM-Bright, Concrete):

```
\DeclareEncodingSubset{TS1}{cmr}     {0}
\DeclareEncodingSubset{TS1}{cmss}    {0}
\DeclareEncodingSubset{TS1}{cmtt}    {0}
\DeclareEncodingSubset{TS1}{cmvtt}   {0}
\DeclareEncodingSubset{TS1}{cmbr}    {0}
\DeclareEncodingSubset{TS1}{cmtl}    {0}
\DeclareEncodingSubset{TS1}{ccr}     {0}
```

PSNFSS fonts:

```
\DeclareEncodingSubset{TS1}{ptm}     {4}
\DeclareEncodingSubset{TS1}{pcr}     {4}
\DeclareEncodingSubset{TS1}{phv}     {4}
\DeclareEncodingSubset{TS1}{ppl}     {3}
\DeclareEncodingSubset{TS1}{pag}     {4}
\DeclareEncodingSubset{TS1}{pbk}     {4}
\DeclareEncodingSubset{TS1}{pnc}     {4}
\DeclareEncodingSubset{TS1}{pzc}     {4}
\DeclareEncodingSubset{TS1}{bch}     {4}
\DeclareEncodingSubset{TS1}{put}     {5}
```

Other CTAN fonts (probably not complete):

```
\DeclareEncodingSubset{TS1}{uag}     {5}
\DeclareEncodingSubset{TS1}{ugq}     {5}
\DeclareEncodingSubset{TS1}{ul8}     {4}
\DeclareEncodingSubset{TS1}{ul9}     {4}  % (LuxiSans, one day)
\DeclareEncodingSubset{TS1}{augie}   {5}
\DeclareEncodingSubset{TS1}{dayrom}  {3}
\DeclareEncodingSubset{TS1}{dayroms} {3}
\DeclareEncodingSubset{TS1}{pxr}     {0}
\DeclareEncodingSubset{TS1}{pxss}    {0}
\DeclareEncodingSubset{TS1}{pxtt}    {0}
\DeclareEncodingSubset{TS1}{txr}     {0}
\DeclareEncodingSubset{TS1}{txss}    {0}
\DeclareEncodingSubset{TS1}{txtt}    {0}
```

Latin Modern and TeX Gyre:

```
\DeclareEncodingSubset{TS1}{lmr}     {0}
\DeclareEncodingSubset{TS1}{lmdh}    {0}
\DeclareEncodingSubset{TS1}{lmss}    {0}
\DeclareEncodingSubset{TS1}{lmssq}   {0}
\DeclareEncodingSubset{TS1}{lmvtt}   {0}
\DeclareEncodingSubset{TS1}{lmtt}    {0}

\DeclareEncodingSubset{TS1}{qhv}     {0}
\DeclareEncodingSubset{TS1}{qag}     {0}
\DeclareEncodingSubset{TS1}{qbk}     {0}
\DeclareEncodingSubset{TS1}{qcr}     {0}
\DeclareEncodingSubset{TS1}{qcs}     {0}
\DeclareEncodingSubset{TS1}{qpl}     {0}
\DeclareEncodingSubset{TS1}{qtm}     {0}
\DeclareEncodingSubset{TS1}{qzc}     {0}
\DeclareEncodingSubset{TS1}{qhvc}    {0}
```

Fourier-GUTenberg:

```
\DeclareEncodingSubset{TS1}{futs}    {4}
```

```
\DeclareEncodingSubset{TS1}{futx}    {4}
\DeclareEncodingSubset{TS1}{futj}    {4}
```

Y&Y's Lucida Bright

```
\DeclareEncodingSubset{TS1}{hlh}     {3}
\DeclareEncodingSubset{TS1}{hls}     {3}
\DeclareEncodingSubset{TS1}{hlst}    {3}
```

The remaining settings for Lucida are conservative: the following fonts contain the `\textohm` character but not the `\texteuro`, i.e., belong to neither subset 4 nor subset 3. If you want to use the `\textohm` with these fonts copy these definition to `textcomp.cfg` and change the subset to 3. However in that case make sure that you do not use the `\texteuro`.

```
\DeclareEncodingSubset{TS1}{hlct}    {5}
\DeclareEncodingSubset{TS1}{hlx}     {5}
\DeclareEncodingSubset{TS1}{hlce}    {5}
\DeclareEncodingSubset{TS1}{hlcn}    {5}
\DeclareEncodingSubset{TS1}{hlcw}    {5}
\DeclareEncodingSubset{TS1}{hlcf}    {5}
```

Other commercial families...

```
\DeclareEncodingSubset{TS1}{pplx}    {3}
\DeclareEncodingSubset{TS1}{pplj}    {3}
\DeclareEncodingSubset{TS1}{ptmx}    {4}
\DeclareEncodingSubset{TS1}{ptmj}    {4}
```

If the file `textcomp.cfg` exists it will be loaded at this point. This allows to define further subset encodings for font families not covered by default.

```
\InputIfFileExists{textcomp.cfg}
  {\PackageInfo{textcomp}{Local configuration file used}}{}
\fi
```

⟨/TS1oldsty⟩

6 The `checkencodingsubset.tex` file

This is a simple file that asks for a name of a font family and then displays information about the TS1 encoding for this family and recommends the right encoding subset (to be used with `\DeclareEncodingsubset`) for this family.

```
⟨*TS1check⟩
\ProvidesFile{checkencodingsubset.tex}
 [2024/10/18 v0.5b Figure out safe TS1 encoding subsets]
\let\typeoutdetails\typeout
%\def\typeoutdetails#1{}     % alternative definition used below
```

For the purpose of this check a glyph exists if the font slot is occupied—too bad if that contains the wrong glyph or some tofu. If it "exists" we return 0 otherwise 1. This way we can call this macro several times in a row and obtain a number that is 0 if all glyphs are existing or greater than 0 if any of them is missing.

The second argument (holding the command name for a symbol) is not used during these tests.

```
\def\doesglyphexist#1#2{\iffontchar\testFont #1 0\else 1\relax \fi}
```

This macro also tests and outputs some information about the symbol if it is missing. This time we make use of the second argument.

```
\def\glyphmissingdetails#1#2{\iffontchar\testFont #1 \else
  \typeoutdetails{\space\space\space ==> \string#2 (#1) is missing}\fi}
```

```
\newif\ifsafesubencodingfound
\newif\ifcoremisses
```

Testing a group of symbols that belong to one sub-encoding. More precisely, the symbols that become unavailable if you change from sub-encoding x (#2) to $x+1$ (#3). As far as the code is concerned, the symbols that are supposed to be always available (the core) become available if we test the group -1 and 0.

The first argument contains the testing code and is supposed to return a single number greater or equal to zero.

```
\def\testgroup#1#2#3{%
  \ifnum 0 = #1%
    \ifnum #2<0
    \typeoutdetails{All glyphs in core exist}%
    \else
    \typeoutdetails{All glyphs between sub-encoding #2 and #3  exist}%
    \fi
  \else
    \ifnum #2<0
      \typeoutdetails{**********************************}%
      \typeoutdetails{Some glyphs are missing from core:}%
      \coremissestrue
      \ifsafesubencodingfound \else
        \def\subencodingresult{#2}%
      \fi
    \else
      \typeoutdetails{Some glyphs are missing from sub-encoding #2:}%
      \ifsafesubencodingfound \else
        \def\subencodingresult{#3}%
      \fi
    \fi
```

If some glyphs are missing, we rerun the test code but this time using \glyphmissingdetails.

```
    {\let\doesglyphexist \glyphmissingdetails #1}%
```

And because we had misses we have definitely found the subset.

```
    \safesubencodingfoundtrue
  \fi
}
```

The currently defined subset for the family is either stored in \TS1:⟨*family*⟩ if it was declared, or it is the default subset which is stored in \TS1:?.

```
\def\currsubencoding#1{\csname TS1:\ifcsname TS1:#1\endcsname #1\else ?\fi\endcsname}
```

If a font family is not found when declaring it with \DeclareFixedFont we end up with the following font. This can then be used as a simple test if we failed loading the TS1 font.

```
\DeclareFixedFont\cmrFont{TS1}{cmr}{m}{n}{10pt}
```

Check for all glyphs in all encoding subsets ...

```
\def\testallgroups#1{%
  \DeclareFixedFont\testFont{TS1}{#1}{m}{n}{10pt}%
  \ifx\testFont\cmrFont
    \typeout{***** Font family #1 not found ****}%
  \else
```

We haven't checked anything yet.

```
    \safesubencodingfoundfalse
    \coremissesfalse
    \typeoutdetails{^^J----------------------------------------}%
    \typeoutdetails{Testing font family #1^^J(currently TS1-sub-encoding
      \currsubencoding{#1})}%
    \typeout{----------------------------------------}%
```

Then we start testing the groups beginning with the glyphs between sub-encoding 8 and 9. If any of them is missing (checked with `\doesglyphexist`) then we already know that 9 is the correct answer.

```
    \testgroup{%
       \doesglyphexist{21}{\texttwelveudash}%
       \doesglyphexist{22}{\textthreequartersemdash}%
       \doesglyphexist{134}{\textbardbl}%
       \doesglyphexist{137}{\textcelsius}%
       \doesglyphexist{178}{\texttwosuperior}%
       \doesglyphexist{179}{\textthreesuperior}%
       \doesglyphexist{185}{\textonesuperior}%
    }{8}{9}%
```

Nevertheless we go on with further groups so that the output lists all missing glyphs.

```
    \testgroup{%
       \doesglyphexist{32}{\textblank}%
       \doesglyphexist{148}{\textinterrobang}%
       \doesglyphexist{149}{\textinterrobangdown}%
       \doesglyphexist{191}{\texteuro}%
    }{7}{8}%
    \testgroup{%
       \doesglyphexist{47}{\textfractionsolidus}%
       \doesglyphexist{61}{\textminus}%
       \doesglyphexist{87}{\textohm}%
       \doesglyphexist{181}{\textmu}%
    }{6}{7}%
    \testgroup{%
       \doesglyphexist{140}{\textflorin}%
       \doesglyphexist{164}{\textcurrency}%
    }{5}{6}%
    \testgroup{%
       \doesglyphexist{155}{\textnumero}%
       \doesglyphexist{157}{\textestimated}%
    }{4}{5}%
    \testgroup{%
       \doesglyphexist{24}{\textleftarrow}%
       \doesglyphexist{25}{\textrightarrow}%
       \doesglyphexist{94}{\textuparrow}%
       \doesglyphexist{95}{\textdownarrow}%
       \doesglyphexist{141}{\textcolonmonetary}%
```

```
      \doesglyphexist{142}{\textwon}%
      \doesglyphexist{146}{\textlira}%
      \doesglyphexist{150}{\textdong}%
   }{3}{4}%
   \testgroup{%
      \doesglyphexist{60}{\textlangle}%
      \doesglyphexist{62}{\textrangle}%
   }{2}{3}%
   \testgroup{%
      \doesglyphexist{0}{\capitalgrave}%
      \doesglyphexist{1}{\capitalacute}%
      \doesglyphexist{2}{\capitalcircumflex}%
      \doesglyphexist{3}{\capitaltilde}%
      \doesglyphexist{4}{\capitaldieresis}%
      \doesglyphexist{5}{\capitalhungarumlaut}%
      \doesglyphexist{6}{\capitalring}%
      \doesglyphexist{7}{\capitalcaron}%
      \doesglyphexist{8}{\capitalbreve}%
      \doesglyphexist{9}{\capitalmacron}%
      \doesglyphexist{10}{\capitaldotaccent}%
      \doesglyphexist{11}{\capitalcedilla}%
      \doesglyphexist{12}{\capitalogonek}%
      \doesglyphexist{26}{\t}%
      \doesglyphexist{27}{\capitaltie}%
      \doesglyphexist{28}{\newtie}%
      \doesglyphexist{29}{\capitalnewtie}%
      \doesglyphexist{45}{\textdblhyphen}%
      \doesglyphexist{48}{\textzerooldstyle}%
      \doesglyphexist{49}{\textoneoldstyle}%
      \doesglyphexist{50}{\texttwooldstyle}%
      \doesglyphexist{51}{\textthreeoldstyle}%
      \doesglyphexist{52}{\textfouroldstyle}%
      \doesglyphexist{53}{\textfiveoldstyle}%
      \doesglyphexist{54}{\textsixoldstyle}%
      \doesglyphexist{55}{\textsevenoldstyle}%
      \doesglyphexist{56}{\texteightoldstyle}%
      \doesglyphexist{57}{\textnineoldstyle}%
      \doesglyphexist{77}{\textmho}%
      \doesglyphexist{79}{\textbigcircle}%
      \doesglyphexist{91}{\textlbrackdbl}%
      \doesglyphexist{93}{\textrbrackdbl}%
      \doesglyphexist{96}{\textasciigrave}%
      \doesglyphexist{98}{\textborn}%
      \doesglyphexist{99}{\textdivorced}%
      \doesglyphexist{100}{\textdied}%
      \doesglyphexist{108}{\textleaf}%
      \doesglyphexist{109}{\textmarried}%
      \doesglyphexist{110}{\textmusicalnote}%
      \doesglyphexist{126}{\texttildelow}%
      \doesglyphexist{127}{\textdblhyphenchar}%
      \doesglyphexist{128}{\textasciibreve}%
      \doesglyphexist{129}{\textasciicaron}%
      \doesglyphexist{175}{\textasciimacron}%
      \doesglyphexist{130}{\textacutedbl}%
```

```
      \doesglyphexist{131}{\textgravedbl}%
      \doesglyphexist{138}{\textdollaroldstyle}%
      \doesglyphexist{139}{\textcentoldstyle}%
      \doesglyphexist{143}{\textnaira}%
      \doesglyphexist{144}{\textguarani}%
      \doesglyphexist{145}{\textpeso}%
      \doesglyphexist{147}{\textrecipe}%
      \doesglyphexist{152}{\textpertenthousand}%
      \doesglyphexist{153}{\textpilcrow}%
      \doesglyphexist{154}{\textbaht}%
      \doesglyphexist{156}{\textdiscount}%
      \doesglyphexist{158}{\textopenbullet}%
      \doesglyphexist{159}{\textservicemark}%
      \doesglyphexist{160}{\textlquill}%
      \doesglyphexist{161}{\textrquill}%
      \doesglyphexist{168}{\textasciidieresis}%
      \doesglyphexist{171}{\textcopyleft}%
      \doesglyphexist{173}{\textcircledP}%
      \doesglyphexist{180}{\textasciiacute}%
      \doesglyphexist{184}{\textreferencemark}%
      \doesglyphexist{187}{\textsurd}%
   }{1}{2}%
```

All fonts (up to now) that belong to sub-encoding 1 do have the `\textcircled` glyph, but it is too small to be usable. So this test for this group currently doesn't do much good—but who knows maybe one day a font shows up in which this glyph is actually missing.

```
   \testgroup{%
     \doesglyphexist{79}{\textcircled}%  this is not a proper test because the symbol is
                                      %  usually available but not usable
   }{0}{1}%
   \testgroup{%
      \doesglyphexist{13}{\textquotestraightbase}%
      \doesglyphexist{18}{\textquotestraightdblbase}%
      \doesglyphexist{23}{\textcapitalcompwordmark}%
      \doesglyphexist{31}{\textascendercompwordmark}%
      \doesglyphexist{36}{\textdollar}%
      \doesglyphexist{39}{\textquotesingle}%
      \doesglyphexist{42}{\textasteriskcentered}%
      \doesglyphexist{132}{\textdagger}%
      \doesglyphexist{133}{\textdaggerdbl}%
      \doesglyphexist{135}{\textperthousand}%
      \doesglyphexist{136}{\textbullet}%
      \doesglyphexist{151}{\texttrademark}%
      \doesglyphexist{162}{\textcent}%
      \doesglyphexist{163}{\textsterling}%
      \doesglyphexist{165}{\textyen}%
      \doesglyphexist{166}{\textbrokenbar}%
      \doesglyphexist{167}{\textsection}%
      \doesglyphexist{169}{\textcopyright}%
      \doesglyphexist{170}{\textordfeminine}%
      \doesglyphexist{172}{\textlnot}%
      \doesglyphexist{174}{\textregistered}%
      \doesglyphexist{176}{\textdegree}%
```

```
        \doesglyphexist{177}{\textpm}%
        \doesglyphexist{182}{\textparagraph}%
        \doesglyphexist{183}{\textperiodcentered}%
        \doesglyphexist{186}{\textordmasculine}%
        \doesglyphexist{188}{\textonequarter}%
        \doesglyphexist{189}{\textonehalf}%
        \doesglyphexist{190}{\textthreequarters}%
        \doesglyphexist{214}{\texttimes}%
        \doesglyphexist{246}{\textdiv}%
    }{-1}{0}%
```

If all groups have all glyphs then we have the full encoding (subset 0).

```
    \ifsafesubencodingfound\else
      \def\subencodingresult{0}%
    \fi
```

If the font is missing some of the core glyphs we make a remark about this, because they will never display.

```
    \typeoutdetails{----------------------------------------}%
    \typeout{TS1 encoding subset for #1\ifcoremisses \space(ignoring core misses)\fi
      \space (\ifnum\subencodingresult =
      \currsubencoding{#1} ok\else bad\fi)}%
    \typeout{Use sub-encoding \subencodingresult
      \ifnum\subencodingresult = \currsubencoding{#1}\else
        \space (not \currsubencoding{#1})\fi}
    \typeout{----------------------------------------^^J}%
  \fi
}
```

This tests all declarations (or most of them) that have been added to the kernel. It is called if no family is given interactively.

```
\long\def\testallkerneldefinedfamilies{%
\testallgroups{ccr}%     {0}
\testallgroups{cmbr}%    {0}
%%\testallgroups{cmr}%   {0}  % don't test this one as it is the fallback
                              % thus reports that the family is not found
\testallgroups{cmss}%    {0}
\testallgroups{cmtl}%    {0}
\testallgroups{cmtt}%    {0}
\testallgroups{cmvtt}%   {0}
\testallgroups{pxr}%     {0}
\testallgroups{pxss}%    {0}
\testallgroups{pxtt}%    {0}
\testallgroups{qag}%     {0}
\testallgroups{qbk}%     {0}
\testallgroups{qcr}%     {0}
\testallgroups{qcs}%     {0}
\testallgroups{qhvc}%    {0}
\testallgroups{qhv}%     {0}
\testallgroups{qpl}%     {0}
\testallgroups{qtm}%     {0}
\testallgroups{qzc}%     {0}
\testallgroups{txr}%     {0}
\testallgroups{txss}%    {0}
\testallgroups{txtt}%    {0}
```

```
%
% Next would claim to be 0 (or 2)
%
%\testallgroups{lmr}%     {1}
%\testallgroups{lmdh}%    {1}
%\testallgroups{lmss}%    {1}
%\testallgroups{lmssq}%   {1}
%\testallgroups{lmvtt}%   {1}
%\testallgroups{lmtt}%    {1} % missing TM, SM and pertenthousand so really 2
%
% these are no longer in TeX Live
%
%\testallgroups{ptmx}%    {2} % gone for a long time it seems
%\testallgroups{ptmj}%    {2} % ditto
%\testallgroups{ul8}%     {2} % ditto
%
% next block has tofu chars so results are wrong
%
%\testallgroups{bch}% {5} % tofu for blank, ohm
%\testallgroups{futj}%{5} % tofu for blank, interrobang/down, ohm
%\testallgroups{futs}%{5} % tofu for blank, ohm
%\testallgroups{futx}%{5} % probably (currently broken distrib)
%\testallgroups{pag}% {5} % tofu for blank, interrobang/down, ohm
%\testallgroups{pbk}% {5} % tofu for blank, interrobang/down, ohm
%\testallgroups{pcr}% {5} % tofu for blank, interrobang/down, ohm
%\testallgroups{phv}% {5} % tofu for blank, interrobang/down, ohm
%\testallgroups{pnc}% {5} % tofu for blank, interrobang/down, ohm
%\testallgroups{pplj}%{5} % tofu for blank
%\testallgroups{pplx}%{5} % tofu for blank
%\testallgroups{ppl}% {5} % tofu for blank interrobang/down
%\testallgroups{ptm}% {5} % tofu for blank, interrobang/down, ohm
%\testallgroups{pzc}% {5} % tofu for blank, interrobang/down, ohm
%\testallgroups{ul9}% {5} % tofu for blank, interrobang/down, ohm
%\testallgroups{dayroms}%{6} % tofu for blank, interrobang/down, ohm
%\testallgroups{dayrom}% {6} % tofu for blank, interrobang/down, ohm
%\testallgroups{augie}%{8} % really only missing euro and full of tofu
%\testallgroups{put}%  {8}
%\testallgroups{uag}%  {8} % probably (currently broken distrib)
%\testallgroups{ugq}%  {8}
%
\testallgroups{zi4}%  {9}
%
%% not installed normally
%
%\testallgroups{hls}%  {5}
%\testallgroups{hlst}% {5}
%\testallgroups{hlct}% {5}
%\testallgroups{hlh}%  {5}
%\testallgroups{hlx}%  {8}
%\testallgroups{hlce}% {8}
%\testallgroups{hlcn}% {8}
%\testallgroups{hlcw}% {8}
%\testallgroups{hlcf}% {8}
```

```
\testallgroups{lato-LF}%   {0}  % with a bunch of tofu inside --- should probably be changed
\testallgroups{opensans-TLF}%{0}  % with a bunch of tofu inside --- should probably be change
\testallgroups{cantarell-TLF}%  {0}  % with a bunch of tofu inside --- should probably be cha
\testallgroups{fbb-LF}%     {0}  % missing centoldstyle ---> 2
\testallgroups{tli}%       {1}  % with lots of tofu inside --- should probably be changed
\testallgroups{Alegreya-OsF}%       {2}
\testallgroups{AlegreyaSans-OsF}%   {2}
\testallgroups{DejaVuSans-TLF}%     {2}
\testallgroups{DejaVuSansCondensed-TLF}%  {2}
\testallgroups{DejaVuSansMono-TLF}% {2} this is missing \textfractionsolidus which makes it 7
\testallgroups{EBGaramond-LF}%      {2}
\testallgroups{Tempora-TLF}%        {2}
\testallgroups{Tempora-TOsF}%       {2}
\testallgroups{Arimo-TLF}%          {3}
\testallgroups{Crlt-TLF}%           {3}       changed from Carlito-
\testallgroups{FiraSans-LF}%        {3}     should be 4
\testallgroups{IBMPlexSans-TLF}%    {3}
\testallgroups{Merriwthr-OsF}%   {3}         changed from Merriweather- and should be 2
\testallgroups{Montserrat-LF}%      {3}     now 2
\testallgroups{MontserratAlternates-LF}%{3}     now 2
\testallgroups{SourceCodePro-TLF}%  {3}
\testallgroups{SourceCodePro-TOsF}% {3}
\testallgroups{SourceSansPro-OsF}%  {3}
\testallgroups{SourceSerifPro-LF}%  {3}
\testallgroups{Tinos-TLF}%          {3}
\testallgroups{AccanthisADFStdNoThree-LF}%{4}
\testallgroups{Cabin-TLF}%          {4}
\testallgroups{Caladea-TLF}%        {4}
\testallgroups{Chivo-LF}%           {4}
\testallgroups{ClearSans-TLF}%      {4}
\testallgroups{Coelacanth-LF}%      {4}
\testallgroups{CrimsonPro-LF}%      {4}
\testallgroups{FiraMono-TLF}%       {4}
\testallgroups{FiraMono-TOsF}%      {4}
\testallgroups{Go-TLF}%             {4}
\testallgroups{GoMono-TLF}%         {4}
\testallgroups{InriaSans-LF}%       {4}
\testallgroups{InriaSerif-LF}%      {4}
\testallgroups{LibertinusSans-LF}%  {4}
\testallgroups{LibertinusSerif-LF}% {4}
\testallgroups{LibreBodoni-TLF}%    {4}
\testallgroups{LibreFranklin-TLF}%  {4}
\testallgroups{LinguisticsPro-LF}%  {4}
\testallgroups{LinguisticsPro-OsF}% {4}
\testallgroups{LinuxBiolinumT-LF}%  {4}
\testallgroups{LinuxLibertineT-LF}% {4}
\testallgroups{MerriwthrSans-OsF}%  {4}          name change and now 2
\testallgroups{MintSpirit-LF}%      {4}
\testallgroups{MintSpiritNoTwo-LF}% {4}
\testallgroups{PTMono-TLF}%         {4}
\testallgroups{PTSans-TLF}%         {4}
\testallgroups{PTSansCaption-TLF}%  {4}
\testallgroups{PTSansNarrow-TLF}%   {4}
\testallgroups{PTSerif-TLF}%        {4}
```

```
\testallgroups{PTSerifCaption-TLF}% {4}
\testallgroups{Raleway-TLF}%        {4}
\testallgroups{Raleway-TOsF}%       {4}
\testallgroups{Roboto-LF}%          {4}
\testallgroups{RobotoMono-TLF}%     {4}
\testallgroups{RobotoSlab-TLF}%     {4}
\testallgroups{Rosario-LF}%         {4}
\testallgroups{SticksTooText-LF}%   {4}
\testallgroups{UniversalisADFStd-LF}%{4}
\testallgroups{Almndr-OsF}%       {5}     name change
\testallgroups{Baskervaldx-LF}%   {5}
\testallgroups{BaskervilleF-LF}%  {5}   now 2
\testallgroups{Bttr-TLF}%         {5}   name changed from Bitter-...
\testallgroups{Cinzel-LF}%        {5}
\testallgroups{CinzelDecorative-LF}%{5}
\testallgroups{DejaVuSerif-TLF}%  {5}
\testallgroups{DejaVuSerifCondensed-TLF}% {5}
\testallgroups{GilliusADF-LF}%    {5}
\testallgroups{charssil-TLF} %%  missing should be 5
\testallgroups{GilliusADFCond-LF}%{5}
\testallgroups{GilliusADFNoTwo-LF}%{5}
\testallgroups{GilliusADFNoTwoCond-LF}%{5}
\testallgroups{Lbstr-LF}%      {5}      name change  and should be 7
\testallgroups{OldStandard-TLF}%  {5}
\testallgroups{PlyfrDisplay-LF}%{5}       name change
\testallgroups{PlyfrDisplay-OsF}% {5}       name change
\testallgroups{TheanoDidot-TLF}%  {5}
\testallgroups{TheanoDidot-TOsF}% {5}
\testallgroups{TheanoModern-TLF}% {5}
\testallgroups{TheanoModern-TOsF}%{5}
\testallgroups{TheanoOldStyle-TLF}%{5}
\testallgroups{TheanoOldStyle-TOsF}%{5}
\testallgroups{Crimson-TLF}%      {6}
\testallgroups{IBMPlexMono-TLF}%  {6}  now 3
\testallgroups{IBMPlexSerif-TLF}% {6}  now 3
\testallgroups{LibertinusMono-TLF}%{6}  should be 8
\testallgroups{LibertinusSerifDisplay-LF}%{6}
\testallgroups{LinuxLibertineDisplayT-LF}%{6}
\testallgroups{LinuxLibertineMonoT-LF}%{6}
\testallgroups{LinuxLibertineMonoT-TLF}%{6}
\testallgroups{Ovrlck-LF}%       {6}     name changed
\testallgroups{CormorantGaramond-LF}%{7}
\testallgroups{Heuristica-TLF}%  {7}
\testallgroups{Heuristica-TOsF}% {7}
\testallgroups{IMFELLEnglish-TLF}%{7}
\testallgroups{LibreBskvl-LF}%   {7}  %%  wrong name LibreBaskerville-TLF
\testallgroups{LibreCsln-LF}%    {7}  changed from LibreCaslon-
\testallgroups{Mrcls-LF}%        {7}       %%  wrong name Marcellus-LF
\testallgroups{NotoSans-LF}%     {7}
\testallgroups{NotoSansMono-TLF}%{7}  now 2
\testallgroups{NotoSansMono-TOsF}%{7}  now 2
\testallgroups{NotoSerif-LF}%    {7}
\testallgroups{Quattro-LF}%      {7}  changed from Quattrocento-
\testallgroups{QuattroSans-LF}%  {7}  changed from QuattrocentoSans-
```

```
\testallgroups{XCharter-TLF}%    {7}  now 2
\testallgroups{XCharter-TOsF}%   {7}  now 2
\testallgroups{erewhon-LF}%      {7}  now 2
\testallgroups{ComicNeue-TLF}%   {7}
\testallgroups{ComicNeueAngular-TLF}%{7}
\testallgroups{Frm-LF}%{7}  % the superiors are missing; name changed from Forum-LF
\testallgroups{Cochineal-TLF}%   {8} now 5
\testallgroups{AlgolRevived-TLF}%{9}
}
```

There interaction with the user.

```
\typeout{^^J==================================================================}
\typeout{| Enter font family to check (or <enter> for kernel defined families)}
\typeout{==================================================================}
\typein[\FontFamilyToCheck]{}
```

```
\if!\FontFamilyToCheck!
  \typeout{==================================================================}
  \typeout{| Detailed output? (default no)}
  \typeout{==================================================================}
  \typein[\Details]{}
  \if!\Details!
    \def\typeoutdetails#1{}
  \else
    \let\typeoutdetails\typeout
  \fi
  \testallkerneldefinedfamilies
\else
  \let\typeoutdetails\typeout
  \testallgroups\FontFamilyToCheck
\fi
```

```
\stop
```
⟨/TS1check⟩

File 34
ltpageno.dtx

1 Page Numbering

Page numbers are produced by a page counter, used just like any other counter. The only difference is that `\c@page` contains the number of the next page to be output (the one currently being produced), rather than one minus it. Thus, it is normally initialized to 1 rather than 0. `\c@page` is defined to be `\count0`, rather than a count assigned by `\newcount`.

`\pagenumbering` The user sets the page number style with the `\pagenumbering{`⟨*foo*⟩`}` command, which sets the page counter to 1 and defines `\thepage` to be `\foo`. For example, `\pagenumbering{roman}` causes pages to be numbered i, ii, etc.

```
⟨*2ekernel⟩
\message{page nos.,}
\countdef\c@page=0 \c@page=1
\def\cl@page{}
\def\pagenumbering#1{%
  \global\c@page \@ne \gdef\thepage{\csname @#1\endcsname
   \c@page}}
⟨/2ekernel⟩
```

File 35
ltxref.dtx

1 Cross Referencing

The user writes `\label{⟨foo⟩}` to define the following cross-references:

`\ref*{⟨foo⟩}`: value of most recently incremented referenceable counter. in the current environment. (Chapter, section, theorem, footnote and enumeration counters and other counters stepped with `\refstepcounter` are referenceable.)

`\pageref*{⟨foo⟩}`: page number at which `\label{foo}` command appeared. where foo can be any string of characters not containing '\', '{' or '}'.

Note: The scope of the `\label` command is delimited by environments, so
`\begin{theorem} \label{foo} ... \end{theorem} \label{bar}`
defines `\ref{foo}` to be the theorem number and `\ref{bar}` to be the current section number.

Note: `\label` does the right thing in terms of spacing – i.e., leaving a space on both sides of it is equivalent to leaving a space on either side.

Note: the starred versions `\ref*` and `\pageref*` are provided to align with the use of hyperref. Without hyperref (or some other package using the starred form) the star is simply ignored.

Note: starting with 2023-06-01 `\label` stores also the current value of `\@currentlabelname` which should typically contain a (sanitized) title. (A reference command `\nameref` is provided by the nameref package.) `\label` also stores `\@currentHref` which if set should refer to a target name for links. This value is set and used by hyperref. Unlike the other values `\@currentHref` should be set globally. A fifth value `\@kernel@reserved@label@data` is reserved for the kernel to allow future extensions of the cross-reference system.

1.1 Cross Referencing

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
⟨*2ekernel⟩
\message{x-ref,}
```

This is implemented as follows. A referenceable counter CNT is incremented by the command `\refstepcounter{CNT}` , which sets `\@currentlabel == {CNT}{eval(\p@cnt\theCNT)}`. The command `\label{FOO}` then writes the following on file `\@auxout` :

```
      \newlabel{FOO}{{eval(\@currentlabel)}{eval(\thepage)}%
{eval(\@currentlabelname)}{eval(\@currentHref)}{eval(\@kernel@reserved@label@data)}}
```

```
\ref{FOO} ==
  BEGIN
    if \r@foo undefined
      then  @refundefined := G T
            ??
            Warning: 'reference foo on page ... undefined'
```

```
        else   \@car \eval(\r@FOO)\@nil
      fi
    END

  \pageref{foo} =
    BEGIN
      if \r@foo undefined
        then   @refundefined := G T
               ??
               Warning: 'reference foo on page ... undefined'
        else   \@cdr \eval(\r@FOO)\@nil
      fi
    END
```

End of historical LaTeX 2.09 comments.

`\labelformat` A reference via `\ref` produces by default the data associated with the corresponding `\label` command (typically a number); any additional formatting has to be provided by the user. If, for example, references to equations are always to be typeset as "equation (*number*)", one has to code "`equation (\ref{`*key*`})`". With `\labelformat` there is a possibility to generate such frills automatically without resorting to low-level coding. The command takes two arguments: the first is the name of a counter and the second is its representation when referenced. This means that for a successful usage, one has to know the counter name being used for generating the label, though in practice this should not pose a problem. The current counter number is picked up as an argument. Here are two examples:

```
\labelformat{section}{section~#1}
\labelformat{equation}{equation~(#1)}}
```

`\Ref` A side effect of using `\labelformat` is that, depending on the defined formatting, it becomes impossible to use `\ref` at the beginning of a sentence (if its replacement text starts with a lowercase letter). To overcome this problem we introduce the command `\Ref` that behave like `\ref` except that it uppercases the first token of the generated string.

To make `\Ref` work properly the very first token in the second argument of `\labelformat` has to be a simple ASCII or UTF-8 letter, otherwise the capitalization will fail or worse, you will end up with some error messages. If you actually need something more complicated in this place (e.g., an accented letter not written as a UTF-8 character) you have to explicitly surround it with braces, to identify the part that needs to be capitalized. For example, for figure references in the Hungarian language you might want to write `\labelformat{figure}{{\'a}bra~\thefigure}` or use `\labelformat{figure}{ábra~\thefigure}` which avoids the brace problem.

`\G@refundefinedtrue` `\@refundefined` This does not save on name-space (since `\G@refundefinedfalse` was never needed) but it does make the implementation of such one-way switches more consistent. The extra macro to make the change is used since this change appears several times.

Note despite its name, `\G@refundefinedtrue` does *not* correspond to an `\if` command, and there is no matching ...`false`. It would be more natural to call the command `\G@refundefined` (as inspection of the change log will reveal) but unfortunately such a change would break any package that had defined a `\ref`-like command that mimicked the definition of `\ref`, calling `\G@refundefinedtrue`. Inspection of the TeX archives

revealed several such packages, and so this command has been named ...`true` so that the definition of `\ref` need not be changed, and the packages will work without change.

```
% \newif\ifG@refundefined
% \def\G@refundefinedtrue{\global\let\ifG@refundefined\iftrue}
% \def\G@refundefinedfalse{\global\let\ifG@refundefined\iffalse}
\def\G@refundefinedtrue{%
  \gdef\@refundefined{%
    \@latex@warning@no@line{There were undefined references}}}
\let\@refundefined\relax
```

(*End of definition for* `\G@refundefinedtrue` *and* `\@refundefined`.)

`\ref` `\pageref` `\@setref`

Referencing a `\label`. RmS 91/10/25: added a few extra `\reset@font`, as suggested by Bernd Raichle

RmS 92/08/14: made `\ref` and `\pageref` robust

RmS 93/09/08: Added setting of refundefined switch.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2023/06/01}%
⟨latexrelease⟩                  {\@kernel@sref}{store five arguments}%
\def\@setref#1#2#3{%
  \ifx#1\relax
   \protect\G@refundefinedtrue
   \nfss@text{\reset@font\bfseries ??}%
   \@latex@warning{Reference ‘#3’ on page \thepage \space
             undefined}%
  \else
   \expandafter#2#1\@empty\@empty\@empty\null
  \fi}
```

```
\long\def\@firstoffive#1#2#3#4#5{#1}
\long\def\@secondoffive#1#2#3#4#5{#2}
\def\@kernel@sref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoffive{#1}}
\def\@kernel@spageref#1{\expandafter\@setref\csname r@#1\endcsname
                  \@secondoffive{#1}}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                  {\@kernel@sref}{store five arguments}%
⟨latexrelease⟩\def\@setref#1#2#3{%
⟨latexrelease⟩  \ifx#1\relax
⟨latexrelease⟩   \protect\G@refundefinedtrue
⟨latexrelease⟩   \nfss@text{\reset@font\bfseries ??}%
⟨latexrelease⟩   \@latex@warning{Reference ‘#3’ on page \thepage \space
⟨latexrelease⟩             undefined}%
⟨latexrelease⟩  \else
⟨latexrelease⟩   \expandafter#2#1\null
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\let\@firstoffive\undefined
⟨latexrelease⟩\let\@secondoffive\undefined
⟨latexrelease⟩\def\@kernel@sref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
⟨latexrelease⟩\def\@kernel@spageref#1{\expandafter\@setref\csname r@#1\endcsname
⟨latexrelease⟩                  \@secondoftwo{#1}}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
```

```
⟨latexrelease⟩                  {\@kernel@sref}{store five arguments}%
⟨latexrelease⟩\def\@setref#1#2#3{%
⟨latexrelease⟩  \ifx#1\relax
⟨latexrelease⟩   \protect\G@refundefinedtrue
⟨latexrelease⟩   \nfss@text{\reset@font\bfseries ??}%
⟨latexrelease⟩   \@latex@warning{Reference ‘#3’ on page \thepage \space
⟨latexrelease⟩              undefined}%
⟨latexrelease⟩  \else
⟨latexrelease⟩   \expandafter#2#1\null
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\let\@firstoffive\undefined
⟨latexrelease⟩\let\@secondoffive\undefined
⟨latexrelease⟩\let\@kernel@sref\undefined
⟨latexrelease⟩\let\@kernel@spageref\undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                  {\ref}{Add starred reference commands}%
```

```
\let\@kernel@ref\@kernel@sref
\let\@kernel@pageref\@kernel@spageref
\NewDocumentCommand\ref{s}
   {\IfBooleanTF{#1}{\@kernel@sref}{\@kernel@ref}}
\NewDocumentCommand\pageref{s}
   {\IfBooleanTF{#1}{\@kernel@spageref}{\@kernel@pageref}}
```

As the commands are now protected we also need expandable versions for use in `\ifthenelse`:

```
\def\@kernel@pageref@exp#1{\csname cs_if_exist:cTF\endcsname
   {r@#1}{\csname tl_item:cn\endcsname{r@#1}{2}}{0}}
\def\@kernel@ref@exp#1{\csname cs_if_exist:cTF\endcsname
   {r@#1}{\csname tl_item:cn\endcsname{r@#1}{1}}{0}}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\ref}{Add starred reference commands}%
⟨latexrelease⟩\def\ref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
⟨latexrelease⟩\def\pageref#1{\expandafter\@setref\csname r@#1\endcsname
⟨latexrelease⟩                                    \@secondoftwo{#1}}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\ref`*,* `\pageref`*, and* `\@setref`*.*)

`\newlabel` This command will be written to the `.aux` file to pass label information from one run to another.

`\@newl@bel` The internal form of `\newlabel` and `\bibcite`. Note that this macro does it's work inside a group. That way the local assignments it needs to do don't clutter the save stack. This prevents large documents with many labels to run out of save stack.

```
\def\@newl@bel#1#2#3{{%
  \@ifundefined{#1@#2}%
    \relax
    {\gdef \@multiplelabels {%
       \@latex@warning@no@line{There were multiply-defined labels}}%
```

```
    \@latex@warning@no@line{Label ‘#2’ multiply defined}}%
  \global\@namedef{#1@#2}{#3}}}
\def\newlabel{\@newl@bel r}
\@onlypreamble\@newl@bel
```

(*End of definition for* `\newlabel` *and* `\@newl@bel`.)

`\if@multiplelabels` `\@multiplelabels` This is redefined to produce a warning if at least one label is defined more than once. It is executed by the `\enddocument` command.

```
\let \@multiplelabels \relax
```

(*End of definition for* `\if@multiplelabels` *and* `\@multiplelabels`.)

`\label` The commands `\label` and `\refstepcounter` have been changed to allow `\protect`'ed commands to work properly. For example,

```
\def\thechapter{\protect\foo{\arabic{chapter}.\roman{section}}}
```

will cause a `\label{bar}` command to define `\ref{bar}` to expand to something like `\foo{4.d}`. Change made 20 Jul 88.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2023/06/01}%
⟨latexrelease⟩                  {\label}{store five label arguments}%
\providecommand\@currentlabelname{}
\providecommand\@currentHref{}
\providecommand\@kernel@reserved@label@data{}
\NewHookWithArguments{label}{1}
\def\label#1{\@bsphack
  \begingroup
  \UseHookWithArguments{label}{1}{#1}%
  \protected@write\@auxout{}%
         {\string\newlabel{#1}{{\@currentlabel}{\thepage}%
          {\@currentlabelname}{\@currentHref}{\@kernel@reserved@label@data}}}%
  \endgroup
  \@esphack}
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\label`. *This function is documented on page* 750.)

`\refstepcounter` Step the counter and allow for labels to point to its current value.

```
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                  {\Ref}{Add starred version}%
\def\@currentcounter{}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2024/11/01}%
⟨latexrelease⟩                  {\@currentHref}{set theHcounter representation}%
```

`refstepcounter` (*socket*) This socket takes the whole code as argument. The default kernel plug is identity. By changing the plug hyperref can add a conditional and e.g. suppress the processing in a PDF context.

```
\NewSocket{refstepcounter}{1}
```

`refstepcounter/target` (*socket*) This socket takes an argument, the counter name, and should at least set from it the target name `\@currentHref`. With hyperref it sets also the actual target. This is done with a socket so that the target name is not set more than once to (possibly) different names. The socket is not used in `\@kernel@refstepcounter`. The tagging code needs the target name so it is added after this socket.

```
\NewSocket{refstepcounter/target}{1}
```

`(refstepcounter/target)` (*plug*)

```
\NewSocketPlug{refstepcounter/target}{kernel}
 {\xdef\@currentHref {#1.\csname theH#1\endcsname}}%
\AssignSocketPlug{refstepcounter/target}{kernel}
```

```
\def\refstepcounter#1{%
  \UseSocket{refstepcounter}{%
    \stepcounter{#1}%
    \edef\@currentcounter{#1}%
    \protected@edef\@currentlabel
```

By generating the second csname first the `\p@...` command can grab it as an argument which can be helpful for more complicated typesetting arrangements.

The trick is to ensure that `\csname the#1\endcsname` is turned into a single token before `\p@...` is expanded further. This way, if the `\p@...` command is a macro with one argument it will receive `\the...`. With the original kernel code (i.e., without the `\expandafter`) it will instead pick up `\csname` which would be disastrous.

Using `\expandafter` instead of braces delimiting the argument is better because, assuming that the `\p@...` command is not defined as a macro with one argument, the braces will stay and prohibit kerning that might otherwise happen between the glyphs generated by `\the...` and surrounding glyphs.

```
       {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
     \UseSocket{refstepcounter/target}{#1}%
     \UseTaggingSocket{recordtarget}%
   }%
}
```

This is a version of `\refstepcounter` which does not set and use targets.

`\@kernel@refstepcounter`

```
\def\@kernel@refstepcounter#1{%
  \UseSocket{refstepcounter}{%
    \stepcounter{#1}%
    \edef\@currentcounter{#1}%
    \protected@edef\@currentlabel
       {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}}}%
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                  {\@currentHref}{set theHcounter representation}%
⟨latexrelease⟩\def\refstepcounter#1{\stepcounter{#1}%
⟨latexrelease⟩    \edef\@currentcounter{#1}%
⟨latexrelease⟩    \protected@edef\@currentlabel
⟨latexrelease⟩      {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
⟨latexrelease⟩}
⟨latexrelease⟩\let\@kernel@refstepcounter\refstepcounter
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\refstepcounter` *and* `\@kernel@refstepcounter`.)

```
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                  {\Ref}{Add starred version}%
```

`\labelformat` A shortcut to set the `\p@...` macro for a counter. It will pick up the counter representation as an argument so that it can be specially formatted.

```
\def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
```

(*End of definition for* `\labelformat`.)

`\Ref` This macro expands the result of `\ref` and then uppercases the first token. Only useful if the label was generated via `\labelformat` and contains some lower case letter at its start. If the label starts with a complicated construct (e.g., an accented letter that is provided via a command, e.g., `\"a` instead of a UTF-8 character like ä) one has to surround everything that needs uppercasing in a brace group in the definition of `\labelformat`.[39]

```
\def\@kernel@Ref#1{\protected@edef\@tempa{\@kernel@ref{#1}}%
       \expandafter\MakeUppercase\@tempa}
\def\@kernel@sRef#1{\protected@edef\@tempa{\@kernel@sref{#1}}%
       \expandafter\MakeUppercase\@tempa}
\NewDocumentCommand\Ref{s}
   {\IfBooleanTF{#1}{\@kernel@sRef}{\@kernel@Ref}}
```

(*End of definition for* `\Ref`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\label}{store five label arguments}%
⟨latexrelease⟩\let\@currenttitle\@undefined
⟨latexrelease⟩\let\@currenttarget\@undefined
⟨latexrelease⟩\let\@kernel@currentdata\@undefined
⟨latexrelease⟩\def\label#1{\@bsphack
⟨latexrelease⟩  \protected@write\@auxout{}%
⟨latexrelease⟩         {\string\newlabel{#1}{{\@currentlabel}{\thepage}}}%
⟨latexrelease⟩  \@esphack}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\Ref}{Add starred version}%
⟨latexrelease⟩\def\@currentcounter{}
⟨latexrelease⟩\def\refstepcounter#1{\stepcounter{#1}%
⟨latexrelease⟩    \edef\@currentcounter{#1}%
⟨latexrelease⟩    \protected@edef\@currentlabel
⟨latexrelease⟩       {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
⟨latexrelease⟩}
⟨latexrelease⟩\def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
⟨latexrelease⟩\DeclareRobustCommand\Ref[1]{\protected@edef\@tempa{\ref{#1}}%
⟨latexrelease⟩   \expandafter\MakeUppercase\@tempa}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\refstepcounter}{Add \labelformat and \Ref}%
⟨latexrelease⟩\let\@currentcounter\@undefined
```

[39] There is one problem with this approach: the braces are kept in a normal `\ref` which might spoil kerning. Perhaps one day this needs redoing.

```
⟨latexrelease⟩\def\refstepcounter#1{\stepcounter{#1}%
⟨latexrelease⟩    \protected@edef\@currentlabel
⟨latexrelease⟩      {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
⟨latexrelease⟩}
⟨latexrelease⟩\def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
⟨latexrelease⟩\DeclareRobustCommand\Ref[1]{\protected@edef\@tempa{\ref{#1}}%
⟨latexrelease⟩   \expandafter\MakeUppercase\@tempa}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\refstepcounter}{Add \labelformat and \Ref}%
⟨latexrelease⟩
⟨latexrelease⟩\def\refstepcounter#1{\stepcounter{#1}%
⟨latexrelease⟩    \protected@edef\@currentlabel
⟨latexrelease⟩       {\csname p@#1\endcsname\csname the#1\endcsname}%
⟨latexrelease⟩}
⟨latexrelease⟩\let\labelformat\@undefined
⟨latexrelease⟩\let\Ref\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\@currentlabel` Default for `\label` commands that come before any environment.

```
\def\@currentlabel{}
```

(*End of definition for* `\@currentlabel`.)

```
⟨/2ekernel⟩
```

File 36
ltproperties.dtx

Abstract

This code implements command to record and (expandably) reference document properties. It extends the standard `\label`/`\ref`/`\pageref` commands.

1 Introduction

The module allows to record the "current state" of various document properties (typically the content of macros and values of counters) and to access them in other places through a label. The list of properties that can be recorded and retrieved are not fix and can be extended by the user. The values of the properties are recorded in the `.aux` file and can be retrieved at the second compilation.

The module uses the ideas of properties and labels. A label is a document reference point: a name for the user. An property is something that LaTeX can track, such as a page number, section number or name. The names of labels and properties may be arbitrary. Note that there is a single namespace for each.

2 Design discussion

The design here largely follows ideas from zref. In particular, there are two independent concepts: properties that can be recorded between runs, and labels which consist of lists of these properties. The reason for the split is that individual labels will want to record some but not all properties. For examples, a label concerned with position would track the x and y coordinates of the current point, but not for example the page number.

In the current implementation, properties share a single namespace. This allows multiple lists to re-use the same properties, for example page number, absolute page number, etc. This does mean that *changing* a standard property is an issue. However, some properties have complex definitions (again, see zref at present): having them in a single shared space avoids the need to copy code.

Labels could be implemented as `prop` data. That is not done at present as there is no obvious need to map to or copy the data. As such, faster performance is available using a hash table approach as in a "classical" set up. Data written to the `.aux` file uses simple paired *balanced text* not keyvals: this avoids any restrictions on names and again offers increased performance.

The expl3 versions of the label command do not use `\@bsphack`/`\@esphack` to avoid double spaces, but the LaTeX 2_ε command does as it lives at the document command level.

The reference commands are expandable.

Currently the code has nearly no impact on the main `\label` and `\ref` commands as too many external packages rely on the concrete implementation. There is one exception: the label names share the same namespace. That means that if both `\label{ABC}` and `\RecordProperties{ABC}{page}` are used there is a warning `Label 'ABC' multiply defined`.

3 Handling unknown labels and properties

With the standard `\label`/`\ref` commands the requested label is either in the `.aux`-file (and so known) or not. In the first case the stored value can be used, in the second case the reference commands print two question marks.

With flexible property lists a reference commands asks for the value of a specific property stored under a label name and we have to consider more variants:

- If the requested property is unknown (not declared) the system is not correctly set up and an error is issued.
- If the label is unknown, the default of the property is used.
- If the label is known, but doesn't provide a value for the property then again the default of the property is used.
- The command `\property_ref:nnn` allows to give a local default which is used instead of the property default in the two cases before.

4 Rerun messages

As the reference commands are expandable they can neither issue a message that the label or the label-property combination is unknown, nor can they trigger the rerun message at the end of the LaTeX run.

Where needed such messages must therefore be triggered manually. For this two commands are provided: `\property_ref_undefined_warn:` and `\property_ref_undefined_warn:nn`. See below for a description.

5 Open points

- The `xpos` and `ypos` properties require that the position is stored first but there is no (public) engine independent interface yet. Code must use `\tex_savepos:D`.

6 Code interfaces

`\property_new:nnnn` `\property_new:nnnn {⟨property⟩} {⟨setpoint⟩} {⟨default⟩} {⟨code⟩}`
`\property_gset:nnnn` `\property_gset:nnnn {⟨property⟩} {⟨setpoint⟩} {⟨default⟩} {⟨code⟩}`

LaTeX 2ε-interface: see `\NewProperty`, `\SetProperty`.
Sets the ⟨*property*⟩ to have the ⟨*default*⟩ specified, and at the ⟨*setpoint*⟩ (either `now` or `shipout`) to write the result of the ⟨*code*⟩ as part of a label. The ⟨*code*⟩ should be expandable. The expansion of ⟨*code*⟩ (the value of the property) is written to the `.aux` file and read back from there at the next compilation. Values should assume that the standard LaTeX catcode régime with `@` a letter is active then.

If the property is declared within a package it is suggested that its name is build from letters, hyphens and slashes, and is always structured as follows:
⟨*package-name*⟩/⟨*property-name*⟩.

\property_record:nN
\property_record:nn
\property_record:(nV|ee)

\property_record:nN {⟨*label*⟩} ⟨*clist var*⟩
\property_record:nn {⟨*label*⟩} {⟨*clist*⟩}

LaTeX 2ε-interface: see \RecordProperties.
Writes the list of properties given by the ⟨*clist*⟩ to the .aux file with the ⟨*label*⟩ specified.

\property_ref:nn ⋆
\property_ref:ee ⋆

\property_ref:nn {⟨*label*⟩} {⟨*property*⟩}

LaTeX 2ε-interface: see \RefProperty.
Expands to the value of the ⟨*property*⟩ for the ⟨*label*⟩, if available, and the default value of the property otherwise. If ⟨*property*⟩ has not been declared with \property_new:nnnn an error is issued. The command raises an internal, expandable, local flag if the reference can not be resolved.

\property_ref:nnn ⋆
\property_ref:een ⋆

\property_ref:nnn {⟨*label*⟩} {⟨*property*⟩} {⟨*local default*⟩}

LaTeX 2ε-interface: see \RefProperty.
Expands to the value of the ⟨*property*⟩ for the ⟨*label*⟩, if available, and to ⟨*local default*⟩ otherwise. If ⟨*property*⟩ has not been declared with \property_new:nnnn an error is issued. The command raises an internal, expandable local flag if the reference can not be resolved.

\property_ref_undefined_warn:

\property_ref_undefined_warn:

LaTeX 2ε-interface: not provided.
The commands triggers the standard warning

```
LaTeX Warning: There were undefined references.
```

at the end of the document if there was a recent \property_ref:nn or \property_ref:nnn which couldn't be resolved and so raised the flag. "Recent" means in the same group or in some outer group!

\property_ref_undefined_warn:n
\property_ref_undefined_warn:e

\property_ref_undefined_warn:n {⟨*label*⟩}

LaTeX 2ε-interface: not provided.
The commands triggers the standard warning

```
LaTeX Warning: There were undefined references.
```

at the end of the document if ⟨*label*⟩ is not known. At the point where it is called it also issues the warning

```
Reference '⟨label⟩' on page ⟨page⟩ undefined.
```

\property_ref_undefined_warn:nn
\property_ref_undefined_warn:ee

\property_ref_undefined_warn:nn {⟨*label*⟩} {⟨*property*⟩}

LaTeX 2ε-interface: see \RefUndefinedWarn.
The commands triggers the standard warning

```
LaTeX Warning: There were undefined references.
```

at the end of the document if the reference can not be resolved. At the point where it is called it also issues the warning

```
Reference '⟨label⟩' on page ⟨page⟩ undefined
```

if the label is unknown, or the more specific

```
Property '⟨property⟩' undefined for reference '⟨label⟩' on page ⟨page⟩
```

if the label is known but doesn't provide a value for the requested property.

\property_if_exist_p:n ⋆ \property_if_exist_p:n {⟨*property*⟩}
\property_if_exist_p:e ⋆ \property_if_exist:nTF {⟨*property*⟩} {⟨*true code*⟩} {⟨*false code*⟩}
\property_if_exist:nTF ⋆ LaTeX 2ε-interface: \IfPropertyExistsTF.
\property_if_exist:eTF ⋆ Tests if the ⟨*property*⟩ has been declared.

\property_if_recorded_p:n ⋆ \property_if_recorded_p:n {⟨*label*⟩}
\property_if_recorded_p:e ⋆ \property_if_recorded:nTF {⟨*label*⟩} {⟨*true code*⟩} {⟨*false code*⟩}
\property_if_recorded:nTF ⋆
\property_if_recorded:eTF ⋆

LaTeX 2ε-interface: \IfLabelExistsTF
Tests if the ⟨*label*⟩ is known. This is also true if the label has been set with the standard \label command.

\property_if_recorded_p:nn ⋆ \property_if_recorded_p:nn {⟨*label*⟩} {⟨*property*⟩}
\property_if_recorded_p:ee ⋆ \property_if_recorded:nnTF {⟨*label*⟩} {⟨*property*⟩} {⟨*true code*⟩} {⟨*false code*⟩}
\property_if_recorded:nnTF ⋆
\property_if_recorded:eeTF ⋆

LaTeX 2ε-interface: \IfPropertyRecordedTF.
Tests if the label ⟨*label*⟩ is known and if it provides a value of the ⟨*property*⟩.

7 Auxiliary file interfaces

\new@label@record \new@label@record {⟨*label*⟩} {⟨*data*⟩}

This is a command only for use in the .aux file. It loads the key–value list of ⟨*data*⟩ to be available for the ⟨*label*⟩.

8 LaTeX 2ε interface

The LaTeXe interfaces always expand label and property arguments. This means that one must be careful when using active chars or commands in the names. UTF8-chars are protected and should be safe, similar most babel shorthands.

\NewProperty \NewProperty {⟨*property*⟩} {⟨*setpoint*⟩} {⟨*default*⟩} {⟨*code*⟩}
\SetProperty \SetProperty {⟨*property*⟩} {⟨*setpoint*⟩} {⟨*default*⟩} {⟨*code*⟩}

Sets the ⟨*property*⟩ to have the ⟨*default*⟩ specified, and at the ⟨*setpoint*⟩ (either now or shipout) to write the result of the ⟨*code*⟩ as part of a label. The ⟨*code*⟩ should be expandable. The expansion of ⟨*code*⟩ (the value of the property) is written to the .aux file and read back from there at the next compilation (at which point normally the standard LaTeX catcode régime with @ a letter is active).

\RecordProperties \RecordProperties {⟨*label*⟩} {⟨*clist*⟩}

Writes the list of properties given by the ⟨*clist*⟩ to the .aux file with the ⟨*label*⟩ specified. Similar to the standard \label command the arguments are expanded. So ⟨*clist*⟩ can be a macro containing a list of properties. Also similar to the standard \label command, the command is surrounded by an \@bsphack/\@esphack pair to preserve spacing.

\RefProperty ⋆ `\RefProperty [⟨local default⟩] {⟨label⟩} {⟨property⟩}`

Expands to the value of the ⟨*property*⟩ for the ⟨*label*⟩, if available, and the default value of the property or – if given – to ⟨*local default*⟩ otherwise. If {⟨*property*⟩} has not been declared an error is issued.

\IfPropertyExistsTF
\IfPropertyExistsT
\IfPropertyExistsF

`\IfPropertyExistsTF {⟨property⟩} {⟨true code⟩} {⟨false code⟩}`

Tests if the ⟨*property*⟩ has been declared.

\IfLabelExistsTF
\IfLabelExistsT
\IfLabelExistsF

`\IfLabelExistsTF {⟨label⟩} {⟨true code⟩} {⟨false code⟩}`

Tests if the ⟨*label*⟩ has been recorded. This is also true if a label has been set with the standard `\label` command.

\IfPropertyRecordedTF
\IfPropertyRecordedT
\IfPropertyRecordedF

`\IfPropertyRecordedTF {⟨label⟩} {⟨property⟩} {⟨true code⟩} {⟨false code⟩}`

Tests if the label and a value of the ⟨*property*⟩ for the ⟨*label*⟩ are both known.

\RefUndefinedWarn `\RefUndefinedWarn {⟨label⟩} {⟨property⟩}`

This commands triggers the standard warning
`LaTeX Warning: There were undefined references.`
at the end of the document if the reference for ⟨*label*⟩ and ⟨*property*⟩ can not be resolved. At the point where it is called it also issues the warning
`Reference '⟨label⟩' on page ⟨page⟩ undefined`
if the label is unknown, or the more specific
`Property '⟨property⟩' undefined for reference '⟨label⟩' on page ⟨page⟩` if the label is known but doesn't provide a value for the requested property.

9 Pre-declared properties

abspage (shipout) The absolute value of the current page: starts at 1 and increases monotonically at each shipout.

page (shipout) The current page as given by `\thepage`: this may or may not be a numerical value, depending on the current style. Contrast with `\abspage`. You get this value also with the standard `\label`/`\pageref`.

pagenum (shipout) The current page as arabic number. This is suitable for integer operations and comparisons.

label (now) The content of `\@currentlabel`. This is the value that you get also with the standard `\label`/`\ref`.

title (now) The content of `\@currentlabelname`. This command is filled beside others by the nameref package and some classes (e.g. memoir).

target (now) The content of `\@currentHref`. This command is normally filled by for example hyperref and gives the name of the last destination it created.

pagetarget (shipout) The content of `\@currentHpage`. This command is filled for example by a recent version of hyperref and then gives the name of the last page destination it created.

counter (now) The content of `\@currentcounter`. This command contains after a `\refstepcounter` the name of the counter.

xpos **ypos** (shipout) This stores the x and y coordinates of a point previously stored with `\pdfsavepos`/`\savepos`. E.g. (if bidi is used it can be necessary to save the position before and after the label):

```
\tex_savepos:D
\property_record:nn{myposition}{xpos,ypos}
\tex_savepos:D
```

10 The Implementation

```
⟨*2ekernel | latexrelease⟩
\ExplSyntaxOn
⟨@@=property⟩
⟨latexrelease⟩\NewModuleRelease{2023/11/01}{ltproperties}
⟨latexrelease⟩                    {Cross-referencing~properties}
```

The approach here is based closely on that from zref; separate out lists of properties and the properties themselves, so the latter can be used multiple times and in varying combinations. However, not everything is a straight copy. Firstly, we treat lists of properties as simple comma lists: that allows us to have either saved or dynamic lists and to avoid another data structure. The cost is that errors are detected at point-of-use, but in any real case that should be true anyway (and is true for `\zref@labelbyprop` already). Secondly, we allow properties to have arbitrary names, as the code does not require them to tokenize as control sequences.

`\property_new:nnnn` `\property_gset:nnnn` `\__property_gset:nnnn`

As properties can be reset, they are not constants. But they also have various pieces of required data. So we use the same approach as color and make them declarations. Data-wise, we need the detail of the implementation, the default and a flag to show if the code works now or at shipout. This last entry is done using text so needs a check. We could use a set of `prop` here, but as we never need to map or copy the lists, we can gain performance using the hash table approach.

```
\cs_new_protected:Npn \property_new:nnnn #1#2#3#4
```

```
  {
    \cs_if_free:cTF { __property_code_ #1 : }
      {
        \exp_args:Nx \__property_gset:nnnn { \tl_to_str:n {#1} }
         {#2} {#3} {#4}
      }
      {
        \msg_error:nn { property }{ exists }{#1}
      }
  }
\cs_new_protected:Npn \property_gset:nnnn #1#2#3#4
  {
    \__property_gset:ennn { \tl_to_str:n {#1} }
      {#2} {#3} {#4}
  }
\cs_new_protected:Npn \__property_gset:nnnn #1#2#3#4
  {
    \cs_gset:cpn { __property_code_ #1 : } {#4}
    \tl_gclear_new:c { g__property_default_ #1 _tl }
    \tl_gset:cn { g__property_default_ #1 _tl } {#3}
    \bool_if_exist:cF { g__property_shipout_ #1 _tl }
      { \bool_new:c { g__property_shipout_ #1 _tl } }
    \str_case:nnF {#2}
      {
        { now } { { \bool_gset_false:c { g__property_shipout_ #1 _tl } } }
        { shipout }
          { \bool_gset_true:c { g__property_shipout_ #1 _tl } }
      }
      { \msg_error:nnnn { property } { unknown-setpoint } {#1} {#2} }
  }
\cs_generate_variant:Nn \__property_gset:nnnn {ennn}
```

(*End of definition for* `\property_new:nnnn`, `\property_gset:nnnn`, *and* `\__property_gset:nnnn`. *These functions are documented on page* 747.)

`\NewProperty`
`\SetProperty`

For consistency we expand the property name, but this doesn't warant a variant of the L3-commands.

```
\cs_new_protected:Npn \NewProperty #1#2#3#4
  {
    \protected@edef\reserved@a{#1}
    \exp_args:No \property_new:nnnn {\reserved@a} {#2}{#3}{#4}
  }
\cs_new_protected:Npn \SetProperty #1#2#3#4
  {
    \protected@edef\reserved@a{#1}
    \exp_args:No \property_gset:nnnn {\reserved@a} {#2}{#3}{#4}
  }
```

(*End of definition for* `\NewProperty` *and* `\SetProperty`. *These functions are documented on page* 749.)

`\property_record:nN`
`\property_record:nn`
`\property_record:nV`
`\property_record:ee`
`\property_record:oo`
`\__property_record:nn`
`\__property_record:en`
`\__property_record_value:n`
`\__property_record_value_aux:n`
`\__property_record_value_aux:e`

Writing data when it is labelled means expanding at this stage and possibly later too. That is all pretty easy using expl3: we accept a stray comma at the end of the list as that is easier to deal with than trying to tidy up, and there is no real downside.

```
\cs_new_protected:Npn \property_record:nN #1#2
```

```
  { \property_record:nV {#1} #2 }
\cs_new_protected:Npn \property_record:nn #1#2
  { \__property_record:en { \tl_to_str:n {#1} } {#2} }
\cs_generate_variant:Nn \property_record:nn { nV , ee, oo }
\cs_new_protected:Npn \__property_record:nn #1#2
  {
    \protected@write \@auxout {}
      {
        \token_to_str:N \new@label@record
          {#1}
          { \clist_map_function:nN {#2} \__property_record_value:n }
      }
  }
\cs_generate_variant:Nn \__property_record:nn { e }
\cs_new:Npn \__property_record_value:n #1
  { \__property_record_value_aux:e { \tl_to_str:n {#1} } }
\cs_new:Npn \__property_record_value_aux:n #1
  {
    \cs_if_exist:cTF { __property_code_ #1 : }
      {
        {#1}
        {
          \bool_if:cTF { g__property_shipout_ #1 _tl }
            { \exp_not:c }
            { \use:c }
              { __property_code_ #1 : }
        }
      }
      { \msg_expandable_error:nnn { property } { not-declared } {#1} }
  }
\cs_generate_variant:Nn \__property_record_value_aux:n { e }
```

(*End of definition for* `\property_record:nN` *and others. These functions are documented on page* 748.)

`\RecordProperties`

```
\NewDocumentCommand\RecordProperties { m m }
  {
    \@bsphack
    \protected@edef\reserved@a{#1}
    \protected@edef\reserved@b{#2}
    \property_record:oo {\reserved@a}{\reserved@b}
    \@esphack
  }
```

(*End of definition for* `\RecordProperties`. *This function is documented on page* 749.)

10.1 Reference commands

`l__property_ref_flag` A flag that is set if a reference couldn't be resolved.

```
\flag_new:n { l__property_ref_flag }
```

(*End of definition for* `l__property_ref_flag`.)

\property_ref:nn
\property_ref:ee

Search for the label/property combination, and if not found fall back to the default of the property.

```
\cs_new:Npn \property_ref:nn #1#2
  {
    \__property_ref:een
      { \tl_to_str:n {#1} }
      { \tl_to_str:n {#2} }
      { \tl_use:c { g__property_default_ #2 _tl } }
  }
\cs_generate_variant:Nn \property_ref:nn {ee}
```

(*End of definition for* \property_ref:nn. *This function is documented on page* 748.)

\property_ref:nnn
\property_ref:een
__property_ref:nnn
__property_ref:een

This allows to set a local default value which overrides the default value of the property.

```
\cs_new:Npn \property_ref:nnn #1#2#3
  {
    \__property_ref:een
      { \tl_to_str:n {#1} }
      { \tl_to_str:n {#2} }
      {#3}
  }
\cs_new:Npn \__property_ref:nnn #1#2#3
  {
    \tl_if_exist:cTF { g__property_label_ #1 _ #2 _tl }
      { \tl_use:c { g__property_label_ #1 _ #2 _tl } }
      {
        \flag_if_raised:nF
          { l__property_ref_flag } { \flag_raise:n { l__property_ref_flag } }
```

We test for the default of the property only to check if the property has been declared.

```
        \tl_if_exist:cTF { g__property_default_ #2 _tl }
          { #3 }
          { \msg_expandable_error:nnn { property } { not-declared } {#2} }
      }
  }
\cs_generate_variant:Nn \__property_ref:nnn { ee }
\cs_generate_variant:Nn \property_ref:nnn {een}
```

(*End of definition for* \property_ref:nnn *and* __property_ref:nnn. *This function is documented on page* 748.)

\RefProperty

Search for the label/property combination, and if not found fall back to the default of the property or the given default.

```
\NewExpandableDocumentCommand \RefProperty { o m m }
  {
    \IfNoValueTF {#1}
      {
        \property_ref:ee {#2}{#3}
      }
      {
        \property_ref:een {#2}{#3}{#1}
      }
  }
```

(*End of definition for* \RefProperty. *This function is documented on page* 750.)

\new@label@record
__property_data:nnn

A standard recursion loop.

```
\cs_new_protected:Npn \new@label@record #1#2
  {
    \tl_if_exist:cTF { r@#1 }
      {
        \gdef \@multiplelabels
          { \@latex@warning@no@line { There~were~multiply~defined~labels } }
        \@latex@warning@no@line { Label~`#1'~multiply~defined }
      }
      {
        \tl_new:c { r@#1 }
        \tl_gset:cn { r@#1 }{#2}
      }
    \__property_data:nnn {#1} #2 { \q_recursion_tail } { ? } \q_recursion_stop
  }
\cs_new_protected:Npn \__property_data:nnn #1#2#3
  {
    \quark_if_recursion_tail_stop:n {#2}
    \tl_gclear_new:c { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl }
    \tl_gset:cn { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl } {#3}
    \__property_data:nnn {#1}
  }
```

This command is used in `\enddocument` to test if some label values have changed.

```
\cs_new_protected:Npn \@kernel@new@label@record@testdef #1 #2
  {
    \tl_if_eq:cnF { r@#1 } {#2}
      { \@tempswatrue }
  }
```

(*End of definition for* `\new@label@record` *and* `\__property_data:nnn`*. This function is documented on page 749.*)

10.2 Tests and warnings

\property_if_exist_p:n
\property_if_exist:n*TF*

Tests if property has been declared.

```
\prg_new_conditional:Npnn \property_if_exist:n #1 { p , T , F,  TF }
  % #1 property
  {
    \cs_if_exist:cTF { __property_code_ #1 : }
      {
        \prg_return_true:
      }
      {
        \prg_return_false:
      }
  }
\prg_generate_conditional_variant:Nnn \property_if_exist:n {e} {TF}
```

(*End of definition for* `\property_if_exist:nTF`*. This function is documented on page 749.*)

\IfPropertyExistsTF
\IfPropertyExistsT
\IfPropertyExistsF

```
\cs_new_eq:NN \IfPropertyExistsTF \property_if_exist:eTF
\cs_new:Npn   \IfPropertyExistsT #1#2 {\property_if_exist:eTF {#1}{#2}{} }
\cs_new:Npn   \IfPropertyExistsF #1   {\property_if_exist:eTF {#1}{} }
```

(*End of definition for* `\IfPropertyExistsTF`, `\IfPropertyExistsT`, *and* `\IfPropertyExistsF`. *These functions are documented on page* 750.)

`\property_if_recorded_p:n`
`\property_if_recorded:nTF`

Tests if the label has been set. This can then be used to setup e.g. rerun messages.

```
\prg_new_conditional:Npnn \property_if_recorded:n #1 { p , T , F,  TF }
  % #1 label
  {
    \tl_if_exist:cTF { r@#1 }
      {
        \prg_return_true:
      }
      {
        \prg_return_false:
      }
  }
\prg_generate_conditional_variant:Nnn \property_if_recorded:n {e} {TF}
```

(*End of definition for* `\property_if_recorded:nTF`. *This function is documented on page* 749.)

`\IfLabelExistsTF`
`\IfLabelExistsT`
`\IfLabelExistsF`

```
\cs_new_eq:NN \IfLabelExistsTF \property_if_recorded:eTF
\cs_new:Npn   \IfLabelExistsT #1#2 {\property_if_recorded:eTF {#1}{#2}{} }
\cs_new:Npn   \IfLabelExistsF #1   {\property_if_recorded:eTF {#1}{} }
```

(*End of definition for* `\IfLabelExistsTF`, `\IfLabelExistsT`, *and* `\IfLabelExistsF`. *These functions are documented on page* 750.)

`\property_if_recorded_p:nn`
`\property_if_recorded:nnTF`

tests if the label/property combination has been set This can then be used to setup e.g. rerun messages.

```
\prg_new_conditional:Npnn \property_if_recorded:nn #1#2 { p , T , F,  TF }
  % #1 label #2 property
  {
    \tl_if_exist:cTF { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl }
      {
        \prg_return_true:
      }
      {
        \prg_return_false:
      }
  }
\prg_generate_conditional_variant:Nnn \property_if_recorded:nn {ee} {TF}
```

(*End of definition for* `\property_if_recorded:nnTF`. *This function is documented on page* 749.)

`\IfPropertyRecordedTF`
`\IfPropertyRecordedT`
`\IfPropertyRecordedF`

```
\cs_new_eq:NN \IfPropertyRecordedTF \property_if_recorded:eeTF
\cs_new:Npn \IfPropertyRecordedT #1#2#3 { \property_if_recorded:eeTF {#1}{#2}{#3}{} }
\cs_new:Npn \IfPropertyRecordedF #1#2#3 { \property_if_recorded:eeTF {#1}{#2}{}{#3} }
```

(*End of definition for* `\IfPropertyRecordedTF`, `\IfPropertyRecordedT`, *and* `\IfPropertyRecordedF`. *These functions are documented on page* 750.)

\property_ref_undefined_warn: `\G@refundefinedtrue` is defined in `ltxref` and redefines a warning message.

```
\cs_new_protected:Npn \property_ref_undefined_warn:
 {
   \flag_if_raised:nT { l__property_ref_flag }
    {
      \G@refundefinedtrue
    }
 }
```

(*End of definition for* `\property_ref_undefined_warn:`. *This function is documented on page 748.*)

\property_ref_undefined_warn:n

```
\cs_new_protected:Npn \property_ref_undefined_warn:n #1 %#1 label
  {
   \property_if_recorded:nF {#1}
    {
       \G@refundefinedtrue
       \@latex@warning{Reference~`#1'~on~page~\thepage\space undefined}%
    }
  }
```

(*End of definition for* `\property_ref_undefined_warn:n`. *This function is documented on page 748.*)

\property_ref_undefined_warn:nn
\property_ref_undefined_warn:ee
\RefUndefinedWarn

```
\cs_new_protected:Npn \property_ref_undefined_warn:nn #1#2 %#1 label, #2 property
 {
   \property_if_recorded:nTF {#1}
    {
      \property_if_recorded:nnF {#1}{#2}
        {
          \G@refundefinedtrue
          \@latex@warning
           { Property~`#2'~undefined~for~reference~`#1'~on~page~\thepage }
        }
    }
    {
       \G@refundefinedtrue
       \@latex@warning { Reference~`#1'~on~page~\thepage\space undefined }%
    }
 }
\cs_generate_variant:Nn \property_ref_undefined_warn:nn {ee}
\cs_set_eq:NN \RefUndefinedWarn \property_ref_undefined_warn:ee
```

(*End of definition for* `\property_ref_undefined_warn:nn` *and* `\RefUndefinedWarn`. *These functions are documented on page 748.*)

10.3 Predeclared properties

abspage

```
\property_new:nnnn { abspage } { shipout }
  { 0 } { \int_use:N \g_shipout_readonly_int }
```

(*End of definition for* `abspage`. *This variable is documented on page 750.*)

page

```
\property_new:nnnn  { page } { shipout } { 0 } { \thepage }
```

(*End of definition for* page. *This variable is documented on page* 750.)

pagenum

```
\property_new:nnnn  { pagenum } { shipout } { 0 } { \the \value { page } }
```

(*End of definition for* pagenum. *This variable is documented on page* 750.)

label

```
\property_new:nnnn  { label } { now } { ?? } { \@currentlabel }
```

(*End of definition for* label. *This variable is documented on page* 750.)

title

```
\property_new:nnnn  { title } { now }
  { \exp_not:n { \textbf { ?? } } } { \@currentlabelname }
```

(*End of definition for* title. *This variable is documented on page* 751.)

target

```
\property_new:nnnn  { target } { now } { } { \@currentHref }
```

(*End of definition for* target. *This variable is documented on page* 751.)

target

```
\newcommand\@currentHpage{}
\property_new:nnnn  { pagetarget } { shipout } { } { \@currentHpage }
```

(*End of definition for* target. *This variable is documented on page* 751.)

counter

```
\property_new:nnnn  { counter } { now }  { } { \@currentcounter }
```

(*End of definition for* counter. *This variable is documented on page* 751.)

xpos
ypos

```
\property_new:nnnn  { xpos } { shipout } { 0} { \int_use:N \tex_lastxpos:D }
\property_new:nnnn  { ypos } { shipout } { 0} { \int_use:N \tex_lastypos:D }
```

(*End of definition for* xpos *and* ypos. *These variables are documented on page* 751.)

10.4 Messages

```
\msg_new:nnnn { property } { exists }
    { Property~'#1'~ has~ already~ been~ declared. }
    { There~ already~ exists~ a~ property~ declaration~ with~ this~
      name.\\
      Please~ use~ a~ different~ name~ for~ your~ property.}

\msg_new:nnnn { property } { not-declared }
  { Property~'#1'~not~declared. }
  {
    LaTeX~has~been~asked~to~use~property~'#1',~but~this~
    name~has~not~been~declared.
  }
\msg_new:nnnn { property } { unknown-setpoint }
  { Unknown~keyword~'#2'~for~setting~property~'#1'. }
  {
    LaTeX~has~been~asked~to~set~the~property~'#1',~but~the~keyword~
    '#2'~is~not~one~of~the~two~known~values:~'now'~or~'shipout'.
  }
```

```
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{ltproperties}
⟨latexrelease⟩                 {cross-referencing~properties~(undo)}%
⟨latexrelease⟩
⟨latexrelease⟩\let \NewProperty  \@undefined
⟨latexrelease⟩\let \SetProperty \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let \RecordProperties  \@undefined
⟨latexrelease⟩\let \RefProperty \@undefined
⟨latexrelease⟩\let \RefUndefinedWarn  \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let \IfPropertyExistsTF \@undefined
⟨latexrelease⟩\let \IfLabelExistsTF \@undefined
⟨latexrelease⟩\let \IfPropertyRecordedTF  \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let\new@label@record \@undefined
⟨latexrelease⟩\let\@kernel@new@label@record@testdef\@undefined
⟨latexrelease⟩\EndModuleRelease
```

```
\ExplSyntaxOff
```

```
⟨/2ekernel | latexrelease⟩
```

Reset module prefix:

```
⟨@@=⟩
```

File 37
ltmiscen.dtx

1 Miscellaneous Environments

This section implements the basic environment mechanism, and also a few specific environments including `document`, The math environments and related commands, the 'flushing' environments, (`center`, `flushleft`, `flushright`), and `verbatim`.

```
⟨*2ekernel⟩
\message{environments,}
```

1.1 Environments

`\begin{foo}` and `\end{foo}` are used to delimit environment `foo`.

`\begin{foo}` starts a group and calls `\foo` if it is defined, otherwise it does nothing.

`\end{foo}` checks to see that it matches the corresponding `\begin` and if so, it calls `\endfoo` and does an `\endgroup`. Otherwise, `\end{foo}` does nothing.

If `\end{foo}` needs to ignore blanks after it, then `\endfoo` should globally set the `@ignore` switch true with `\@ignoretrue` (this will automatically be global).

NOTE: `\@@end` is defined to be the `\end` command of TEX82.

`\enddocument` is the user's command for ending the manuscript file.

`\stop` is a panic button — to end TEX in the middle.

Historical LATEX 2.09 comments (not necessarily accurate any more):

```
\enddocument ==
  BEGIN
   \@checkend{document}    %% checks for unmatched \begin
   \clearpage
   \begingroup
     if @filesw = true
       then  close file @mainaux
             if G@refundefined = true
              then LaTeX Warning: 'There are undefined references.' fi
             if @multiplelabels = true
               then LaTeX Warning:
                    'One or more label(s) multiply defined.'
               else
               \@setckpt {ARG1}{ARG2} == null
               \newlabel{LABEL}{VAL} ==
                    BEGIN
                      \reserved@a == VAL
                      if def(\reserved@a) = def(\r@LABEL)
                        else @tempswa := true            fi
                    END
               \bibcite{LABEL}{VAL} == null
                    BEGIN
                      \reserved@a == VAL
                      if def(\reserved@a) = def(\g@LABEL)
                        else @tempswa := true            fi
```

```
                    END
                @tempswa := false
                make @ a letter
                \input \jobname.AUX
                if @tempswa = true
                  then LaTeX Warning: 'Label may have changed.
                                      Rerun to get cross-references right.'
        fi      fi      fi
    \endgroup
    finish up
  END

  \@writefile{EXT}{ENTRY} ==
      if tf@EXT undefined
        else \write\tf@EXT{ENTRY}
      fi
```

End of historical LaTeX 2.09 comments.

`\@currenvir` The name of the current environment. Initialized to `document` to so that `\end{document}` works correctly.

```
\def\@currenvir{document}
```

(*End of definition for* `\@currenvir`.)

`\if@ignore`
`\@ignoretrue`
`\@ignorefalse`

```
\def\@ignorefalse{\global\let\if@ignore\iffalse}
\def\@ignoretrue {\global\let\if@ignore\iftrue}
\@ignorefalse
```

(*End of definition for* `\if@ignore`, `\@ignoretrue`, *and* `\@ignorefalse`.)

`\ignorespacesafterend`

```
\let\ignorespacesafterend\@ignoretrue
```

(*End of definition for* `\ignorespacesafterend`.)

`\enddocument` (*end*)

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                    {\enddocument}{check property labels}%
\def\enddocument{%
```

The `\end{document}` hook is executed first. If necessary it can contain a `\clearpage` to output dangling floats first. In this position it can also contain something like `\end{foo}` so that the whole document effectively starts and ends with some special environment. However, this must be used with care, eg if two applications would use this without knowledge of each other the order of the environments will be wrong after all. `\AtEndDocument` is redefined at this point so that and such commands that get into the hook do not chase their tail...

```
    \@kernel@before@enddocument
    \UseOneTimeHook{enddocument}%
    \@kernel@after@enddocument
```

```
  \@checkend{document}%
  \clearpage
  \UseOneTimeHook{enddocument/afterlastpage}%
  \@kernel@after@enddocument@afterlastpage
  \begingroup
    \if@filesw
      \immediate\closeout\@mainaux
      \let\@setckpt\@gobbletwo
      \let\@newl@bel\@testdef
      \let\new@label@record\@kernel@new@label@record@testdef
```

The previous line is equiv to setting

```
\def\newlabel{\@testdef r}%
\def\bibcite{\@testdef b}%
```

We use `\@@input` to load the `.aux` file, so that it doesn't show up in the list of files produced by `\listfiles`.

```
      \@tempswafalse
      \makeatletter \@@input\jobname.aux
    \fi
    \UseOneTimeHook{enddocument/afteraux}%
```

Next hook is expect to contain only code for writing info messages on the terminal.

```
    \UseOneTimeHook{enddocument/info}%
  \endgroup
  \UseOneTimeHook{enddocument/end}%
  \deadcycles\z@\@@end}
```

The public hooks used in `\enddocument`:

```
\NewHook{enddocument}
\NewHook{enddocument/afterlastpage}
\NewHook{enddocument/afteraux}
\NewHook{enddocument/info}
\NewHook{enddocument/end}
```

This is one of the few places where we already add data and rules to a hook already in the kernel.

If we roll back we have to drop stuff before adding chunks, otherwise the code will just be appended, and thus doubled. This would result in a harmless warning during the format generation, because in that case the code chunk label doesn't exist, and therefore can't be dropped.

```
⟨latexrelease⟩\RemoveFromHook{enddocument/info}[kernel/filelist]
⟨latexrelease⟩\RemoveFromHook{enddocument/info}[kernel/warnings]
⟨latexrelease⟩\RemoveFromHook{enddocument/info}[kernel/release]
\AddToHook{enddocument/info}[kernel/filelist]{\@dofilelist}
\AddToHook{enddocument/info}[kernel/warnings]{\@enddocument@kernel@warnings}
\AddToHook{enddocument/info}[kernel/release]{%
  \let\show@release@info\wlog
  \show@release@info{ ***********}%
  \the\LaTeXReleaseInfo
  \show@release@info{ ***********}}

\DeclareHookRule{enddocument/info}{kernel/release}{before}{kernel/filelist}
\DeclareHookRule{enddocument/info}{kernel/filelist}{before}{kernel/warnings}
```

(*End of definition for* `\enddocument`.)

`\@enddocument@kernel@warnings`

```
\def\@enddocument@kernel@warnings{%
```

First we check for font size substitution bigger than `\fontsubfuzz`. The `\relax` is necessary because this is a macro not a register.

```
  \ifdim \font@submax >\fontsubfuzz\relax
```

In case you wonder about the `\@gobbletwo` inside the message below, this is a horrible hack to remove the tokens `\on@line.` that are added by `\@font@warning` at the end.

```
    \@font@warning{Size substitutions with differences\MessageBreak
                 up to \font@submax\space have occurred.\@gobbletwo}%
  \fi
```

The macro `\@defaultsubs` is initially `\relax` but gets redefined to produce a warning if there have been some default font substitutions.

```
  \@defaultsubs
```

The macro `\@refundefined` is initially `\relax` but gets redefined to produce a warning if there are undefined refs.

```
  \@refundefined
```

If a label is defined more than once, `\@tempswa` will always be true and thus produce a "Label(s) may ..." warning. But since a rerun will not solve that problem (unless one uses a package like `varioref` that generates labels on the fly), we suppress this message.

```
  \if@filesw
    \ifx \@multiplelabels \relax
      \if@tempswa
        \@latex@warning@no@line{Label(s) may have changed.
            Rerun to get cross-references right}%
      \fi
    \else
      \@multiplelabels
    \fi
    \ifx \@extra@page@added \relax
        \@latex@warning@no@line{Temporary extra page added at the end.
            Rerun to get it removed}%
    \fi
```

We could think of adding a warning that nothing can be corrected while `\nofiles` is in force. In the past the warnings related to the `aux` file are simply suppressed in this case.

```
  \fi
}
```

(*End of definition for* `\@enddocument@kernel@warnings`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\enddocument}{Use Hooks}%
⟨latexrelease⟩\def\enddocument{%
⟨latexrelease⟩    \@kernel@before@enddocument
⟨latexrelease⟩    \UseOneTimeHook{enddocument}%
⟨latexrelease⟩    \@kernel@after@enddocument
```

```
⟨latexrelease⟩   \@checkend{document}%
⟨latexrelease⟩   \clearpage
⟨latexrelease⟩   \UseOneTimeHook{enddocument/afterlastpage}%
⟨latexrelease⟩   \@kernel@after@enddocument@afterlastpage
⟨latexrelease⟩   \begingroup
⟨latexrelease⟩     \if@filesw
⟨latexrelease⟩       \immediate\closeout\@mainaux
⟨latexrelease⟩       \let\@setckpt\@gobbletwo
⟨latexrelease⟩       \let\@newl@bel\@testdef
⟨latexrelease⟩       \@tempswafalse
⟨latexrelease⟩       \makeatletter \@@input\jobname.aux
⟨latexrelease⟩     \fi
⟨latexrelease⟩     \UseOneTimeHook{enddocument/afteraux}%
⟨latexrelease⟩     \UseOneTimeHook{enddocument/info}%
⟨latexrelease⟩   \endgroup
⟨latexrelease⟩   \UseOneTimeHook{enddocument/end}%
⟨latexrelease⟩   \deadcycles\z@\@@end}
⟨latexrelease⟩\NewHook{enddocument}
⟨latexrelease⟩\NewHook{enddocument/afterlastpage}
⟨latexrelease⟩\NewHook{enddocument/afteraux}
⟨latexrelease⟩\NewHook{enddocument/info}
⟨latexrelease⟩\NewHook{enddocument/end}
```

If we roll back we have to drop stuff before adding chunks, otherwise the code will just be appended, and thus doubled.

```
⟨latexrelease⟩\RemoveFromHook{enddocument/info}[kernel/filelist]
⟨latexrelease⟩\RemoveFromHook{enddocument/info}[kernel/warnings]
⟨latexrelease⟩\RemoveFromHook{enddocument/info}[kernel/release]
```

```
⟨latexrelease⟩\AddToHook{enddocument/info}[kernel/filelist]{\@dofilelist}
⟨latexrelease⟩\AddToHook{enddocument/info}[kernel/warnings]{\@enddocument@kernel@warnings}
⟨latexrelease⟩\AddToHook{enddocument/info}[kernel/release]{%
⟨latexrelease⟩   \let\show@release@info\wlog
⟨latexrelease⟩   \show@release@info{ ***********}%
⟨latexrelease⟩   \the\LaTeXReleaseInfo
⟨latexrelease⟩   \show@release@info{ ***********}}
⟨latexrelease⟩
⟨latexrelease⟩\DeclareHookRule{enddocument/info}{kernel/release}{before}{kernel/filelist}
⟨latexrelease⟩\DeclareHookRule{enddocument/info}{kernel/filelist}{before}{kernel/warnings}
⟨latexrelease⟩\def\@enddocument@kernel@warnings{%
⟨latexrelease⟩   \ifdim \font@submax >\fontsubfuzz\relax
⟨latexrelease⟩     \@font@warning{Size substitutions with differences\MessageBreak
⟨latexrelease⟩                 up to \font@submax\space have occurred.\@gobbletwo}%
⟨latexrelease⟩   \fi
⟨latexrelease⟩   \@defaultsubs
⟨latexrelease⟩   \@refundefined
⟨latexrelease⟩   \if@filesw
⟨latexrelease⟩     \ifx \@multiplelabels \relax
⟨latexrelease⟩       \if@tempswa
⟨latexrelease⟩         \@latex@warning@no@line{Label(s) may have changed.
⟨latexrelease⟩             Rerun to get cross-references right}%
⟨latexrelease⟩       \fi
⟨latexrelease⟩     \else
⟨latexrelease⟩       \@multiplelabels
⟨latexrelease⟩     \fi
```

```
⟨latexrelease⟩     \ifx \@extra@page@added \relax
⟨latexrelease⟩         \@latex@warning@no@line{Temporary extra page added at the end.
⟨latexrelease⟩             Rerun to get it removed}%
⟨latexrelease⟩     \fi
⟨latexrelease⟩   \fi
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\enddocument}{Use Hooks}%
⟨latexrelease⟩
⟨latexrelease⟩\def\enddocument{%
⟨latexrelease⟩   \let\AtEndDocument\@firstofone
⟨latexrelease⟩   \@enddocumenthook
⟨latexrelease⟩   \@checkend{document}%
⟨latexrelease⟩   \clearpage
⟨latexrelease⟩   \begingroup
⟨latexrelease⟩     \if@filesw
⟨latexrelease⟩       \immediate\closeout\@mainaux
⟨latexrelease⟩       \let\@setckpt\@gobbletwo
⟨latexrelease⟩       \let\@newl@bel\@testdef
⟨latexrelease⟩       \@tempswafalse
⟨latexrelease⟩       \makeatletter \@@input\jobname.aux
⟨latexrelease⟩     \fi
⟨latexrelease⟩     \@dofilelist
⟨latexrelease⟩     \ifdim \font@submax >\fontsubfuzz\relax
⟨latexrelease⟩       \@font@warning{Size substitutions with differences\MessageBreak
⟨latexrelease⟩                  up to \font@submax\space have occurred.\@gobbletwo}%
⟨latexrelease⟩     \fi
⟨latexrelease⟩     \@defaultsubs
⟨latexrelease⟩     \@refundefined
⟨latexrelease⟩     \if@filesw
⟨latexrelease⟩       \ifx \@multiplelabels \relax
⟨latexrelease⟩         \if@tempswa
⟨latexrelease⟩           \@latex@warning@no@line{Label(s) may have changed.
⟨latexrelease⟩               Rerun to get cross-references right}%
⟨latexrelease⟩         \fi
⟨latexrelease⟩       \else
⟨latexrelease⟩         \@multiplelabels
⟨latexrelease⟩       \fi
⟨latexrelease⟩     \fi
⟨latexrelease⟩   \endgroup
⟨latexrelease⟩   \deadcycles\z@\@@end}
⟨latexrelease⟩
⟨latexrelease⟩\let\@enddocument@kernel@warnings\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\@kernel@before@enddocument` The `\@kernel@before@enddocument` hook is slightly different because we initialize it with `\par` so that `\enddocument` always returns to vertical mode as its first action.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                 {\@kernel@before@enddocument}{kernel before hook}%
```

```
\def\@kernel@before@enddocument{\par}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

The rollback code renders it harmless.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@kernel@before@enddocument}{kernel before hook}%
⟨latexrelease⟩
⟨latexrelease⟩\let\@kernel@before@enddocument\@empty
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@kernel@before@enddocument`.)

`\@testdef`

```
\def\@testdef #1#2#3{%
  \def\reserved@a{#3}\expandafter \ifx \csname #1@#2\endcsname
 \reserved@a  \else \@tempswatrue \fi}
```

(*End of definition for* `\@testdef`.)

Reading data from auxiliary files (like `.toc` normally happens in vertical mode and it therefore doesn't matter if line endings are converted to spaces by TEX during that process.

However, especially the `.toc` file might be read in L-R mode (in cases the `\tableofcontents` attempts to put, say, a list of sub-sections as a paragraph). In that case the newlines after a line like

```
\contentsline {subsubsection}{\numberline {1.1.1}A C-head}{2}
```

might result in spurious spaces (e.g., when that level is not included).

That could be fixed by reading in the file using `\endlinechar=-1` but that has the danger that it drops some valid endlines that should be converted to spaces (for example, when the user edited the TOC and then used `\nofiles` to preserve it).

So the approach taken instead is this:

- `\addcontentsline` adds the command `\protected@file@percent` to the end of the second argument of `\@writefile` that is written to the `.aux`. As the name indicates this is a protected macro so it doesn't change if it is written out.
- When the `.aux` is read back in at the end of the run, `\@writefile` is executed and writes its second argument unmodified to the file with the extension given by its first argument. Or rather that was how it was in the past.
- Instead we change `\@writefile` slightly: basically it looks at the second argument and if the last token in there is `\protected@file@percent` then it is replaced by a percent character and that is then written out. If not (for example, if the data came from a user issued `\addtocontents`, or from some package that uses `\@writefile` for writing its own files) then the command behaves exactly as before.

`\protected@file@percent` Dummy cs to be replaced by a percent sign inside `\@writefile`. If it survives (when used incorrectly) it will expand to nothing in a typsetting context.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                  {\protected@file@percent}{Mask line endings}%
\protected\def\protected@file@percent{}
```

(*End of definition for* `\protected@file@percent`.)

`\add@percent@to@temptokena` Helper function which is used to inspect a sequence of tokens (the second argument of `\@writefile` and it the last token is `\protected@file@percent` it will replace it by a harmless percent. The result is saved in `\@temptokena` for later use.

```
\catcode`\^^A=9
\long\gdef\add@percent@to@temptokena
    #1\protected@file@percent#2\add@percent@to@temptokena
```

When we call this macro in `\@writefile` we stick in `\@empty` at the beginning, so that in case the tokenlist consists of a single brace group the braces aren't stripped. The `\expandafter` then expands this extra token away again.

```
    {\expandafter\ifx\expandafter X\detokenize{#2}X\expandafter\dont@add@percent@to@temptoken
             \expandafter\do@add@percent@to@temptokena\fi{#1}}
```

```
\long\def\dont@add@percent@to@temptokena#1{%
  \@temptokena\expandafter{#1}}
```

`latexrelease` will read this code in high-speed mode in certain situations. During that it will only look for `\if` tests but not actually execute the `\catcode` change above. As a result it will drop anything after the `%` character in the definition. Therefore the `\fi` needs to be on the next line and we need locally another comment character to avoid getting spaces into the definition—a weird problem :-)

```
\begingroup
\catcode`\%=12
\catcode`\^^A=14
\long\gdef\do@add@percent@to@temptokena#1{\@temptokena\expandafter{#1%^^A
```

Can't be on the same line as the `%` — see above.

```
  }}
\endgroup
```

(*End of definition for* `\add@percent@to@temptokena`.)

`\@writefile`

```
\long\def\@writefile#1#2{%
  \@ifundefined{tf@#1}\relax
    {%
```

If we write to the file we first prepare `#2` using `\add@percent@to@temptokena` and then write the token register out.

```
      \add@percent@to@temptokena
        \@empty#2\protected@file@percent
        \add@percent@to@temptokena
     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
    }%
}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\protected@file@percent}{Mask line endings}%
⟨latexrelease⟩\let\protected@file@percent\@undefined
⟨latexrelease⟩\let\add@percent@to@temptokena\@undefined
⟨latexrelease⟩\let\do@add@percent@to@temptokena\@undefined
⟨latexrelease⟩\let\dont@add@percent@to@temptokena\@undefined
⟨latexrelease⟩\long\def\@writefile#1#2{%
⟨latexrelease⟩  \@ifundefined{tf@#1}\relax
⟨latexrelease⟩    {\@temptokena{#2}%
⟨latexrelease⟩     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
⟨latexrelease⟩    }%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@writefile`.)

`\stop`

```
\def\stop{\clearpage\deadcycles\z@\let\par\@@par\@@end}
```

(*End of definition for* `\stop`.)

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\everypar{\@nodocument} %% To get an error if text appears before the
\nullfont               %% \begin{document}
```

`\begin`, `\end`, and `\@checkend` changed so `\end`{document} will catch an unmatched `\begin`. Changed 24 May 89 as suggested by Frank Mittelbach and Rainer Sch\"opf.

```
\begin{NAME} ==
 BEGIN
   IF \NAME undefined   THEN   \reserved@a == BEGIN report error END
                         ELSE  \reserved@a ==
                                   (\@currenvir :=L NAME) \NAME
   FI
   @ignore :=G F        %% Added 30 Nov 88
   \begingroup
   \@endpe := F
   \@currenvir :=L NAME
   \NAME
 END

\end{NAME} ==
 BEGIN
  \endNAME
  \@checkend{NAME}
  \endgroup
  IF @endpe = T                %% @endpe set True by \@endparenv
```

```
      THEN \@doendpe            %% \@doendpe redefines \par and \everypar
                                %% to suppress paragraph indentation in
    FI                          %% immediately following text
    IF @ignore = T
      THEN @ignore :=G F
           \ignorespaces
    FI
  END

 \@checkend{NAME} ==
  BEGIN
   IF \@currenvir = NAME
     ELSE \@badend{NAME}
   FI
  END
```

End of historical LaTeX 2.09 comments.

\begin

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\begin}{Use hook system}%
\protected\def\begin#1{%
  \UseHook{env/#1/before}%
  \@ifundefined{#1}%
    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
    {\def\reserved@a{\def\@currenvir{#1}%
        \edef\@currenvline{\on@line}%
        \@execute@begin@hook{#1}%
        \csname #1\endcsname}}%
  \@ignorefalse
  \begingroup\@endpefalse\reserved@a}
```

Before the \document code is executed we have to first undo the \endgroup as there should be none for this environment to avoid that changes on top-level unnecessarily go to TeX's savestack, and we have to initialize all hooks in the hook system. So we need to test for this environment name. But once it has be found all this testing is no longer needed and so we redefine \@execute@begin@hook to simply use the hook.

```
\def\@execute@begin@hook #1{%
  \expandafter\ifx\csname #1\endcsname\document
    \endgroup
    \gdef\@execute@begin@hook##1{\UseHook{env/##1/begin}}%
    \@expl@@@initialize@all@@
  \fi
```

If this is an environment before \begin{document} we just run the hook so this can be outside the test.

```
  \UseHook{env/#1/begin}%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\begin}{Making \begin/\end robust}%
⟨latexrelease⟩\DeclareRobustCommand\begin[1]{%
⟨latexrelease⟩  \@ifundefined{#1}%
⟨latexrelease⟩    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
⟨latexrelease⟩    {\def\reserved@a{\def\@currenvir{#1}%
⟨latexrelease⟩     \edef\@currenvline{\on@line}%
⟨latexrelease⟩     \csname #1\endcsname}}%
⟨latexrelease⟩  \@ignorefalse
⟨latexrelease⟩  \begingroup\@endpefalse\reserved@a}
⟨latexrelease⟩\EndIncludeInRelease
```

A version that doesn't start out with `\relax` when in typesetting mode would be the following, but since `\begin` issues a `\begingroup` it wouldn't help much with respect to allowing things like `\noalign` or `\multicolumn` inside.

```
%\edef\begin
%  {\unexpanded{%
%       \ifx\protect\@typeset@protect
%         \expandafter\@gobble
%       \fi
%       \protect
%     }%
%   \expandafter\noexpand\csname begin \endcsname
%  }
%\@namedef{begin }#1{%
%  \@ifundefined{#1}%
%    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
%    {\def\reserved@a{\def\@currenvir{#1}%
%     \edef\@currenvline{\on@line}%
%     \csname #1\endcsname}}%
%  \@ignorefalse
%  \begingroup\@endpefalse\reserved@a}
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\begin}{Making \begin/\end robust}%
⟨latexrelease⟩\def\begin#1{%
⟨latexrelease⟩  \@ifundefined{#1}%
⟨latexrelease⟩    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
⟨latexrelease⟩    {\def\reserved@a{\def\@currenvir{#1}%
⟨latexrelease⟩     \edef\@currenvline{\on@line}%
⟨latexrelease⟩     \csname #1\endcsname}}%
⟨latexrelease⟩  \@ignorefalse
⟨latexrelease⟩  \begingroup\@endpefalse\reserved@a}
⟨latexrelease⟩
```

Also undo the internal commands as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```
⟨latexrelease⟩\expandafter\let\csname begin \endcsname\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\begin`.)

`\end` The top level definition for `\end`.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\end}{Making \begin/\end robust}%
```

While `\begin` was made robust simply by using `\DeclareRobustCommand` we need to be a bit more subtle with `\end` as there are packages out there that try to look into the top-level contents of `\end{foo}` (that is at the expansion of `\endfoo`) to see if it contains certain macros. This is done by hitting `\end{foo}` with three `\expandafter`s, the first to get

```
\csname endfoo\endcsname        \@checkend{foo}% etc.
```

the second to expand the `\csname`, i.e., to get to

```
\endfoo                         \@checkend{foo}% etc.
```

and the third to finally get to the top-level content of `\endfoo`, i.e.

```
<top-level content of \endfoo> \@checkend{foo}% etc.
```

Therefore a robust replacement should produce the same results after three expansions (there first is obviously different).

Basically the definition of `\end` should either produce `\protect\end␣` (when not doing typesetting) or it should produce `\end␣` (without the `\protect`) when doing typesetting. Furthermore, it should (when in typesetting mode) show exactly the same result as `\end␣` (which is the original fragile definition of `\end`) when you expand either of them twice, i.e.,

```
\endfoo                         \@checkend{foo}% etc.
```

That is achieved with the code below (which is worth studying carefully).

There is some trickery involved here: in particular we use `\romannumeral` to change a single expansion into three successive expansions in one go. That primitive expands until it has scanned a number (0 in this case, so it doesn't produce any output) and so it allows us to place arbitrary many `\expandafter`s inside that are all going to be executed when `\romannumeral` is hit by a single `\expandafter`.

```
\edef\end
  {\unexpanded{%
     \romannumeral
       \ifx\protect\@typeset@protect
       \expandafter       %1
         \expandafter        %2
       \expandafter       %1
           \expandafter         %3 expands the \csname inside \end<space>
       \expandafter       %1
         \expandafter        %2  expands \end<space>
       \expandafter       %1     expands the \else
           \z@
       \else
         \expandafter\z@\expandafter\protect
       \fi
   }%
   \expandafter\noexpand\csname end \endcsname
  }
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

And here is the original definition of `\end` the way it was in LaTeX for several decades now hidden in `\end␣`.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\end}{Making \begin/\end robust}%
⟨latexrelease⟩\def\end#1{%
⟨latexrelease⟩  \csname end#1\endcsname\@checkend{#1}%
⟨latexrelease⟩  \expandafter\endgroup\if@endpe\@doendpe\fi
⟨latexrelease⟩  \if@ignore\@ignorefalse\ignorespaces\fi}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\end`.)

`\end\verbvisiblespace` The internal version with a space at the end.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/11/01}%
⟨latexrelease⟩                  {\end!space}{New @endpe handling}%
\@namedef{end }#1{%
  \romannumeral
    \IfHookEmptyTF{env/#1/end}%
        {\expandafter\z@}%
        {\z@\UseHook{env/#1/end}}%
    \csname end#1\endcsname\@checkend{#1}%
```

We can now close the environment group and due to the new `\if@endpe` handling we no longer need to `\expandafter` out of the group.

```
%    \expandafter\endgroup\if@endpe\@doendpe\fi
    \endgroup
    \UseHook{env/#1/after}%
    \if@ignore\@ignorefalse\ignorespaces\fi
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

Version that adds hooks (so different from the 2019 version). It fixes tlb3722 but the change should perhaps be made in `tabularx` instead.

```
%    \begin{macrocode}
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\end!space}{Use hook system}%
⟨latexrelease⟩
⟨latexrelease⟩\@namedef{end }#1{%
⟨latexrelease⟩  \romannumeral
⟨latexrelease⟩    \IfHookEmptyTF{env/#1/end}%
⟨latexrelease⟩        {\expandafter\z@}%
⟨latexrelease⟩        {\z@\UseHook{env/#1/end}}%
⟨latexrelease⟩    \csname end#1\endcsname\@checkend{#1}%
⟨latexrelease⟩    \expandafter\endgroup\if@endpe\@doendpe\fi
⟨latexrelease⟩    \UseHook{env/#1/after}%
⟨latexrelease⟩    \if@ignore\@ignorefalse\ignorespaces\fi
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
```

Version without the fix for tlb3722 for the record:

```
@namedef{end }#1{%
 \UseHook{env/#1/end}%
 \csname end#1\endcsname\@checkend{#1}%
 \expandafter\endgroup\if@endpe\@doendpe\fi
 \UseHook{env/#1/after}%
 \if@ignore\@ignorefalse\ignorespaces\fi}%
```

```
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                {\end!space}{Making \begin/\end robust}%
⟨latexrelease⟩
⟨latexrelease⟩\@namedef{end }#1{%
⟨latexrelease⟩  \csname end#1\endcsname\@checkend{#1}%
⟨latexrelease⟩  \expandafter\endgroup\if@endpe\@doendpe\fi
⟨latexrelease⟩  \if@ignore\@ignorefalse\ignorespaces\fi}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                {\end!space}{Making \begin/\end robust}%
```

Undo the internal command as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```
⟨latexrelease⟩\expandafter\let\csname end \endcsname\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\end\verbvisiblespace`.)

`\@checkend`

```
\def\@checkend#1{\def\reserved@a{#1}\ifx
      \reserved@a\@currenvir \else\@badend{#1}\fi}
```

(*End of definition for* `\@checkend`.)

`\@currenvline`

We do need a default value for `\@currenvline` on top-level since the document environment cancels the brace group. This means that a mismatch with `\begin{document}` will not produce a line number. Thus the outer default must be `\@empty` or we will end up with two spaces.

```
\let\@currenvline\@empty
```

(*End of definition for* `\@currenvline`.)

`\AtBeginEnvironment`
`\AtEndEnvironment`
`\BeforeBeginEnvironment`
`\AfterEndEnvironment`

We provide 4 high-level hook interfaces directly, the others only when etoolbox is loaded

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                {\AtBeginEnvironment}{Hooks for environments}%
```

```
\newcommand\AtBeginEnvironment[2][.]    {\AddToHook{env/#2/begin}[#1]}
\newcommand\AtEndEnvironment[2][.]      {\AddToHook{env/#2/end}[#1]}
\newcommand\BeforeBeginEnvironment[2][.]{\AddToHook{env/#2/before}[#1]}
\newcommand\AfterEndEnvironment[2][.]   {\AddToHook{env/#2/after}[#1]}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\AtBeginEnvironment}{Hooks for environments}%
⟨latexrelease⟩
⟨latexrelease⟩\let\AtBeginEnvironment\@undefined
⟨latexrelease⟩\let\AtEndEnvironment\@undefined
⟨latexrelease⟩\let\BeforeBeginEnvironment\@undefined
⟨latexrelease⟩\let\AfterEndEnvironment\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\AtBeginEnvironment` *and others. These functions are documented on page 218.*)

1.2 Center, Flushright, Flushleft

```
\message{center,}
```

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\center, \flushright and \flushleft set
  \rightskip = 0pt or \@flushglue (as appropriate)
  \leftskip  = 0pt or \@flushglue (as appropriate)
  \parindent = 0pt
  \parfillskip   = 0pt. (except \flushleft)
  \\          == \par \vskip -\parskip
  \\[LENGTH] == \\ \vskip LENGTH
  \\*         == \par \penalty 10000 \vskip -\parskip
  \\*[LEN]    == \\* \vskip LENGTH

They invoke the trivlist environment to handle vertical spacing before
and after them.

\centering, \raggedright and \raggedleft are the declaration analogs
of the above.

\raggedright has a more universal effect, however.  It sets
\@rightskip := flushglue.  Every environment, like the list
environments,
that set \rightskip to its 'normal' value set it to \@rightskip
```

End of historical LaTeX 2.09 comments.

`\@centercr`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                  {\@centercr}{Make robust}%
\protected\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
        \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@centercr}{Make robust}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
⟨latexrelease⟩        \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@centercr`.)

`\@xcentercr`

```
\def\@xcentercr{\addvspace{-\parskip}\@ifnextchar
    [\@icentercr\ignorespaces}
```

(*End of definition for* `\@xcentercr`.)

`\@icentercr`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@icentercr}{centering, etc support calc}%
\def\@icentercr[#1]{\@vspace@calcify{#1}\ignorespaces}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@icentercr}{centering, etc support calc}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@icentercr[#1]{\vskip #1\ignorespaces}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@icentercr`.)

`center` (*env.*) We use `\relax` to prevent `\item` scanning too far.

```
\def\center{\trivlist \centering\item\relax}
```

```
\def\endcenter{\endtrivlist}
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\centering}{Set finaldhypendemerits}%
```

`\centering`

```
\DeclareRobustCommand\centering{%
  \let\\\@centercr
  \rightskip\@flushglue\leftskip\@flushglue
  \finalhyphendemerits=\z@
  \parindent\z@\parfillskip\z@skip}
```

(*End of definition for* `\centering`.)

\raggedright

```
\DeclareRobustCommand\raggedright{%
  \let\\\@centercr\@rightskip\@flushglue \rightskip\@rightskip
  \finalhyphendemerits=\z@
  \leftskip\z@skip
  \parindent\z@}
```

(*End of definition for* \raggedright.)

\raggedleft

```
\DeclareRobustCommand\raggedleft{%
  \let\\\@centercr
  \rightskip\z@skip\leftskip\@flushglue
  \finalhyphendemerits=\z@
  \parindent\z@\parfillskip\z@skip}
```

(*End of definition for* \raggedleft.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\centering}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\centering{%
⟨latexrelease⟩  \let\\\@centercr
⟨latexrelease⟩  \rightskip\@flushglue\leftskip\@flushglue
⟨latexrelease⟩  \parindent\z@\parfillskip\z@skip}
⟨latexrelease⟩\DeclareRobustCommand\raggedright{%
⟨latexrelease⟩  \let\\\@centercr\@rightskip\@flushglue \rightskip\@rightskip
⟨latexrelease⟩  \leftskip\z@skip
⟨latexrelease⟩  \parindent\z@}
⟨latexrelease⟩\DeclareRobustCommand\raggedleft{%
⟨latexrelease⟩  \let\\\@centercr
⟨latexrelease⟩  \rightskip\z@skip\leftskip\@flushglue
⟨latexrelease⟩  \parindent\z@\parfillskip\z@skip}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\centering}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\centering
⟨latexrelease⟩\kernel@make@fragile\raggedright
⟨latexrelease⟩\kernel@make@fragile\raggedleft
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@rightskip

```
\newskip\@rightskip \@rightskip \z@skip
```

(*End of definition for* \@rightskip.)

flushleft (*env.*) We use \relax to prevent \item scanning too far.

```
\def\flushleft{\trivlist \raggedright\item\relax}
\def\endflushleft{\endtrivlist}
```

`flushright` (*env.*) We use `\relax` to prevent `\item` scanning too far.

```
\def\flushright{\trivlist \raggedleft\item\relax}
\def\endflushright{\endtrivlist}
```

1.3 Verbatim

```
\message{verbatim,}
```

The verbatim environment uses the fixed-width `\ttfamily` font, turns blanks into spaces, starts a new line for each carriage return (or sequence of consecutive carriage returns), and interprets *every* character literally. I.e., all special characters `\, {, $`, etc. are `\catcode`'d to 'other'.

The command `\verb` produces in-line verbatim text, where the argument is delimited by any pair of characters. E.g., `\verb #...#` takes '`...`' as its argument, and sets it verbatim in `\ttfamily` font.

The *-variants of these commands are the same, except that spaces print as the TEXbook's space character instead of as blank spaces.

`\@vobeyspaces`
`\@vobeytabs`

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                  {\@vobeytabs}{Obeyed tabs}%
⟨*2ekernel | latexrelease⟩
{\catcode`\ =\active%
\gdef\@vobeyspaces{\catcode`\ \active\let \@xobeysp\@vobeytabs}}
{\catcode`\^^I=\active
\gdef\@vobeytabs{\catcode`\^^I\active\let^^I\@xobeytab}%
\global\let^^I=\space
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@vobeytabs}{Obeyed tabs}%
⟨latexrelease⟩{\catcode`\ =\active%
⟨latexrelease⟩\gdef\@vobeyspaces{\catcode`\ \active\let \@xobeysp}}
⟨latexrelease⟩\let\@vobeytabs\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@vobeyspaces` *and* `\@vobeytabs`.)

`\@xobeysp`

(*End of definition for* `\@xobeysp`.)

`\@xverbatim`
`\@sxverbatim`

```
\begingroup \catcode `|=0 \catcode `[= 1
\catcode`]=2 \catcode `\{=12 \catcode `\}=12
\catcode`\\=12 |gdef|@xverbatim#1\end{verbatim}[#1|end[verbatim]]
|gdef|@sxverbatim#1\end{verbatim*}[#1|end[verbatim*]]
|endgroup
```

(*End of definition for* `\@xverbatim` *and* `\@sxverbatim`.)

\@verbatim Real start of verbatim environment We use \relax to prevent \item scanning too far.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2017-04-15}{\@verbatim}%
⟨latexrelease⟩                  {Disable hyphenation in verbatim}%
\def\@verbatim{\trivlist \item\relax
  \if@minipage\else\vskip\parskip\fi
  \leftskip\@totalleftmargin\rightskip\z@skip
  \parindent\z@\parfillskip\@flushglue\parskip\z@skip
```

Added \@@par to clear possible \parshape definition from a surrounding list (the verbatim guru says). Switch language when in vertical mode.

```
  \@@par
```

Set \language here to suppress hyphenation. Done this way rather than setting \hyphenchar as that is a global setting.

```
  \language\l@nohyphenation
  \@tempswafalse
  \def\par{%
    \if@tempswa
```

A \leavevmode added: needed if, for example, a blank verbatim line is the first thing in a list item (wow!).

```
      \leavevmode \null \@@par\penalty\interlinepenalty
    \else
      \@tempswatrue
      \ifhmode\@@par\penalty\interlinepenalty\fi
    \fi}%
```

To allow customization we hide the font used in a separate macro.

```
  \let\do\@makeother \dospecials
  \obeylines \verbatim@font \@noligs
```

To avoid a breakpoint after the labels box, we remove the penalty put there by the list macros: another use of \unpenalty!

```
  \everypar \expandafter{\the\everypar \unpenalty}%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\@verbatim}%
⟨latexrelease⟩                  {Disable hyphenation in verbatim}%
⟨latexrelease⟩\def\@verbatim{\trivlist \item\relax
⟨latexrelease⟩  \if@minipage\else\vskip\parskip\fi
⟨latexrelease⟩  \leftskip\@totalleftmargin\rightskip\z@skip
⟨latexrelease⟩  \parindent\z@\parfillskip\@flushglue\parskip\z@skip
⟨latexrelease⟩  \@@par
⟨latexrelease⟩  \@tempswafalse
⟨latexrelease⟩  \def\par{%
⟨latexrelease⟩    \if@tempswa
⟨latexrelease⟩      \leavevmode \null \@@par\penalty\interlinepenalty
⟨latexrelease⟩    \else
⟨latexrelease⟩      \@tempswatrue
⟨latexrelease⟩      \ifhmode\@@par\penalty\interlinepenalty\fi
⟨latexrelease⟩    \fi}%
⟨latexrelease⟩  \let\do\@makeother \dospecials
```

```
⟨latexrelease⟩  \obeylines \verbatim@font \@noligs
⟨latexrelease⟩  \hyphenchar\font\m@ne
⟨latexrelease⟩  \everypar \expandafter{\the\everypar \unpenalty}%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@verbatim`.)

`verbatim` (*env.*) `\endverbatim` (RmS 93/09/19) Protected against 'missing item' error message triggered by empty verbatim environment.

```
\def\verbatim{\@verbatim \frenchspacing\@vobeyspaces \@xverbatim}
\def\endverbatim{\if@newlist \leavevmode\fi\endtrivlist}
```

(*End of definition for* `\verbatim` *and* `\endverbatim`.)

`\verbatim@font` Macro to select the font used for verbatim typesetting. It also does other work if necessary for the font used.

```
\def\verbatim@font{\normalfont\ttfamily}
```

(*End of definition for* `\verbatim@font`.)

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                  {\verbvisiblespace}{Setup visible space for \verb}%
```

`\asciispace` The character in slot 32, in typewriter fonts (historically) a visible space but in other fonts a real space or something else

```
\DeclareRobustCommand\asciispace{\char 32 }
```

(*End of definition for* `\asciispace`.)

`\verbvisiblespace` This defines how to get a visible space in `\verb*` and friends. In classic TEX this is just the slot 32, but in TU encoded fonts we switch fonts and take the character from cmtt.

```
\ifx\Umathcode\@undefined
  \let\verbvisiblespace\asciispace                                % Pdftex version
\else
  \DeclareRobustCommand\verbvisiblespace
           {\leavevmode{\usefont{OT1}{cmtt}{m}{n}\asciispace}}    % xetex/luatex version
\fi
```

(*End of definition for* `\verbvisiblespace`.)

`\@verbvisiblespacebox` The box to hold the visible space character if it isn't in slot 32 in the current typewriter font.

```
\newbox\@verbvisiblespacebox
```

(*End of definition for* `\@verbvisiblespacebox`.)

`verbatim*` (*env.*) For `verbatim*` we also set up the correct visible space character definition and then run `\@vobeyspaces`. As this code is not called as part of the normal verbatim environment (the method is done the other way around this time) we don't have to check if space is already active—it shouldn't be.

```
\@namedef{verbatim*}{\@verbatim
  \@setupverbvisiblespace
  \frenchspacing\@vobeyspaces\@sxverbatim}
\expandafter\let\csname endverbatim*\endcsname =\endverbatim
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\verbvisiblespace}{Setup visible space for \verb}%
⟨latexrelease⟩
⟨latexrelease⟩\@namedef{verbatim*}{\@verbatim\@sxverbatim}
⟨latexrelease⟩
⟨latexrelease⟩\let\asciispace            \@undefined
⟨latexrelease⟩\let\verbvisiblespace      \@undefined
⟨latexrelease⟩\let\@setupverbvisiblespace\@undefined
⟨latexrelease⟩\let\@verbvisiblespacebox  \@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\@setupverbvisiblespace` In pdfTEX a catcode 12 space will produce the character in slot 32 which is assumed to be a visible space character (in a typewriter font in OT1 or T1 encoding). In XeTEX or LuaTEX a font in TU encoding is normally used and that has a real space in this slot. So what we do in this case is this: we check the definition of `\verbvisiblespace` and if it is `\asciispace` we assume that the char32 can be used (e.g., in pdfTEX). We then redefine `\@xobeysp` so that after running `\@vobeyspaces` we get characters from slot 32 for each active space.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                  {\@setupverbvisiblespace}{Setup visible tab for \verb}%
\def\@setupverbvisiblespace{%
  \ifx\verbvisiblespace\asciispace
    \let\@xobeysp\asciispace
  \else
```

Otherwise we measure the width of a character in the mono-spaced current font and place a `\verbvisiblespace` into a box of the right width which we are then using as the character for a space. By default this will be the space character from OT1 cmtt but by changing `\verbvisiblespace` one could use, for example, the `\textvisiblespace` of the current typewriter font.

```
    \setbox\z@\hbox{x}%
    \setbox\@verbvisiblespacebox\hbox to\wd\z@{\hss\verbvisiblespace\hss}%
    \def\@xobeysp{\leavevmode\copy\@verbvisiblespacebox}%
  \fi
  \@setupverbvisibletab
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                  {\@setupverbvisiblespace}{Setup visible space for \verb}%
⟨latexrelease⟩\def\@setupverbvisiblespace{%
⟨latexrelease⟩  \ifx\verbvisiblespace\asciispace
⟨latexrelease⟩    \let\@xobeysp\asciispace
⟨latexrelease⟩  \else
⟨latexrelease⟩    \setbox\z@\hbox{x}%
⟨latexrelease⟩    \setbox\@verbvisiblespacebox\hbox to\wd\z@{\hss\verbvisiblespace\hss}%
⟨latexrelease⟩    \def\@xobeysp{\leavevmode\copy\@verbvisiblespacebox}%
⟨latexrelease⟩  \fi
```

```
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@setupverbvisiblespace}{Setup visible space for \verb}%
⟨latexrelease⟩\let\@setupverbvisiblespace\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@setupverbvisiblespace`.)

`\@setupverbvisibletab` A redirection: just a simple wrapper.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                  {\@setupverbvisibletab}{Setup visible tab for \verb}%
⟨*2ekernel | latexrelease⟩
\def\@setupverbvisibletab{\let\@xobeytab\@xobeysp}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@setupverbvisibletab}{Setup visible tab for \verb}%
⟨latexrelease⟩\let\@setupverbvisibletab\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@setupverbvisibletab`.)

`\@sverb` `\@@sverb` Definitions of `\@sverb` and `\@verb` changed so `\verb+ foo+` does not lose leading blanks when it comes at the beginning of a line. Change made 24 May 89. Suggested by Frank Mittelbach and Rainer Schöpf.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                  {\@sverb}{Support visible tabs}%
```

If the users types `\verb !~! foo` then surprisingly we would get the space as the delimiter and thus "`!~!foo`" in the output. To avoid this scenario we check if `#1` has the character code of a space, if so we recurse otherwise we call `\@@sverb` (which is the original definition of `\@sverb`).

```
\def\@sverb#1{\if\noexpand#1 \expandafter\@sverb\else\@@sverb{#1}\fi}
```

```
\def\@@sverb#1{%
  \catcode`#1\active
  \lccode`\~`#1%
  \gdef\verb@balance@group{\verb@egroup
     \@latex@error{\noexpand\verb illegal in argument}\@ehc}%
  \aftergroup\verb@balance@group
  \lowercase{\let~\verb@egroup}%
```

If `\@sverb` is called from `\@verb` then space is already active and supposed to produce a real space. In this case we do nothing. Otherwise we run `\@setupverbvisiblespace` to setup the right visible space char and afterwards `\@vobeyspaces` to make it the definition for the active space character.

```
  \ifnum0%
    \ifnum\catcode`\ =\active\else 1\fi
    \ifnum\catcode`\^^I=\active\else 1\fi
```

```
    =0 %
  \else  \@setupverbvisiblespace \@vobeyspaces \fi
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                 {\@sverb}{Drop spaces before \verb delimiter}%
⟨latexrelease⟩\def\@@sverb#1{%
⟨latexrelease⟩  \catcode`#1\active
⟨latexrelease⟩  \lccode`\~`#1%
⟨latexrelease⟩  \gdef\verb@balance@group{\verb@egroup
⟨latexrelease⟩     \@latex@error{\noexpand\verb illegal in argument}\@ehc}%
⟨latexrelease⟩  \aftergroup\verb@balance@group
⟨latexrelease⟩  \lowercase{\let~\verb@egroup}%
⟨latexrelease⟩  \ifnum\catcode`\ =\active
⟨latexrelease⟩  \else  \@setupverbvisiblespace \@vobeyspaces \fi
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                 {\@sverb}{Setup visible space for \verb}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@sverb#1{%
⟨latexrelease⟩  \catcode`#1\active
⟨latexrelease⟩  \lccode`\~`#1%
⟨latexrelease⟩  \gdef\verb@balance@group{\verb@egroup
⟨latexrelease⟩     \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
⟨latexrelease⟩  \aftergroup\verb@balance@group
⟨latexrelease⟩  \lowercase{\let~\verb@egroup}%
⟨latexrelease⟩  \ifnum\catcode`\ =\active
⟨latexrelease⟩  \else  \@setupverbvisiblespace \@vobeyspaces \fi
⟨latexrelease⟩}
⟨latexrelease⟩\let\@@sverb\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@sverb}{Setup visible space for \verb}%
⟨latexrelease⟩\def\@sverb#1{%
⟨latexrelease⟩  \catcode`#1\active
⟨latexrelease⟩  \lccode`\~`#1%
⟨latexrelease⟩  \gdef\verb@balance@group{\verb@egroup
⟨latexrelease⟩     \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
⟨latexrelease⟩  \aftergroup\verb@balance@group
⟨latexrelease⟩  \lowercase{\let~\verb@egroup}}%
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@sverb` *and* `\@@sverb`.)

`\@makeother`

```
\def\@makeother#1{\catcode`#112\relax}
```

(*End of definition for* `\@makeother`.)

\verb@balance@group

```
\let\verb@balance@group\@empty
```

(*End of definition for* \verb@balance@group.)

\verb@egroup

```
\def\verb@egroup{\global\let\verb@balance@group\@empty\egroup}
```

(*End of definition for* \verb@egroup.)

\verb@eol@error

```
\begingroup
  \obeylines%
  \gdef\verb@eol@error{\obeylines%
    \def^^M{\verb@egroup\@latex@error{%
            \noexpand\verb ended by end of line}\@ehc}}%
\endgroup
```

(*End of definition for* \verb@eol@error.)

\verb Typesetting a small piece verbatim.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2017-04-15}{\verb}%
⟨latexrelease⟩                 {Disable hyphenation in verb}%
\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
  \bgroup
    \verb@eol@error \let\do\@makeother \dospecials
    \verbatim@font\@noligs
```

Set \language here to suppress hyphenation. Done this way rather than setting \hyphenchar as that is a global setting.

```
    \language\l@nohyphenation
    \@ifstar\@sverb\@verb}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}{\verb}%
⟨latexrelease⟩                 {Disable hyphenation in verb}%
⟨latexrelease⟩\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
⟨latexrelease⟩  \bgroup
⟨latexrelease⟩    \verb@eol@error \let\do\@makeother \dospecials
⟨latexrelease⟩    \verbatim@font\@noligs
⟨latexrelease⟩    \@ifstar\@sverb\@verb}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \verb.)

\@verb

```
\def\@verb{\@vobeyspaces \frenchspacing \@sverb}
```

(*End of definition for* \@verb.)

\verbatim@nolig@list

```
\def\verbatim@nolig@list{\do\`\do\<\do\>\do\,\do\'\do\-}
```

(*End of definition for* `\verbatim@nolig@list`.)

`\do@noligs`

```
\def\do@noligs#1{%
  \catcode`#1\active
  \begingroup
     \lccode`\~`#1\relax
     \lowercase{\endgroup\def~{\leavevmode\kern\z@\char`#1}}}
```

(*End of definition for* `\do@noligs`.)

`\@noligs` To stay compatible with packages that use `\@noligs` we keep it.

```
\def\@noligs{\let\do\do@noligs \verbatim@nolig@list}
```

(*End of definition for* `\@noligs`.)

⟨/2ekernel⟩

File 38
ltmath.dtx

1 Math setup

This file contains a lot of the original plain TEX code, as well as the LATEX environments for math. It still needs sorting out.

```
⟨*2ekernel⟩
\message{math definitions,}
```

1.1 Math commands based on plain TEX

1.1.1 The log-like functions

\log The standard operators:

```
\DeclareRobustCommand\log{\mathop{\operator@font log}\nolimits}
\DeclareRobustCommand\lg{\mathop{\operator@font lg}\nolimits}
\DeclareRobustCommand\ln{\mathop{\operator@font ln}\nolimits}
\DeclareRobustCommand\lim{\mathop{\operator@font lim}}
\DeclareRobustCommand\limsup{\mathop{\operator@font lim\,sup}}
\DeclareRobustCommand\liminf{\mathop{\operator@font lim\,inf}}
\DeclareRobustCommand\sin{\mathop{\operator@font sin}\nolimits}
\DeclareRobustCommand\arcsin{\mathop{\operator@font arcsin}\nolimits}
\DeclareRobustCommand\sinh{\mathop{\operator@font sinh}\nolimits}
\DeclareRobustCommand\cos{\mathop{\operator@font cos}\nolimits}
\DeclareRobustCommand\arccos{\mathop{\operator@font arccos}\nolimits}
\DeclareRobustCommand\cosh{\mathop{\operator@font cosh}\nolimits}
\DeclareRobustCommand\tan{\mathop{\operator@font tan}\nolimits}
\DeclareRobustCommand\arctan{\mathop{\operator@font arctan}\nolimits}
\DeclareRobustCommand\tanh{\mathop{\operator@font tanh}\nolimits}
\DeclareRobustCommand\cot{\mathop{\operator@font cot}\nolimits}
\DeclareRobustCommand\coth{\mathop{\operator@font coth}\nolimits}
\DeclareRobustCommand\sec{\mathop{\operator@font sec}\nolimits}
\DeclareRobustCommand\csc{\mathop{\operator@font csc}\nolimits}
\DeclareRobustCommand\max{\mathop{\operator@font max}}
\DeclareRobustCommand\min{\mathop{\operator@font min}}
\DeclareRobustCommand\sup{\mathop{\operator@font sup}}
\DeclareRobustCommand\inf{\mathop{\operator@font inf}}
\DeclareRobustCommand\arg{\mathop{\operator@font arg}\nolimits}
\DeclareRobustCommand\ker{\mathop{\operator@font ker}\nolimits}
\DeclareRobustCommand\dim{\mathop{\operator@font dim}\nolimits}
\DeclareRobustCommand\hom{\mathop{\operator@font hom}\nolimits}
\DeclareRobustCommand\det{\mathop{\operator@font det}}
\DeclareRobustCommand\exp{\mathop{\operator@font exp}\nolimits}
\DeclareRobustCommand\Pr{\mathop{\operator@font Pr}}
\DeclareRobustCommand\gcd{\mathop{\operator@font gcd}}
\DeclareRobustCommand\deg{\mathop{\operator@font deg}\nolimits}
```

(*End of definition for* `\log`.)

\bmod And some operators have to be done by hand:

```
\DeclareRobustCommand\bmod{%
  \nonscript\mskip-\medmuskip\mkern5mu%
  \mathbin{\operator@font mod}\penalty900\mkern5mu%
  \nonscript\mskip-\medmuskip}
```

(*End of definition for* `\bmod`.)

`\pmod`

```
\DeclareRobustCommand\pmod[1]{%
  \allowbreak\mkern18mu({\operator@font mod}\,\,#1)}
```

(*End of definition for* `\pmod`.)

1.1.2 Biggggg

`\big` Variants on `\big` and friends for use with delimiters:

```
\DeclareRobustCommand\bigl{\mathopen\big}
\DeclareRobustCommand\bigm{\mathrel\big}
\DeclareRobustCommand\bigr{\mathclose\big}
\DeclareRobustCommand\Bigl{\mathopen\Big}
\DeclareRobustCommand\Bigm{\mathrel\Big}
\DeclareRobustCommand\Bigr{\mathclose\Big}
\DeclareRobustCommand\biggl{\mathopen\bigg}
\DeclareRobustCommand\biggm{\mathrel\bigg}
\DeclareRobustCommand\biggr{\mathclose\bigg}
\DeclareRobustCommand\Biggl{\mathopen\Bigg}
\DeclareRobustCommand\Biggm{\mathrel\Bigg}
\DeclareRobustCommand\Biggr{\mathclose\Bigg}
```

(*End of definition for* `\big`.)

1.1.3 The UNSORTED Rest

The other math commands are lifted from plain TEX.

`\jot`

```
\newdimen\jot
\jot=3pt
```

(*End of definition for* `\jot`.)

`\interdisplaylinepenalty`

```
\newcount\interdisplaylinepenalty
\interdisplaylinepenalty=100
```

(*End of definition for* `\interdisplaylinepenalty`.)

`\choose`

```
\def\choose{\atopwithdelims()}
```

(*End of definition for* `\choose`.)

`\brack`

```
\def\brack{\atopwithdelims[]}
```

(*End of definition for* `\brack`.)

\brace

```
\def\brace{\atopwithdelims\{\}}
```

(*End of definition for* \brace.)

\mathpalette

```
\def\mathpalette#1#2{%
  \mathchoice
    {#1\displaystyle{#2}}%
    {#1\textstyle{#2}}%
    {#1\scriptstyle{#2}}%
    {#1\scriptscriptstyle{#2}}}
```

(*End of definition for* \mathpalette.)

\root
\rootbox
\r@@t

```
\newbox\rootbox
\def\root#1\of{%
  \setbox\rootbox\hbox{$\m@th\scriptscriptstyle{#1}$}%
  \mathpalette\r@@t}
\def\r@@t#1#2{%
  \setbox\z@\hbox{$\m@th#1\sqrtsign{#2}$}%
  \dimen@\ht\z@ \advance\dimen@-\dp\z@
  \mkern5mu\raise.6\dimen@\copy\rootbox
  \mkern-10mu\box\z@}
```

(*End of definition for* \root, \rootbox, *and* \r@@t.)

\phantom
\hphantom
\vphantom

```
\newif\ifv@
\newif\ifh@
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\vphantom}{Make commands robust}%
\DeclareRobustCommand\vphantom{\v@true\h@false\ph@nt}
\DeclareRobustCommand\hphantom{\v@false\h@true\ph@nt}
\DeclareRobustCommand\phantom{\v@true\h@true\ph@nt}
```

\mathstrut

```
\DeclareRobustCommand\mathstrut{\vphantom(}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\vphantom}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\vphantom
⟨latexrelease⟩\kernel@make@fragile\hphantom
⟨latexrelease⟩\kernel@make@fragile\phantom
⟨latexrelease⟩\kernel@make@fragile\mathstrut
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\def\ph@nt{%
  \ifmmode
    \expandafter\mathpalette\expandafter\mathph@nt
  \else
    \expandafter\makeph@nt
  \fi}
\def\makeph@nt#1{%
  \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finph@nt}
\def\mathph@nt#1#2{%
  \setbox\z@\hbox{$\m@th#1{#2}$}\finph@nt}
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                  {\finph@nt}{Start LR-mode}%
\def\finph@nt{%
  \setbox\tw@\null
  \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
  \ifh@ \wd\tw@\wd\z@\fi

  \leavevmode@ifvmode\box\tw@}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\finph@nt}{Start LR-mode}%
⟨latexrelease⟩\def\finph@nt{%
⟨latexrelease⟩  \setbox\tw@\null
⟨latexrelease⟩  \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
⟨latexrelease⟩  \ifh@ \wd\tw@\wd\z@\fi \box\tw@}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\phantom` *and others.*)

`\smash`

```
\DeclareRobustCommand\smash{%
  \relax % \relax, in case this comes first in \halign
  \ifmmode
    \expandafter\mathpalette\expandafter\mathsm@sh
  \else
    \expandafter\makesm@sh
  \fi}
\def\makesm@sh#1{%
  \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finsm@sh}
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/11/01}%
⟨latexrelease⟩                  {\mathsm@sh}{Guard against reboxing}%
\def\mathsm@sh#1#2{%
  \setbox\z@\hbox{$\m@th#1{#2}$}%
```

The empty brace groups in front of the smashed box (which is placed by `\finsm@sh`) ensures that a `\smash` in math is not just producing a single box with its dimensions altered, but a box plus this second ord atom. The reason is that TEX sometimes reboxes

a box if its the only thing in a place like the denominator of a fraction. This would then undo the smashing and the additional ord atom prevents that. Two ord atoms in a row do not alter the horizontal spacing in a formula so this is otherwise transparent.

```
  {}\finsm@sh}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\mathsm@sh}{Guard against reboxing}%
⟨latexrelease⟩\def\mathsm@sh#1#2{%
⟨latexrelease⟩   \setbox\z@\hbox{$\m@th#1{#2}$}\finsm@sh}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                  {\finsm@sh}{Start LR-mode}%
\def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \leavevmode@ifvmode\box\z@}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\finsm@sh}{Start LR-mode}%
⟨latexrelease⟩\def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\smash`.)

`\buildrel`

```
\def\buildrel#1\over#2{\mathrel{\mathop{\kern\z@#2}\limits^{#1}}}
```

(*End of definition for* `\buildrel`.)

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\cases}{Make commands robust}%
```

`\cases`

```
\DeclareRobustCommand*\cases[1]{\left\{\,\vcenter{\normalbaselines\m@th
    \ialign{$##\hfil$&\quad{##}\hfil\crcr#1\crcr}}\right.}
```

(*End of definition for* `\cases`.)

`\matrix`

```
\DeclareRobustCommand*\matrix[1]{\null\,\vcenter{\normalbaselines\m@th
    \ialign{\hfil$##$\hfil&&\quad\hfil$##$\hfil\crcr
      \mathstrut\crcr\noalign{\kern-\baselineskip}
      #1\crcr\mathstrut\crcr\noalign{\kern-\baselineskip}}}\,}
```

(*End of definition for* `\matrix`.)

`\pmatrix`

```
\DeclareRobustCommand*\pmatrix[1]{\left(\matrix{#1}\right)}
```

(*End of definition for* \pmatrix.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\cases}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\cases
⟨latexrelease⟩\kernel@make@fragile\matrix
⟨latexrelease⟩\kernel@make@fragile\pmatrix
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\bordermatrix

```
\def\bordermatrix#1{\begingroup \m@th
  \@tempdima 8.75\p@
  \setbox\z@\vbox{%
    \def\cr{\crcr\noalign{\kern2\p@\global\let\cr\endline}}%
    \ialign{$##$\hfil\kern2\p@\kern\@tempdima&\thinspace\hfil$##$\hfil
      &&\quad\hfil$##$\hfil\crcr
      \omit\strut\hfil\crcr\noalign{\kern-\baselineskip}%
      #1\crcr\omit\strut\cr}}%
  \setbox\tw@\vbox{\unvcopy\z@\global\setbox\@ne\lastbox}%
  \setbox\tw@\hbox{\unhbox\@ne\unskip\global\setbox\@ne\lastbox}%
  \setbox\tw@\hbox{$\kern\wd\@ne\kern-\@tempdima\left(\kern-\wd\@ne
    \global\setbox\@ne\vbox{\box\@ne\kern2\p@}%
    \vcenter{\kern-\ht\@ne\unvbox\z@\kern-\baselineskip}\,\right)$}%
  \null\;\vbox{\kern\ht\@ne\box\tw@}\endgroup}
```

(*End of definition for* \bordermatrix.)

\openup

```
\protected\def\openup{\afterassignment\@penup\dimen@}
```

```
\def\@penup{\advance\lineskip\dimen@
  \advance\baselineskip\dimen@
  \advance\lineskiplimit\dimen@}
```

(*End of definition for* \openup.)

\displaylines

```
\newif\ifdt@p
```

```
\def\displ@y{\global\dt@ptrue\openup\jot\m@th
  \everycr{\noalign{\ifdt@p \global\dt@pfalse \ifdim\prevdepth>-1000\p@
      \vskip-\lineskiplimit \vskip\normallineskiplimit \fi
      \else \penalty\interdisplaylinepenalty \fi}}}
```

```
\def\@lign{\tabskip\z@skip\everycr{}} % restore inside \displ@y
```

```
\def\displaylines#1{\displ@y \tabskip\z@skip
  \halign{\hb@xt@\displaywidth{$\@lign\hfil\displaystyle##\hfil$}\crcr
    #1\crcr}}
```

(*End of definition for* \displaylines.)

\sp
\sb

```
\let\sp=^
\let\sb=_
```

(*End of definition for* \sp *and* \sb.)

\tmspace
\
\thinspace
\!
\negthinspace
\:
\medspace
\negmedspace
\;
\thickspace
\negthickspace

Originally LaTeX only provided a small set of spacing commands for use in text and math, some of the commands like `\;` were only supported in math mode. `amsmath` normalized and provided all of them in text and math. This code has now been moved to the kernel so that it is generally available.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\tmspace}{amsmath spacing commands}%
```

`\tmspace` is really meant to be an internal command so it doesn't necessarily has to be robust but it was robust in `amsmath` so we leave it like that.

```
\DeclareRobustCommand\tmspace[3]{%
  \ifmmode\mskip#1#2\else\leavevmode@ifvmode\kern#1#3\fi\relax}
```

In `amsmath` the text kern is `.1667em`. For compatibility reasons we keep the longer one.

```
\DeclareRobustCommand\,{\tmspace+\thinmuskip{.16667em}}
\let\thinspace\,
```

```
\DeclareRobustCommand\!{\tmspace-\thinmuskip{.16667em}}
\let\negthinspace\!
```

```
\DeclareRobustCommand\:{\tmspace+\medmuskip{.2222em}}
\let\medspace\:
```

LaTeX has a second name for this in its manual:

```
\let\>=\:
\DeclareRobustCommand\negmedspace{\tmspace-\medmuskip{.2222em}}
```

```
\DeclareRobustCommand\;{\tmspace+\thickmuskip{.2777em}}
\let\thickspace\;
\DeclareRobustCommand\negthickspace{\tmspace-\thickmuskip{.2777em}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\tmspace}{amsmath spacing commands}%
⟨latexrelease⟩
⟨latexrelease⟩\let\tmspace\@undefined
⟨latexrelease⟩\DeclareRobustCommand{\,}{%
⟨latexrelease⟩   \relax\ifmmode\mskip\thinmuskip\else\thinspace\fi}
⟨latexrelease⟩\DeclareRobustCommand\thinspace{\leavevmode@ifvmode\kern .16667em }
⟨latexrelease⟩\DeclareRobustCommand\negthinspace{\leavevmode@ifvmode\kern-.16667em }
⟨latexrelease⟩\def\>{\mskip\medmuskip}
⟨latexrelease⟩\let\:=\>
⟨latexrelease⟩\def\;{\mskip\thickmuskip}
⟨latexrelease⟩\def\!{\mskip-\thinmuskip}
⟨latexrelease⟩
```

```
⟨latexrelease⟩\let\negmedspace\@undefined
⟨latexrelease⟩\let\negthickspace\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\tmspace` *and others.*)

`\*`

```
\DeclareRobustCommand\*{\discretionary{\thinspace\the\textfont2\char2}{}{}}
```

(*End of definition for* `\*`.)

`\:` Nickname for the medium space since `\>` is not available inside `tabbing`.

```
%\let\:=\>
```

(*End of definition for* `\:`.)

`\active@math@prime` This is the definition of the active math prime.

```
\def\active@math@prime{^\bgroup\prim@s}
```

(*End of definition for* `\active@math@prime`.)

`\prime@s`

```
{\catcode`\'=\active \global\let'\active@math@prime}
\def\prim@s{%
  \prime\futurelet\@let@token\pr@m@s}
\def\pr@m@s{%
  \ifx'\@let@token
    \expandafter\pr@@@s
  \else
    \ifx^\@let@token
      \expandafter\expandafter\expandafter\pr@@@t
    \else
      \egroup
    \fi
  \fi}
\def\pr@@@s#1{\prim@s}
\def\pr@@@t#1#2{#2\egroup}
```

(*End of definition for* `\prime@s`.)

```
{\catcode`\_=\active \gdef_{\_}} % _ in math is
                                 % either subscript or \_
```

1.2 Math Environments

\(\) Produces $...$ with checks that \(isn't used in math mode, and that \) is only used in math mode begun with \(.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\(}{Make \( robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\({%
  \relax\ifmmode\@badmath\else$\fi}%
\DeclareRobustCommand\){%
  \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\(}{Make \( robust}%
⟨latexrelease⟩\def\({%
⟨latexrelease⟩  \relax\ifmmode\@badmath\else$\fi}%
⟨latexrelease⟩\expandafter\let\csname\string( \endcsname\@undefined
⟨latexrelease⟩\def\){%
⟨latexrelease⟩  \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}%
⟨latexrelease⟩\expandafter\let\csname\string) \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \(*and* \)*.*)

\[\] Produces $$...$$ with checks that \[isn't used in math mode, and that \] is only used in display math mode (though there is no real test that this display math started with \[and not with $$).

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\[}{Make \[ robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\[{%
   \relax\ifmmode
      \@badmath
   \else
      \ifvmode
         \nointerlineskip
         \makebox[.6\linewidth]{}%
      \fi
      $$%%$$ BRACE MATCH HACK
   \fi
}%
\DeclareRobustCommand\]{%
   \relax\ifmmode
      \ifinner
         \@badmath
      \else
         $$%%$$ BRACE MATCH HACK
      \fi
   \else
      \@badmath
   \fi
   \ignorespaces
}%
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\[}{Make \[ robust}%
⟨latexrelease⟩\def\[{%
⟨latexrelease⟩   \relax\ifmmode
⟨latexrelease⟩      \@badmath
⟨latexrelease⟩   \else
⟨latexrelease⟩      \ifvmode
⟨latexrelease⟩         \nointerlineskip
⟨latexrelease⟩         \makebox[.6\linewidth]{}%
⟨latexrelease⟩      \fi
⟨latexrelease⟩      $$%%$$ BRACE MATCH HACK
⟨latexrelease⟩   \fi
⟨latexrelease⟩}%
⟨latexrelease⟩\expandafter\let\csname\string[ \endcsname\@undefined
⟨latexrelease⟩\def\]{%
⟨latexrelease⟩   \relax\ifmmode
⟨latexrelease⟩      \ifinner
⟨latexrelease⟩         \@badmath
⟨latexrelease⟩      \else
⟨latexrelease⟩         $$%%$$ BRACE MATCH HACK
⟨latexrelease⟩      \fi
⟨latexrelease⟩   \else
⟨latexrelease⟩      \@badmath
⟨latexrelease⟩   \fi
⟨latexrelease⟩   \ignorespaces
⟨latexrelease⟩}%
⟨latexrelease⟩\expandafter\let\csname\string] \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\[` *and* `\]`.)

`math` (*env.*) `displaymath` (*env.*) Disguises for `\(...\)` and `\[...\]`.

```
\let\math=\(
\let\endmath=\)
\def\displaymath{\[}
\def\enddisplaymath{\]\@ignoretrue}
```

`equation` (*env.*) Numbered equations, using the counter `\c@equation`. *Note*: The document style must define `\theequation` etc., and do the appropriate `\@addtoreset`. It should also redefine `\@eqnnum` if another format for the equation number is desired other than the standard (...), or to move the equation numbers to the flushleft. (See comment on the `\def` of `\@eqnnum`.)

```
\@definecounter{equation}
\def\equation{$$\refstepcounter{equation}}
\def\endequation{\eqno \hbox{\@eqnnum}$$\@ignoretrue}
```

(*End of definition for* `\c@equation`.)

`\@eqnnum` Produces the equation number for equation and eqnarray environments. The following definition is for flushright numbers; for flushleft numbers, see leqno.clo. The equation number is set in black roman type even if an eqnarray environment appears in an italic environment.

```
\def\@eqnnum{{\normalfont \normalcolor (\theequation)}}
```

(*End of definition for* `\@eqnnum`.)

`\stackrel` A disguise for plain TEX's buildrel.

```
\DeclareRobustCommand\stackrel[2]{\mathrel{\mathop{#2}\limits^{#1}}}
```

(*End of definition for* `\stackrel`.)

`\frac` A disguise for plain TEX's `\over`.

```
\DeclareRobustCommand\frac[2]{{\begingroup#1\endgroup\over#2}}
```

(*End of definition for* `\frac`.)

`\sqrt` `\@sqrt` Add an optional argument to plain's `\sqrt` to give the nth root of an expression $\sqrt[n]{e}$.

```
\DeclareRobustCommand\sqrt{\@ifnextchar[\@sqrt\sqrtsign}
\def\@sqrt[#1]{\root #1\of}
```

(*End of definition for* `\sqrt` *and* `\@sqrt`.)

`eqnarray` `\@eqcnt` `\@eqpen` `\if@eqnsw` `\@eqnsel` Here's the eqnarray environment: Default is for left-hand side of equations to be flushright. To make them flushleft, `\let\@eqnsel = \hfil`.

```
\newcount\@eqcnt
\newcount\@eqpen
\newif\if@eqnsw\@eqnswtrue
\newskip\@centering
\@centering = 0pt plus 1000pt
```

To get a proper `\@currentlabel` we have to redefine it for the whole display. Note that we can't use `\refstepcounter` as this results in `\@currentlabel` getting restored at the wrong and thus always writing the first label to the `.aux` file.

```
\def\eqnarray{%
   \stepcounter{equation}%
   \def\@currentlabel{\p@equation\theequation}%
   \def\@currentcounter{equation}%
   \global\@eqnswtrue
   \m@th
   \global\@eqcnt\z@
   \tabskip\@centering
   \let\\\@eqncr
   $$\everycr{}\halign to\displaywidth\bgroup
       \hskip\@centering$\displaystyle\tabskip\z@skip{##}$\@eqnsel
      &\global\@eqcnt\@ne\hskip \tw@\arraycolsep \hfil${##}$\hfil
      &\global\@eqcnt\tw@ \hskip \tw@\arraycolsep
         $\displaystyle{##}$\hfil\tabskip\@centering
      &\global\@eqcnt\thr@@ \hb@xt@\z@\bgroup\hss##\egroup
         \tabskip\z@skip
      \cr
}
```

```
\def\endeqnarray{%
      \@@eqncr
      \egroup
      \global\advance\c@equation\m@ne
   $$\@ignoretrue
}
```

```
\let\@eqnsel=\relax
```

(*End of definition for* `\@eqcnt` *and others.*)

`\nonumber` Switches off equation numbering.

```
\def\nonumber{\global\@eqnswfalse}
```

(*End of definition for* `\nonumber`.)

`\@eqncr` `\@xeqncr` `\@yeqncr`

```
\protected\def\@eqncr{%
   {\ifnum0=`}\fi
   \@ifstar{%
      \global\@eqpen\@M\@yeqncr
   }{%
      \global\@eqpen\interdisplaylinepenalty \@yeqncr
   }%
}
```

```
\def\@yeqncr{\@testopt\@xeqncr\z@skip}
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@xeqncr}{eqnarray support calc syntax}%
\def\@xeqncr[#1]{%
   \ifnum0=`{\fi}%
   \@@eqncr
   \noalign{\penalty\@eqpen\vskip\jot\@vspace@calcify{#1}}%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@xeqncr}{eqnarray support calc syntax}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@xeqncr[#1]{%
⟨latexrelease⟩   \ifnum0=`{\fi}%
⟨latexrelease⟩   \@@eqncr
⟨latexrelease⟩   \noalign{\penalty\@eqpen\vskip\jot\vskip #1\relax}%
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@eqncr`, `\@xeqncr`, *and* `\@yeqncr`.)

`\@@eqncr`

```
\def\@@eqncr{\let\reserved@a\relax
    \ifcase\@eqcnt \def\reserved@a{& & &}\or \def\reserved@a{& &}%
     \or \def\reserved@a{&}\else
       \let\reserved@a\@empty
       \@latex@error{Too many columns in eqnarray environment}\@ehc\fi
     \reserved@a \if@eqnsw\@eqnnum\stepcounter{equation}\fi
     \global\@eqnswtrue\global\@eqcnt\z@\cr}
```

(*End of definition for* `\@@eqncr`.)

eqnarray\@seqncr (*end*) Here's the eqnarray* environment:

```
\let\@seqncr=\@eqncr
\@namedef{eqnarray*}{\protected\def\@eqncr{\nonumber\@seqncr}\eqnarray}
\@namedef{endeqnarray*}{\nonumber\endeqnarray}
```

(*End of definition for* `\@seqncr`.)

`\lefteqn` `\lefteqn{FORMULA}` typesets `FORMULA` in display math style flushleft in a box of width zero.

```
\def\lefteqn#1{\rlap{$\displaystyle #1$}}
```

(*End of definition for* `\lefteqn`.)

`\ensuremath` In math mode, `\ensuremath{text}` is equivalent to text; in LR or paragraph mode, it is equivalent to `$text$`. `\relax` is not needed in front of the `\ifmmode` as `\protect` will be `\let` to `\relax`. This version (due to Donald Arseneau) avoids duplicating its argument in the 'then' and 'else' part of the `\ifmath` which is necessary in nested 'tabular' like environments. See amslatex/2104.

```
\DeclareRobustCommand{\ensuremath}{%
  \ifmmode
    \expandafter\@firstofone
  \else
    \expandafter\@ensuredmath
  \fi}
```

(*End of definition for* `\ensuremath`.)

`\@ensuredmath` The `\relax` stops `\ensuremath{}` starting display math.

```
\long\def\@ensuredmath#1{$\relax#1$}
```

(*End of definition for* `\@ensuredmath`.)

LuaTEX contains new math primitives to place expression over or under horizontally extensible glyphs. Before LuaTEX 1.14 these did not work correctly with the `\mathstyle` primitive and sometimes did not use cramped style in consistent ways. For newer version, we opt into the corrected behavior.

```
\ifx\mathdefaultsmode\@undefined\else
  \mathdefaultsmode=1
\fi
```

`\eqno` `\leqno` Ensure the (deprecated) `$$..\eqno 1 $$` ignores spaces.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2023/06/01}%
⟨latexrelease⟩                  {\eqno}{add ignorespaces to eqno}%
\let\@kernel@eqno\eqno
\let\@kernel@leqno\leqno
\protected\def\eqno{\@kernel@eqno\aftergroup\ignorespaces}
\protected\def\leqno{\@kernel@leqno\aftergroup\ignorespaces}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\eqno}{add ignorespaces to eqno}%
⟨latexrelease⟩\let\eqno\@kernel@eqno
⟨latexrelease⟩\let\leqno\@kernel@leqno
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\eqno` *and* `\leqno`.)

1.3 External options to the standard document classes

1.3.1 Left equation numbering

`\@eqnnum` To put the equation number on the left side of an equation we have to use a little trick. The number is shifted `\displaywidth` to the left inside a box of (approximately) zero width. This fails when the equation is too wide, the equation number than may overprint the equation itself.

```
⟨*leqno⟩
\renewcommand\@eqnnum{\hb@xt@.01\p@{}%
                      \rlap{\normalfont\normalcolor
                        \hskip -\displaywidth(\theequation)}}
⟨/leqno⟩
```

(*End of definition for* `\@eqnnum`.)

1.3.2 Flush left equations

To get the displayed math environments to print the contents flush left (with an indentation) we have to redefine all of LaTeX 2_ε's displayed math environments.

`\mathindent` The amount of indentation of the equations is stored in a register.

```
⟨*fleqn⟩
\newskip\mathindent
```

The setting of `\mathindent` has to be deferred until the class file has been processed, because `\leftmargini` is still 0pt wide at the moment `fleqn.clo` is read in.

```
\AtEndOfClass{\mathindent\leftmargini}
```

(*End of definition for* `\mathindent`.)

`\[` Begin display math;

```
\IncludeInRelease{2015/01/01}{\[}{Make \[ robust}%
\DeclareRobustCommand\[{\relax
                \ifmmode\@badmath
                \else
                  \begin{trivlist}%
                    \@beginparpenalty\predisplaypenalty
                    \@endparpenalty\postdisplaypenalty
                    \item[]\leavevmode
                    \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
                      \hskip\mathindent\bgroup
                \fi}
\EndIncludeInRelease
```

```
\IncludeInRelease{0000/00/00}{\[}{Make \[ robust}%
\renewcommand\[{\relax
                \ifmmode\@badmath
                \else
                  \begin{trivlist}%
                    \@beginparpenalty\predisplaypenalty
                    \@endparpenalty\postdisplaypenalty
                    \item[]\leavevmode
```

```
                    \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
                      \hskip\mathindent\bgroup
                \fi}
\EndIncludeInRelease
```

(*End of definition for* `\[`.)

`\]` end display math;

```
\IncludeInRelease{2015/01/01}{\]}{Make \] robust}%
\DeclareRobustCommand\]{\relax
                \ifmmode
                      \egroup $\hfil% $
                    \egroup
                  \end{trivlist}%
                \else \@badmath
                \fi}
\EndIncludeInRelease
```

```
\IncludeInRelease{0000/00/00}{\]}{Make \] robust}%
\renewcommand\]{\relax
                \ifmmode
                      \egroup $\hfil% $
                    \egroup
                  \end{trivlist}%
                \else \@badmath
                \fi}
\EndIncludeInRelease
```

(*End of definition for* `\]`.)

`equation` (*env.*) The `equation` environment

```
\renewenvironment{equation}%
    {\@beginparpenalty\predisplaypenalty
     \@endparpenalty\postdisplaypenalty
     \refstepcounter{equation}%
     \trivlist \item[]\leavevmode
       \hb@xt@\linewidth\bgroup $\m@th% $
         \displaystyle
         \hskip\mathindent}%
```

Ensure that there is at least a space between formula and equation number so that they don't bump in each other.

```
        {$\hskip .3em minus.3em\hfil % $
         \displaywidth\linewidth\hbox{\@eqnnum}%
       \egroup
     \endtrivlist}
```

`eqnarray` (*env.*) The `eqnarray` environment

```
\renewenvironment{eqnarray}{%
    \stepcounter{equation}%
    \def\@currentlabel{\p@equation\theequation}%
    \def\@currentcounter{equation}%
    \global\@eqnswtrue\m@th
    \global\@eqcnt\z@
    \tabskip\mathindent
```

```
    \let\\=\@eqncr
    \setlength\abovedisplayskip{\topsep}%
    \ifvmode
      \addtolength\abovedisplayskip{\partopsep}%
    \fi
```

When the documentclass uses a non-zero `\parskip` setting the `\topsep` might have a negative value to compensate for that. Therefore we add `\parskip` to `\abovedisplayskip`.

```
    \addtolength\abovedisplayskip{\parskip}%
    \setlength\belowdisplayskip{\abovedisplayskip}%
    \setlength\belowdisplayshortskip{\abovedisplayskip}%
    \setlength\abovedisplayshortskip{\abovedisplayskip}%
    $$\everycr{}\halign to\linewidth% $$
    \bgroup
      \hskip\@centering
      $\displaystyle\tabskip\z@skip{##}$\@eqnsel&%
      \global\@eqcnt\@ne \hskip \tw@\arraycolsep \hfil${##}$\hfil&%
      \global\@eqcnt\tw@ \hskip \tw@\arraycolsep
        $\displaystyle{##}$\hfil \tabskip\@centering&%
      \global\@eqcnt\thr@@
        \hb@xt@\z@\bgroup\hss##\egroup\tabskip\z@skip\cr}%
      {\@@eqncr
    \egroup
    \global\advance\c@equation\m@ne$$% $$
    \@ignoretrue
    }
⟨/fleqn⟩
```

File 39
ltlists.dtx

1 List, and related environments

The generic commands for creating an indented environment – `enumerate`, `itemize`, `quote`, etc – are:

`\list{`⟨*LABEL*⟩`}{`⟨*COMMANDS*⟩`}` ... `\endlist`

which can be invoked by the user as the list environment. The LABEL argument specifies item labeling. COMMANDS contains commands for changing the horizontal and vertical spacing parameters.

Each item of the environment is begun by the command `\item[`ITEMLABEL`]` which produces an item labeled by ITEMLABEL. If the argument is missing, then the LABEL argument of the `\list` command is used as the item label.

The label is formed by putting `\makelabel{`⟨*ITEMLABEL*⟩`}` in an hbox whose width is either its natural width or else `\labelwidth`, whichever is larger. The `\list` command defines `\makelabel` to have the default definition:

`\makelabel{`⟨*ARG*⟩`}` == BEGIN `\hfil` ARG END

which, for a label of width less than `\labelwidth`, puts the label flushright, `\labelsep` to the left of the item's text. However, `\makelabel` can be `\let` to another command by the `\list`'s COMMANDS argument.

A `\usecounter{`⟨*foo*⟩`}` command in the second argument causes the counter *foo* to be initialized to zero, and stepped by every `\item` command without an argument. (`\label` commands within the list refer to this counter.)

When you leave a list environment, returning either to an enclosing list or normal text mode, LaTeX begins a new paragraph if and only if you leave a blank line after the `\end` command. This is accomplished by the `\@endparenv` command.

Blank lines are ignored every other reasonable place–i.e.:

- Between the `\begin{list}` and the first `\item`,
- Between the `\item` and the text of that item.
- Between the end of the last item and the `\end{list}`.

For an environment like quotation, in which items are not labeled, the entire environment is a single item. It is defined by letting `\quotation` == `\list{}{...}\item\relax`. (Note the `\relax`, there in case the first character in the environment is a '['.) The spacing parameters provide a great deal of flexibility in designing the format, including the ability to let the indentation of the first paragraph be different from that of the subsequent ones.

The trivlist environment is equivalent to a list environment whose second argument sets the following parameter values:

`\leftmargin` **= 0:** causes no indentation of left margin

`\labelwidth` **= 0:** see below for precise effect this has.

\itemindent = 0: with a null label, makes first paragraph have no indentation. Succeeding paragraphs have \parindent indentation. To give first paragraph same indentation, set \itemindent = \parindent before the \item[].

Every \item in a trivlist environment must have an argument—in many cases, this will be the null argument (\item[]). The trivlist environment is mainly used for paragraphing environments, like verbatim, in which there is no margin change. It provides the same vertical spacing as the list environment, and works reasonably well when it occurs immediately after an \item command in an enclosing list.

1.1 List and Trivlist

The following variables are used inside a list environment:

\@totalleftmargin The distance that the prevailing left margin is indented from the outermost left margin,

\linewidth The width of the current line. Must be initialized to \hsize.

\@listdepth A count for holding current list nesting depth.

\makelabel A macro with a single argument, used to generate the label from the argument (given or implied) of the \item command. Initialized to \@mklab by the \list command. This command must produce some stretch—i.e., an \hfil.

\@inlabel A switch that is false except between the time an \item is encountered and the time that TEX actually enters horizontal mode. Should be tested by commands that can be messed up by the list environment's use of \everypar.

\box\@labels When @inlabel = true, it holds the labels to be put out by \everypar.

@noparitem A switch set by \list when @inlabel = true. Handles the case of a \list being the first thing in an item.

@noparlist A switch set true for a list that begins an item. No \topsep space is added before or after \item's such a list.

@newlist Set true by \list, set false by the first text (by \everypar).

@noitemarg Set true when executing an \item with no explicit argument. Used to save space. To save time, make two separate \@item commands.

@nmbrlist Set true by \usecounter command, causes list to be numbered.

\@listctr \def'ed by \usecounter to name of counter.

\@noskipsec A switch set true by a sectioning command when it is creating an in-text heading with \everypar.

Throughout a list environment, \hsize is the width of the current line, measured from the outermost left margin to the outermost right margin. Environments like tabbing should use \linewidth instead of \hsize.

Here are the parameters of a list that can be set by commands in the \list's COMMANDS argument. These parameters are all TEX skips or dimensions (defined by \newskip or \newdimen), so the usual TEX or LATEX commands can be used to set them. The commands will be executed in vmode if and only if the \list was preceded by a \par (or something like an \end{list}), so the spacing parameters can be set according to whether the list is inside a paragraph or is its own paragraph.

1.2 Vertical Spacing (skips)

`\topsep`: Space between first item and preceding paragraph.

`\partopsep`: Extra space added to `\topsep` when environment starts a new paragraph (is called in vmode).

`\itemsep`: Space between successive items.

`\parsep`: Space between paragraphs within an item – the `\parskip` for this environment.

1.3 Penalties

`\@beginparpenalty`: put at the beginning of a list

`\@endparpenalty`: put at end of list

`\@itempenalty`: put between items.

1.4 Horizontal Spacing (dimens)

`\leftmargin`: space between left margin of enclosing environment (or of page if top level list) and left margin of this list. Must be nonnegative.

`\rightmargin`: analogous.

`\listparindent`: extra indentation at beginning of every paragraph of a list except the one started by the `\item` command. May be negative! Usually, labeled lists have `\listparindent` equal to zero.

`\itemindent`: extra indentation added right BEFORE an item label.

`\labelwidth`: nominal width of box that contains the label. If the natural width of the label $\leq$ `\labelwidth`, then the label is flushed right inside a box of width `\labelwidth` (with an `\hfil`). Otherwise, a box of the natural width is employed, which causes an indentation of the text on that line.

`\labelsep`: space between end of label box and text of first item.

1.5 Default Values

Defaults for the list environment are set as follows. First, `\rightmargin`, `\listparindent` and `\itemindent` are set to 0pt. Then, one of the commands `\@listi`, `\@listii`, ... , `\@listvi` is called, depending upon the current level of the list. The `\@list` ... commands should be defined by the document style. A convention that the document style should follow is to set `\leftmargin` to `\leftmargini`,..., `\leftmarginvi` for the appropriate level. Items that aren't changed may be left alone, but everything that could possibly be changed must be reset. *Historical LaTeX 2.09 comments (not necessarily accurate any more):*

```
\list{LABEL}{COMMANDS} ==
 BEGIN
   if \@listdepth > 5
     then LaTeX error: 'Too deeply nested'
     else \@listdepth :=G \@listdepth + 1
```

```
    fi
    \rightmargin      := 0pt
    \listparindent    := 0pt
    \itemindent       := 0pt
    \eval(@list \romannumeral\the\@listdepth)  %% Set default values:
    \@itemlabel       :=L LABEL
    \makelabel        == \@mklab
    @nmbrlist         :=L false
    COMMANDS

    \@trivlist                % commands common to \list and \trivlist

    \parskip          :=L \parsep
    \parindent        :=L \listparindent
    \linewidth        :=L \linewidth - \rightmargin -\leftmargin
    \@totalleftmargin :=L \@totalleftmargin + \leftmargin
    \parshape 1 \@totalleftmargin \linewidth
    \ignorespaces                     % gobble space up to \item
   END

\endlist == BEGIN \@listdepth :=G \@listdepth -1
                  \endtrivlist
            END

\@trivlist ==
 BEGIN
    if @newlist = T then \@noitemerr fi
                    %% This command removed for some forgotten reason.
    \@topsepadd :=L \topsep
    if @noskipsec then leave vertical mode fi  %% Added 11 Jun 85
    if vertical mode
      then \@topsepadd :=L \@topsepadd + \partopsep
      else \unskip \par             % remove glue from end of last line
    fi
    if @inlabel = true
       then @noparitem :=L true
            @noparlist :=L true
       else @noparlist :=L false
            \@topsep   :=L \@topsepadd
    fi
    \@topsep          :=L \@topsep + \parskip  %% Change 4 Sep 85
    \leftskip         :=L 0pt          % Restore paragraphing parameters
    \rightskip        :=L \@rightskip
    \parfillskip      :=L 0pt + 1fil

  NOTE: \@setpar called on every \list in case \par has been
  temporarily  munged before the \list command.
    \@setpar{if @newlist = false then {\@@par} fi}
    \@newlist         :=G T
    \@outerparskip    :=L \parskip
```

```
END

\trivlist  ==
BEGIN
 \parsep     := \parskip
 @nmbrlist := F
 \@trivlist
 \labelwidth := 0
 \leftmargin := 0
 \itemindent := \parindent
 \@itemlabel :=L "empty"          %% added 93/12/13
 \makelabel{LABEL} == LABEL
END

\endtrivlist ==
  BEGIN
    if @inlabel = T then \indent fi
    if horizontal mode then \unskip \par fi
    if @noparlist = true
      else if \lastskip > 0
               then \@tempskipa := \lastskip
                    \vskip - \lastskip
                    \vskip \@tempskipa -\@outerparskip + \parskip
           fi
           \@endparenv
    fi
  END

\@endparenv ==
  BEGIN
   \addpenalty{@endparpenalty}
   \addvspace{\@topsepadd}
   \endgroup    %% ends the \begin command's \begingroup
   \par  ==  BEGIN
               \@restorepar
               \everypar{}
               \par
             END
   \everypar == BEGIN remove \lastbox \everypar{} END
   \begingroup  %% to match the \end commands \endgroup
  END

\item == BEGIN if math mode then WARNING fi
                 if  next char = [
                 then  \@item
                 else  @noitemarg := true
                       \@item[@itemlabel]
         END

\@item[LAB] ==
```

```
BEGIN
 if @noparitem = true
   then @noparitem := false
            % NOTE: then clause  hardly every taken,
            %  so made a macro \@donoparitem
        \box\@labels :=G \hbox{\hskip -\leftmargin
                                \box\@labels
                                \hskip \leftmargin }
        if @minipage = false then
           \@tempskipa := \lastskip
           \vskip -\lastskip
           \vskip \@tempskipa + \@outerparskip - \parskip
        fi
   else if @inlabel = true
          then \indent \par   % previous item empty.
        fi
        if hmode then 2 \unskip's
                        % To remove any space at end of prev.
                        % paragraph that could cause a blank line.
                 \par
        fi
        if @newlist = T
           then if @nobreak = T   % Kludge if list follows \section
                   then \addvspace{\@outerparskip - \parskip}
                   else \addpenalty{\@beginparpenalty}
                        \addvspace{\@topsep}
                        \addvspace{-\parskip}   %% added 4 Sep 85
                 fi
           else \addpenalty{\@itempenalty}
                \addvspace{\itemsep}
        fi
        @inlabel :=G true
 fi

 \everypar{ @minipage :=G F
            @newlist :=G F
            if  @inlabel = true
              then @inlabel :=G false
                   \hskip -\parindent
                   \box\@labels
                   \penalty 0
                         %% 3 Oct 85  – allow line break here
                   \box\@labels :=G null
            fi
            \everypar{} }
@nobreak :=G false
if  @noitemarg = true
   then @noitemarg := false
        if @nmbrlist
          then \refstepcounter{\@listctr}
```

```
  fi      fi
  \@tempboxa   :=L \hbox{\makelabel{LAB}}
  \box\@labels :=G \@labels \hskip \itemindent
                   \hskip - (\labelwidth + \labelsep)
                   if \wd \@tempboxa > \labelwidth
                      then \box\@tempboxa
                      else \hbox to \labelwidth {\unhbox\@tempboxa}
                   fi
                   \hskip\labelsep
  \ignorespaces                        %gobble space up to text
END

\makelabel{LABEL} == ERROR    %% default to catch lonely \item


\usecounter{CTR} == BEGIN   @nmbrlist :=L true
                          \@listctr == CTR
                          \setcounter{CTR}{0}
                   END
```

DEFINE `\dimen`'s and `\count`

End of historical LaTeX 2.09 comments.

`\topsep`
`\partopsep`
`\itemsep`
`\parsep`
`\@topsep`
`\@topsepadd`
`\@outerparskip`

```
⟨*2ekernel⟩
\newskip\topsep
\newskip\partopsep
\newskip\itemsep
\newskip\parsep
\newskip\@topsep
\newskip\@topsepadd
\newskip\@outerparskip
```

(*End of definition for* `\topsep` *and others.*)

`\leftmargin`
`\rightmargin`
`\listparindent`
`\itemindent`
`\labelwidth`
`\labelsep`
`\linewidth`
`\@totalleftmargin`

```
\newdimen\leftmargin
\newdimen\rightmargin
\newdimen\listparindent
\newdimen\itemindent
\newdimen\labelwidth
\newdimen\labelsep
\newdimen\linewidth
\newdimen\@totalleftmargin \@totalleftmargin=\z@
```

(*End of definition for* `\leftmargin` *and others.*)

\leftmargini
\leftmarginii
\leftmarginiii
\leftmarginiv
\leftmarginv
\leftmarginvi

```
\newdimen\leftmargini
\newdimen\leftmarginii
\newdimen\leftmarginiii
\newdimen\leftmarginiv
\newdimen\leftmarginv
\newdimen\leftmarginvi
```

(*End of definition for* `\leftmargini` *and others.*)

\@listdepth
\@itempenalty
\@beginparpenalty
\@endparpenalty

```
\newcount\@listdepth \@listdepth=0
\newcount\@itempenalty
\newcount\@beginparpenalty
\newcount\@endparpenalty
```

(*End of definition for* `\@listdepth` *and others.*)

\@labels

```
\newbox\@labels
```

(*End of definition for* `\@labels`.)

\if@inlabel
\@inlabelfalse
\@inlabeltrue

```
\newif\if@inlabel \@inlabelfalse
```

(*End of definition for* `\if@inlabel`, `\@inlabelfalse`, *and* `\@inlabeltrue`.)

\if@newlist
\@newlistfalse
\@newlisttrue

```
\newif\if@newlist   \@newlistfalse
```

(*End of definition for* `\if@newlist`, `\@newlistfalse`, *and* `\@newlisttrue`.)

\if@noparitem
\@noparitemfalse
\@noparitemtrue

```
\newif\if@noparitem \@noparitemfalse
```

(*End of definition for* `\if@noparitem`, `\@noparitemfalse`, *and* `\@noparitemtrue`.)

\if@noparlist
\@noparlistfalse
\@noparlisttrue

```
\newif\if@noparlist \@noparlistfalse
```

(*End of definition for* `\if@noparlist`, `\@noparlistfalse`, *and* `\@noparlisttrue`.)

\if@noitemarg
\@noitemargfalse
\@noitemargtrue

```
\newif\if@noitemarg \@noitemargfalse
```

(*End of definition for* `\if@noitemarg`, `\@noitemargfalse`, *and* `\@noitemargtrue`.)

\if@newlist
\@newlistfalse
\@newlisttrue

```
\newif\if@nmbrlist  \@nmbrlistfalse
```

(*End of definition for* `\if@newlist`, `\@newlistfalse`, *and* `\@newlisttrue`.)

list\list

```
\def\list#1#2{%
  \ifnum \@listdepth >5\relax
    \@toodeep
  \else
    \global\advance\@listdepth\@ne
  \fi
  \rightmargin\z@
  \listparindent\z@
  \itemindent\z@
  \csname @list\romannumeral\the\@listdepth\endcsname
  \def\@itemlabel{#1}%
  \let\makelabel\@mklab
  \@nmbrlistfalse
  #2\relax
  \@trivlist
  \parskip\parsep
  \parindent\listparindent
  \advance\linewidth -\rightmargin
  \advance\linewidth -\leftmargin
  \advance\@totalleftmargin \leftmargin
  \parshape \@ne \@totalleftmargin \linewidth
  \ignorespaces}
```

(*End of definition for* `\list`.)

`\par@deathcycles`

```
\newcount\par@deathcycles
```

(*End of definition for* `\par@deathcycles`.)

`\@trivlist` Because `\par` is sometimes made a no-op it is possible for a missing `\item` to produce a loop that does not fill memory and so never gets trapped by TEX. We thus need to trap this here by setting `\par` to count the number of times a paragraph ii is called with no progress being made started.

```
\def\@trivlist{%
  \if@noskipsec \leavevmode \fi
  \@topsepadd \topsep
  \ifvmode
    \advance\@topsepadd \partopsep
  \else
    \unskip \par
  \fi
  \if@inlabel
    \@noparitemtrue
    \@noparlisttrue
  \else
    \if@newlist \@noitemerr \fi
    \@noparlistfalse
    \@topsep \@topsepadd
  \fi
  \advance\@topsep \parskip
  \leftskip \z@skip
  \rightskip \@rightskip
  \parfillskip \@flushglue
```

```
  \par@deathcycles \z@
  \@setpar{\if@newlist
             \advance\par@deathcycles \@ne
             \ifnum \par@deathcycles >\@m
               \@noitemerr
               {\@@par}%
             \fi
           \else
             {\@@par}%
           \fi}%
  \global \@newlisttrue
  \@outerparskip \parskip}
```

(*End of definition for* `\@trivlist`.)

`\trivlist`

(*End of definition for* `\trivlist`.)

```
\def\trivlist{%
  \parsep\parskip
  \@nmbrlistfalse
  \@trivlist
  \labelwidth\z@
  \leftmargin\z@
  \itemindent\z@
```

We initialise `\@itemlabel` so that a `trivlist` with an `\item` not having an optional argument doesn't produce an error message.

```
  \let\@itemlabel\@empty
  \def\makelabel##1{##1}}
```

(*End of definition for* `\trivlist`.)

`\endlist`

```
\def\endlist{%
  \global\advance\@listdepth\m@ne
  \endtrivlist}
```

(*End of definition for* `\endlist`.)

The definition of `\trivlist` used to be in ltspace.dtx so that other commands could be 'let to it'. They now use `\def`.

`\endtrivlist`

```
\def\endtrivlist{%
  \if@inlabel
    \leavevmode
    \global \@inlabelfalse
  \fi
  \if@newlist
    \@noitemerr
    \global \@newlistfalse
  \fi
  \ifhmode\unskip \par
```

We also check if we are in math mode and issue an error message if so (hoping that `\@currenvir` resolves suitably). Otherwise the usual "perhaps a missing item" error will get triggered later which is confusing.

```
  \else
```

```
    \@inmatherr{\end{\@currenvir}}%
  \fi
  \if@noparlist \else
    \ifdim\lastskip >\z@
      \@tempskipa\lastskip \vskip -\lastskip
      \advance\@tempskipa\parskip \advance\@tempskipa -\@outerparskip
      \vskip\@tempskipa
    \fi
    \@endparenv
  \fi
}
```

(*End of definition for* `\endtrivlist`.)

`\@endparenv`
`\@doendpe`

To suppress the paragraph indentation in text immediately following a paragraph-making environment, `\everypar` is changed to remove the space, and `\par` is redefined to restore `\everypar`. Instead of redefining `\par` and `\everypar`, `\@endparenv` was changed to set the @endpe switch, letting `\end` redefine `\par` and `\everypar`.

This allows paragraph-making environments to work right when called by other environments. (Changed 27 Oct 86)

In 2024 this logic was partially replaced with a new algorithm:

- `\if@endpe` is now set globally to `true` or `false`.
- In addition `\@endpetrue` initiates an `\aftergroup` call to `\propagate@doendpe` if it is used inside a group.
- `\propagate@doendpe` in turn checks the status of `\if@endpe` and if that is `true` it calls `\@doendpe` otherwise it does nothing.
- `\@doendpe` in turn calls `\@endpetrue` and also makes the necessary changes to `\par` and `\everypar` so that they handle as before any empty line that follows the environment.
- Because of the `\@endpetrue` we get another `\aftergroup`, so the mechanisnm slowly migrates out of several groups if those follow immediately after the end of the environment. If, however, there is a new paragraph started or an explicit `\par` before the next group ends then this will result in a call to `\@endpefalse` and the migration stops (note that `\propagate@doendpe` is still called once after the group but does nothing).
- Using this approach something like

  ```
  {%  some customization here
   \begin{equation}
     x=y
   \end{equation}}
  some text
  ```

  is still correctly identified as a paragraph continuation so that there is no indentation before `some text`.

- We can get away with using global settings of `\if@endpe` even in nested situations (without keeping track of the status in a stack), because the switch change is made only at the very end of such environments, basically directly before the `\endgroup` in `\end`, and it is later set back to `false` by the next `\everypar` or the next `\par`. Even if the environment is called without using `\begin` `\end`, the situation doesn't change (or rather cause a problem).

- However, there is one scenario where the new approach would change the behavior. If a box is being built, e.g., with `\setbox`, we have now the case that a `\@endpetrue` inside would migrate out into a context in which it should not be true (because the box might get used elsewhere, e.g., a float). In the past, due to local switch changes, that didn't happen, i.e., `\if@endpe` would revert to `false` at the end of the box definition.

- Thus, to avoid that one has to explicitly set it back to false at the end of such constructions, just as we also need to prevent colors from migrating out. Thus the correct place to do this is in `\color@endgroup` because that is always called at that point.

```
\def\@endparenv{%
  \addpenalty\@endparpenalty\addvspace\@topsepadd\@endpetrue}
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@doendpe}{clubpenalty fix}%
\def\@doendpe{\@endpetrue
    \def\par{\@restorepar
```

If a section heading changes `\clubpenalty` to keep lines after it together then this modification is restored via the `\everypar` mechanism at the start of the next paragraph. As we destroy the contents of this token here we explicitly set `\clubpenalty` back to its default.

```
            \clubpenalty\@clubpenalty
            \@endpefalse
            \everypar{}\par}%
```

Use `\setbox0=\lastbox` instead of `\hskip -\parindent` so that a `\noindent` becomes a no-op when used before a line immediately following a list environment(23 Oct 86).

```
    \everypar
        {{\setbox\z@\lastbox}%
         \everypar{}\@endpefalse}}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@doendpe}{clubpenalty fix}%
⟨latexrelease⟩\def\@doendpe{\@endpetrue
⟨latexrelease⟩    \def\par{\@restorepar\everypar{}\par\@endpefalse}\everypar
⟨latexrelease⟩              {{\setbox\z@\lastbox}\everypar{}\@endpefalse}}
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\@endparenv` *and* `\@doendpe`.)

`\if@endpe`
`\@endpefalse`
`\@endpetrue`
`\propagate@doendpe`

As outlined above these are no longer simple switches, but we keep the name because it is used all over the place.

```
\newif\if@endpe
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/11/01}%
⟨latexrelease⟩                  {\@endpetrue}{New @endpe handling}%
\def\@endpefalse{\global\let\if@endpe\iffalse}
\def\@endpetrue {%
  \global\let\if@endpe\iftrue
```

If we are inside a group then propagate to the outside:

```
   \ifnum\currentgrouplevel>\z@
      \aftergroup\propagate@doendpe
   \fi
}
```

If `\if@endpe` is still true after the group ends, we run `\@doendpe` that in turn runs another `\@endpetrue` (besides other things), thus propagating further if necessary. However, if the endpe situation got resolved and `\if@endpe` is `false` then nothing further happens.

```
\def\propagate@doendpe{\if@endpe \@doendpe \fi}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@endpetrue}{New @endpe handling}%
⟨latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩\def\@endpefalse{\let\if@endpe\iffalse}
⟨latexrelease⟩\def\@endpetrue{\let\if@endpe\iftrue}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\if@endpe` *and others.*)

`\@mklab`

```
\def\@mklab#1{\hfil #1}
```

(*End of definition for* `\@mklab`*.*)

`\item`

```
\def\item{%
  \@inmatherr\item
  \@ifnextchar [\@item{\@noitemargtrue \@item[\@itemlabel]}}
```

(*End of definition for* `\item`*.*)

`\@donoparitem`

```
\def\@donoparitem{%
  \@noparitemfalse
  \global\setbox\@labels\hbox{\hskip -\leftmargin
                              \unhbox\@labels
                               \hskip \leftmargin}%
  \if@minipage\else
    \@tempskipa\lastskip
    \vskip -\lastskip
    \advance\@tempskipa\@outerparskip
```

```
    \advance\@tempskipa -\parskip
    \vskip\@tempskipa
  \fi}
```

(*End of definition for* `\@donoparitem`.)

`\@item`

```
\def\@item[#1]{%
  \if@noparitem
    \@donoparitem
  \else
    \if@inlabel
      \indent \par
    \fi
    \ifhmode
      \unskip\unskip \par
    \fi
    \if@newlist
      \if@nobreak
        \@nbitem
      \else
        \addpenalty\@beginparpenalty
        \addvspace\@topsep
        \addvspace{-\parskip}%
      \fi
    \else
      \addpenalty\@itempenalty
      \addvspace\itemsep
    \fi
    \global\@inlabeltrue
  \fi
  \everypar{%
    \@minipagefalse
    \global\@newlistfalse
```

This `\if@inlabel` check is needed in case an item starts of inside a group so that `\everypar` does not become empty outside that group.

```
    \if@inlabel
      \global\@inlabelfalse
```

The paragraph indent is now removed by using `\setbox...` since this makes `\noindent` a no-op here, as it should be. Thus the following comment is redundant but is left here for the sake of future historians: this next command was changed from an hskip to a kern to avoid a break point after the parindent box: the skip could cause a line-break if a very long label occurs in raggedright setting. If `\noindent` was used after `\item` want to cancel the `\itemindent` skip. This case can be detected as the indentation box will be void.

```
      {\setbox\z@\lastbox
       \ifvoid\z@
         \kern-\itemindent
       \fi}%
      \box\@labels
      \penalty\z@
    \fi
```

This code is intended to prevent a page break after the first line of an item that comes immediately after a section title. It may be sensible to always forbid a page break after one line of an item? As with all such settings of `\clubpenalty` it is local so will have no effect if the item starts in a group.

Only resetting `\@nobreak` when it is true is now essential since now it is sometimes set locally.

```
    \if@nobreak
      \@nobreakfalse
      \clubpenalty \@M
    \else
      \clubpenalty \@clubpenalty
      \everypar{}%
    \fi}%

  \if@noitemarg
    \@noitemargfalse
    \if@nmbrlist

      \refstepcounter\@listctr
    \fi
  \fi
```

We use `\sbox` to support colour commands.

```
  \sbox\@tempboxa{\makelabel{#1}}%
  \global\setbox\@labels\hbox{%
    \unhbox\@labels
    \hskip \itemindent
    \hskip -\labelwidth
    \hskip -\labelsep
    \ifdim \wd\@tempboxa >\labelwidth
      \box\@tempboxa

    \else
      \hbox to\labelwidth {\unhbox\@tempboxa}%
    \fi
    \hskip \labelsep}%
  \ignorespaces}
```

(*End of definition for* `\@item`.)

`\makelabel`

```
\def\makelabel#1{%
  \@latex@error{Lonely \string\item--perhaps a missing
        list environment}\@ehc}
```

(*End of definition for* `\makelabel`.)

`\@nbitem`

```
\def\@nbitem{%
  \@tempskipa\@outerparskip
  \advance\@tempskipa -\parskip
  \addvspace\@tempskipa}
```

(*End of definition for* `\@nbitem`.)

\usecounter

```
\def\usecounter#1{\@nmbrlisttrue\def\@listctr{#1}\setcounter{#1}\z@}
```

(*End of definition for* \usecounter.)

1.6 Itemize and Enumerate

Enumeration is done with four counters: enumi, enumii, enumiii and enumiv, where enumN controls the numbering of the Nth level enumeration. The label is generated by the commands \labelenumi ... \labelenumiv, which should be defined by the document style. Note that \p@enumN\theenumN defines the output of a \ref command. A typical definition might be:

```
\def\theenumii{\alph{enumii}}
\def\p@enumii{\theenumi}
\def\labelenumii{(\theenumii)}
```

which will print the labels as '(a)', '(b)', ... and print a \ref as '3a'.

The item numbers are moved to the right of the label box, so they are always a distance of \labelsep from the item.

\@enumdepth holds the current enumeration nesting depth.

Itemization is controlled by four commands: \labelitemi, \labelitemii, \labelitemiii, and \labelitemiv. To cause the second-level list to be bulleted, you just define \labelitemii to be •. \@itemspacing and \@itemdepth are the analogs of \@enumspacing and \@enumdepth.

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\enumerate ==
  BEGIN
    if \@enumdepth > 3
      then errormessage: "Too deeply nested".
      else \@enumdepth :=L \@enumdepth + 1
           \@enumctr :=L eval(enum@\romannumeral\the\@enumdepth)
           \list{\label(\@enumctr)}
                {\usecounter{\@enumctr}
                 \makelabel{LABEL} ==  \hss \llap{LABEL}}
    fi
  END

\endenumerate == \endlist
```

End of historical LaTeX 2.09 comments.

\@enumdepth

```
\newcount\@enumdepth \@enumdepth = 0
```

(*End of definition for* \@enumdepth.)

\c@enumi
\c@enumii
\c@enumii
\c@enumiv

```
\@definecounter{enumi}
\@definecounter{enumii}
\@definecounter{enumiii}
\@definecounter{enumiv}
```

(*End of definition for* `\c@enumi` *and others.*)

`enumerate` (*env.*)

```
\def\enumerate{%
  \ifnum \@enumdepth >\thr@@\@toodeep\else
    \advance\@enumdepth\@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
      \expandafter
      \list
        \csname label\@enumctr\endcsname
        {\usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
  \fi}
\let\endenumerate =\endlist
```

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\itemize ==
  BEGIN
    if \@itemdepth > 3
      then  errormessage: 'Too deeply nested'.
      else \@itemdepth :=L \@itemdepth + 1
           \@itemitem  == eval(labelitem\romannumeral\the\@itemdepth)
           \list{\@nameuse{\@itemitem}}
                 {\makelabel{LABEL} ==  \hss \llap{LABEL}}
    fi
  END

\enditemize ==  \endlist
```

End of historical LaTeX 2.09 comments.

`\@itemdepth`

```
\newcount\@itemdepth \@itemdepth = 0
```

(*End of definition for* `\@itemdepth`.)

`itemize` (*env.*)

```
\def\itemize{%
  \ifnum \@itemdepth >\thr@@\@toodeep\else
    \advance\@itemdepth\@ne
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
    \expandafter
    \list
      \csname\@itemitem\endcsname
      {\def\makelabel##1{\hss\llap{##1}}}%
  \fi}
\let\enditemize =\endlist
⟨/2ekernel⟩
```

File 40
ltboxes.dtx

1 LaTeX Box commands

\makebox `\makebox[⟨wid⟩][⟨pos⟩]{⟨obj⟩}`
Puts ⟨*obj*⟩ in an `\hbox` of width ⟨*wid*⟩, positioned by ⟨*pos*⟩.

The possible ⟨*pos*⟩ are:
`s` stretched,
`l` flushleft,
`r` flushright,
`c` (default) centred.

If ⟨*wid*⟩ is missing, then ⟨*pos*⟩ is also missing and ⟨*obj*⟩ is put in an `\hbox` of its natural width.

`\makebox(⟨x⟩,⟨y⟩)[⟨pos⟩]{⟨obj⟩}`
Puts ⟨*obj*⟩ in an `\hbox` of width $x * $ `\unitlength` and height $y * $ `\unitlength`. ⟨*pos*⟩ arguments are `s`, `l`, `r` or `c` (default) for stretched, flushleft, flushright or centred, and `t` or `b` for top, bottom – or combinations like `tr` or `rb`. Default for horizontal and vertical are centered. Note that in this picture mode version of `\makebox` a [b] aligns on the *bottom* of the text as documented. If you want to align on the *baseline* use `\makebox( , )[b]{\raisebox{0pt}[\height][0pt]{xyz}}` or `\makebox( , )[b]{\smash{xyz}}`

\mbox `\mbox{⟨obj⟩}` The same as `\makebox{⟨obj⟩}`, but is more efficient as no checking for optional arguments is done.

\newsavebox `\newsavebox{\cmd}` : If `\cmd` is undefined, then defines it to be a TEX box register.

\savebox `\savebox{\cmd} ...` : `\cmd` is defined to be a TEX box register, and the '...' are any `\makebox` arguments. It is like `\makebox`, except it doesn't produce text but saves the value in `\box \cmd`.

\sbox `\sbox{⟨cmd⟩}{⟨obj⟩}` is an efficient abbreviation for `\savebox{⟨cmd⟩}{⟨obj⟩}`.

lrbox (*env.*) `\begin{lrbox}{⟨cmd⟩}⟨text⟩\end{lrbox}` is equivalent to `\sbox{⟨cmd⟩}{⟨text⟩}`
except that any white space at the beginning and end of ⟨*text*⟩ is ignored.

\framebox `\framebox ...` : like `\makebox`, except it puts a 'frame' around the box. The frame is made of lines of thickness `\fboxrule`, separated by space `\fboxsep` from the text – except for `\framebox(X,Y) ...` , where the thickness of the lines is as for the picture environment, and there is no separation added.

\fbox `\fbox{⟨obj⟩}` is an abbreviation for `\framebox{⟨obj⟩}`.

\parbox `\parbox[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}{⟨text⟩}` : Makes a box with `\hsize` ⟨*width*⟩, positioned by ⟨*pos*⟩ as follows: c : `\vcenter` (placed in `$...$` if not in math mode) b : `\vbox` t : `\vtop` default value is c. Sets `\hsize` := ⟨*width*⟩ and calls `\@parboxrestore`, which does the following: Restores the original definitions of:

```
\par
\\
\-
\'
 \`
\=
```

Resets the following parameters:

\parindent	=	0pt	
\parskip	=	0pt	added 20 Jan 87
\linewidth	=	\hsize	
\@totalleftmargin	=	0pt	
\leftskip	=	0pt	
\rightskip	=	0pt	
\@rightskip	=	0pt	
\parfillskip	=	0pt plus 1fil	
\lineskip	=	\normallineskip	
\baselineskip	=	\normalbaselineskip	

Calls \sloppy

Note: \@arrayparboxrestore same as \@parboxrestore but it doesn't restore \\.

minipage (*env.*) minipage : Similar to \parbox, except it also makes this look like a page by setting \textwidth == \columnwidth == box width

changes footnotes by redefining:

\@mpfn == mpfootnote
\thempfn == \thempfootnote
\@footnotetext == \@mpfootnotetext

resets the following list environment parameters:

\@listdepth == \@mplistdepth

where \@mplistdepth is initialized to zero,

and executes \@minipagerestore to allow the document style to reset any other parameters it desires. It sets @minipage true, and resets \everypar to set it false. This switch keeps \addvspace from putting space at the top of a minipage.

Change added 24 May 89: \minipage sets @minipage globally; \endminipage resets it false.

\rule \rule[⟨*raised*⟩]{⟨*width*⟩}{⟨*height*⟩} : Makes a ⟨*width*⟩ ∗ ⟨*height*⟩ rule, raised ⟨*raised*⟩.

\underline \underline{⟨*text*⟩} : Makes an underlined hbox with ⟨*text*⟩ in it.

\raisebox \raisebox{⟨*distance*⟩}[⟨*height*⟩][⟨*depth*⟩]{⟨*box*⟩} :

Raises ⟨*box*⟩ up by ⟨*distance*⟩ length (down if ⟨*distance*⟩ negative). Makes TEX think that the new box extends ⟨*height*⟩ above the line and ⟨*depth*⟩ below, for a total vertical length of ⟨*height*⟩+⟨*depth*⟩. Default values of ⟨*height*⟩ & ⟨*depth*⟩ = actual height and depth of box in new position.

```
⟨*2ekernel⟩
\message{boxes,}
```

\makebox \makebox User level command just looks for optional [or (.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\makebox}{Make \makebox robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\makebox{%
  \leavevmode
  \@ifnextchar(%)
    \@makepicbox
    {\@ifnextchar[\@makebox\mbox}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\makebox}{Make \makebox robust}%
⟨latexrelease⟩\def\makebox{%
⟨latexrelease⟩  \leavevmode
⟨latexrelease⟩  \@ifnextchar(%)
⟨latexrelease⟩    \@makepicbox
⟨latexrelease⟩    {\@ifnextchar[\@makebox\mbox}}%
⟨latexrelease⟩\expandafter\let\csname makebox \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\makebox`.)

`\mbox` The basic horizontal box command for LaTeX.

```
\DeclareRobustCommand\mbox[1]{\leavevmode\hbox{#1}}
```

(*End of definition for* `\mbox`.)

`\@makebox` Look for a possible second optional argument (defaults to `c`).

```
\def\@makebox[#1]{%
  \@ifnextchar [{\@imakebox[#1]}{\@imakebox[#1][c]}}
```

(*End of definition for* `\@makebox`.)

`\@begin@tempboxa` Helper macro for supporting `\height`, `\width` etc. Grab `#1` into `\@tempboxa` and measure it.

```
\long\def\@begin@tempboxa#1#2{%
   \begingroup
     \setbox\@tempboxa#1{\color@begingroup#2\color@endgroup}%
     \def\width{\wd\@tempboxa}%
     \def\height{\ht\@tempboxa}%
     \def\depth{\dp\@tempboxa}%
     \let\totalheight\@ovri
     \totalheight\height
     \advance\totalheight\depth}
```

(*End of definition for* `\@begin@tempboxa`.)

`\@end@tempboxa` End the group started by `\@begin@tempboxa`, so that the scope of `\height` only includes the 'length' argument to the user-command.

```
\let\@end@tempboxa\endgroup
```

(*End of definition for* `\@end@tempboxa`.)

`\bm@c` `\bm@l` `\bm@r` `\bm@s` `\bm@t` `\bm@b` Set up spacing.

```
\def\bm@c{\hss\unhbox\@tempboxa\hss}
\def\bm@l{\unhbox\@tempboxa\hss}\let\bm@t\bm@l
\def\bm@r{\hss\unhbox\@tempboxa}\let\bm@b\bm@r
\def\bm@s{\unhbox\@tempboxa}
```

(*End of definition for* `\bm@c` *and others.*)

\@imakebox Internal form of \makebox.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                  {\@imakebox}{Unknown alignment warning}%
⟨*2ekernel | latexrelease⟩
\long\def\@imakebox[#1][#2]#3{%
  \@begin@tempboxa\hbox{#3}%
    \setlength\@tempdima{#1}%       support calc
    \hb@xt@\@tempdima{%
     \expandafter\ifx\csname bm@#2\endcsname\relax
       \bm@c
       \@latex@warning{Unexpected alignment #2}%
     \else
       \csname bm@#2\endcsname
     \fi}%
  \@end@tempboxa}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@imakebox}{Unknown alignment warning}%
⟨latexrelease⟩\long\def\@imakebox[#1][#2]#3{%
⟨latexrelease⟩  \@begin@tempboxa\hbox{#3}%
⟨latexrelease⟩    \setlength\@tempdima{#1}%       support calc
⟨latexrelease⟩    \hb@xt@\@tempdima{\csname bm@#2\endcsname}%
⟨latexrelease⟩  \@end@tempboxa}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \@imakebox.)

\@makepicbox Picture mode form of \makebox.

```
\def\@makepicbox(#1,#2){%
  \@ifnextchar[{\@imakepicbox(#1,#2)}{\@imakepicbox(#1,#2)[]}}
```

(*End of definition for* \@makepicbox.)

\@imakepicbox picture mode version

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@imakepicbox}{default units}%
\long\def\@imakepicbox(#1,#2)[#3]#4{%
  \@defaultunitsset\@tempdimc{#2}\unitlength
  \vbox to\@tempdimc
   {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
    \let\mb@t\vss
    \@tfor\reserved@a :=#3\do{%
      \if s\reserved@a
        \let\mb@l\relax\let\mb@r\relax
      \else
        \expandafter\let\csname mb@\reserved@a\endcsname\relax
      \fi}%
    \mb@t
    \@defaultunitsset\@tempdimc{#1}\unitlength
```

```
    \hb@xt@\@tempdimc{\mb@l #4\mb@r}%
    \mb@b
```

This kern ensures that a `b` option aligns on the bottom of the text rather than the baseline. this is the documented behaviour in the LaTeX Book. The kern is removed in compatibility mode.

```
    \kern\z@}}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@imakepicbox}{default units}%
⟨latexrelease⟩\long\def\@imakepicbox(#1,#2)[#3]#4{%
⟨latexrelease⟩  \vbox to#2\unitlength
⟨latexrelease⟩   {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
⟨latexrelease⟩    \let\mb@t\vss
⟨latexrelease⟩    \@tfor\reserved@a :=#3\do{%
⟨latexrelease⟩      \if s\reserved@a
⟨latexrelease⟩        \let\mb@l\relax\let\mb@r\relax
⟨latexrelease⟩      \else
⟨latexrelease⟩        \expandafter\let\csname mb@\reserved@a\endcsname\relax
⟨latexrelease⟩      \fi}%
⟨latexrelease⟩    \mb@t
⟨latexrelease⟩    \hb@xt@ #1\unitlength{\mb@l #4\mb@r}%
⟨latexrelease⟩    \mb@b
⟨latexrelease⟩    \kern\z@}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@imakepicbox`.)

`\set@color` This macro is initially a no-op, but the color package will redefine it to insert a `\special`.

```
\let\set@color\relax
```

(*End of definition for* `\set@color`.)

`\color@begingroup` `\color@setgroup` `\normalcolor` `\color@hbox` `\color@vbox` `\color@endbox`

In the past these macros were initially no-ops, and the color package redefined redefine them to be `\begingroup`, `\endgroup`, `\begingroup\set@color`, `\hbox\bgroup\color@begingroup`, `\color@endgroup\egroup`. and ⟨*set to main document color*⟩ respectively.

Nowadays we always set the group already in the kernel as this makes the coding simpler.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                  {\color@begingroup}{color group settings}%
\let\color@begingroup\begingroup
\def\color@setgroup{\color@begingroup}        % changed further in color package
\let\normalcolor\relax                        % remains untouched; only changed in a color pa
\def\color@hbox{\hbox\bgroup\color@begingroup}
\def\color@vbox{\vbox\bgroup\color@begingroup}
\def\color@endbox{\color@endgroup\egroup}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\color@begingroup}{color group settings}%
⟨latexrelease⟩
⟨latexrelease⟩\let\color@begingroup\relax
⟨latexrelease⟩\let\color@setgroup\relax
⟨latexrelease⟩\let\normalcolor\relax
⟨latexrelease⟩\let\color@hbox\relax
⟨latexrelease⟩\let\color@vbox\relax
⟨latexrelease⟩\let\color@endbox\relax
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\color@begingroup` *and others.*)

`\color@endgroup` This macro is separated out because it received an update in 2024, so requires its own rollback.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/11/01}%
⟨latexrelease⟩                  {\color@endgroup}{color group settings}%
```

Beside `\endgraf` for handling vertical boxes we also reset `\if@endpe` as we are leaving the context.

```
\def\color@endgroup{\endgraf\@endpefalse\endgroup}
⟨latexrelease⟩
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                  {\color@endgroup}{color group settings}%
⟨latexrelease⟩\def\color@endgroup{\endgraf\endgroup}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\color@endgroup}{color group settings}%
⟨latexrelease⟩
⟨latexrelease⟩\let\color@endgroup\relax
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\color@endgroup`.)

`\newsavebox` Allocate a new 'savebox'.

```
\def\newsavebox#1{\@ifdefinable{#1}{\newbox#1}}
```

(*End of definition for* `\newsavebox`.)

`\savebox` Save `#1` in a box register.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\savebox}{Make \savebox robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\savebox[1]{%
```

```
  \@ifnextchar(%)
    {\@savepicbox#1}{\@ifnextchar[{\@savebox#1}{\sbox#1}}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\savebox}{Make \savebox robust}%
⟨latexrelease⟩\def\savebox#1{%
⟨latexrelease⟩  \@ifnextchar(%)
⟨latexrelease⟩    {\@savepicbox#1}{\@ifnextchar[{\@savebox#1}{\sbox#1}}}%
⟨latexrelease⟩\expandafter\let\csname savebox \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\savebox`.)

`\sbox` Save `#1` in a box register.

```
\DeclareRobustCommand\sbox[2]{\setbox#1\hbox{%
  \color@setgroup#2\color@endgroup}}
```

(*End of definition for* `\sbox`.)

`\@savebox` Look for second optional argument.

```
\def\@savebox#1[#2]{%
  \@ifnextchar [{\@isavebox#1[#2]}{\@isavebox#1[#2][c]}}
```

(*End of definition for* `\@savebox`.)

`\@isavebox`

```
\long\def\@isavebox#1[#2][#3]#4{%
  \sbox#1{\@imakebox[#2][#3]{#4}}}
```

(*End of definition for* `\@isavebox`.)

`\@savepicbox` Picture mode version of `\savebox`.

```
\def\@savepicbox#1(#2,#3){%
  \@ifnextchar[%]
    {\@isavepicbox#1(#2,#3)}{\@isavepicbox#1(#2,#3)[]}}
```

(*End of definition for* `\@savepicbox`.)

`\@isavepicbox` Picture mode version of `\savebox`.

```
\long\def\@isavepicbox#1(#2,#3)[#4]#5{%
  \sbox#1{\@imakepicbox(#2,#3)[#4]{#5}}}
```

(*End of definition for* `\@isavepicbox`.)

`\lrbox` `lrbox`: the new environment form of `\sbox`. Use `\aftergroup` tricks to enable a *local* assignment to be made to the box, in a way that it still has an effect *outside* the `lrbox` environment.

```
\def\lrbox#1{%
  \edef\reserved@a{%
    \endgroup
    \setbox#1\hbox{%
      \begingroup\aftergroup}%
        \def\noexpand\@currenvir{\@currenvir}%
```

```
        \def\noexpand\@currenvline{\on@line}}%
  \reserved@a
    \@endpefalse
    \color@setgroup
      \ignorespaces}
```

(*End of definition for* `\lrbox`.)

`\endlrbox` End the `lrbox` environment.

```
\def\endlrbox{\unskip\color@endgroup}
```

(*End of definition for* `\endlrbox`.)

`\usebox` unchanged

```
\DeclareRobustCommand\usebox[1]{\leavevmode\copy #1\relax}
```

(*End of definition for* `\usebox`.)

`\frame` The following definition of `\frame` was written by Pavel Curtis (Extra space removed 14 Jan 88) RmS 92/08/24: Replaced occurrence of `\@halfwidth` by `\@wholewidth`

```
\DeclareRobustCommand\frame[1]{%
  \leavevmode
  \hbox{%
    \hskip-\@wholewidth
    \vbox{%
      \vskip-\@wholewidth
      \hrule \@height\@wholewidth
      \hbox{%
        \vrule\@width\@wholewidth
        #1%
        \vrule\@width\@wholewidth}%
      \hrule\@height\@wholewidth
      \vskip-\@wholewidth}%
    \hskip-\@wholewidth}}
```

(*End of definition for* `\frame`.)

`\fboxrule` `\fboxsep` user level parameters,

```
\newdimen\fboxrule
\newdimen\fboxsep
```

(*End of definition for* `\fboxrule` *and* `\fboxsep`.)

`\fbox` Abbreviated framed box command.

```
\DeclareRobustCommand\fbox[1]{%
  \leavevmode
  \setbox\@tempboxa\hbox{%
    \color@begingroup
      \kern\fboxsep{#1}\kern\fboxsep
    \color@endgroup}%
  \@frameb@x\relax}
```

(*End of definition for* `\fbox`.)

\framebox Framed version of \makebox.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\framebox}{Make \framebox robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\framebox{%
  \@ifnextchar(%)
    \@framepicbox{\@ifnextchar[\@framebox\fbox}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\framebox}{Make \framebox robust}%
⟨latexrelease⟩\def\framebox{%
⟨latexrelease⟩  \@ifnextchar(%)
⟨latexrelease⟩    \@framepicbox{\@ifnextchar[\@framebox\fbox}}%
⟨latexrelease⟩\expandafter\let\csname framebox \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \framebox.)

\@framebox Deal with optional arguments.

```
\def\@framebox[#1]{%
  \@ifnextchar[%]
    {\@iframebox[#1]}%
    {\@iframebox[#1][c]}}
```

(*End of definition for* \@framebox.)

\@iframebox The handling the optional arguments. In order to set the whole box, including the frame to the specified dimension, we first determine that dimension from the natural size of the text, #3. calculated width.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                  {\@iframebox}{Unknown alignment warning}%
⟨*2ekernel | latexrelease⟩
\long\def\@iframebox[#1][#2]#3{%
  \leavevmode
  \@begin@tempboxa\hbox{#3}%
    \setlength\@tempdima{#1}%
    \setbox\@tempboxa\hb@xt@\@tempdima
         {\kern\fboxsep
          \expandafter\ifx\csname bm@#2\endcsname\relax
            \bm@c
            \@latex@warning{Unexpected alignment #2}%
          \else
            \csname bm@#2\endcsname
          \fi
          \kern\fboxsep}%
    \@frameb@x{\kern-\fboxrule}%
  \@end@tempboxa}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
```

```
⟨latexrelease⟩                  {\@iframebox}{Unknown alignment warning}%
⟨latexrelease⟩\long\def\@iframebox[#1][#2]#3{%
⟨latexrelease⟩  \leavevmode
⟨latexrelease⟩  \@begin@tempboxa\hbox{#3}%
⟨latexrelease⟩    \setlength\@tempdima{#1}%
⟨latexrelease⟩    \setbox\@tempboxa\hb@xt@\@tempdima
⟨latexrelease⟩         {\kern\fboxsep\csname bm@#2\endcsname\kern\fboxsep}%
⟨latexrelease⟩    \@frameb@x{\kern-\fboxrule}%
⟨latexrelease⟩  \@end@tempboxa}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@iframebox`.)

`\@frameb@x` Common part of `\framebox` and `\fbox`. `#1` is a negative kern in the `\framebox` case so that the vertical rules do not add to the width of the box.

```
\def\@frameb@x#1{%
  \@tempdima\fboxrule
  \advance\@tempdima\fboxsep
  \advance\@tempdima\dp\@tempboxa
  \hbox{%
    \lower\@tempdima\hbox{%
      \vbox{%
        \hrule\@height\fboxrule
        \hbox{%
          \vrule\@width\fboxrule
          #1%
          \vbox{%
            \vskip\fboxsep
            \box\@tempboxa
            \vskip\fboxsep}%
          #1%
          \vrule\@width\fboxrule}%
        \hrule\@height\fboxrule}%
                          }%
        }%
}
```

(*End of definition for* `\@frameb@x`.)

`\@framepicbox` Picture mode version.

```
\def\@framepicbox(#1,#2){%
  \@ifnextchar[{\@iframepicbox(#1,#2)}{\@iframepicbox(#1,#2)[]}}
```

(*End of definition for* `\@framepicbox`.)

`\@iframepicbox` Picture mode version.

```
\long\def\@iframepicbox(#1,#2)[#3]#4{%
  \frame{\@imakepicbox(#1,#2)[#3]{#4}}}
```

(*End of definition for* `\@iframepicbox`.)

\parbox The main vertical-box command for LaTeX.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\parbox}{Make \parbox robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\parbox{%
  \@ifnextchar[%]
    \@iparbox
    {\@iiiparbox c\relax[s]}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\parbox}{Make \parbox robust}%
⟨latexrelease⟩\def\parbox{%
⟨latexrelease⟩  \@ifnextchar[%]
⟨latexrelease⟩    \@iparbox
⟨latexrelease⟩    {\@iiiparbox c\relax[s]}}%
⟨latexrelease⟩\expandafter\let\csname parbox \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \parbox.)

\@iparbox Optional argument handling.

```
\def\@iparbox[#1]{%
  \@ifnextchar[%]
    {\@iiparbox{#1}}%
    {\@iiiparbox{#1}\relax[s]}}
```

(*End of definition for* \@iparbox.)

\@iiparbox Optional argument handling.

```
\def\@iiparbox#1[#2]{%
  \@ifnextchar[%]
    {\@iiiparbox{#1}{#2}}%
    {\@iiiparbox{#1}{#2}[#1]}}
```

(*End of definition for* \@iiparbox.)

\@iiiparbox The internal version of \parbox.

\@parboxto

```
\let\@parboxto\@empty
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2023/11/01}%
⟨latexrelease⟩                  {\@iiiparbox}{Unknown alignment warning}%
⟨*2ekernel | latexrelease⟩
\long\def\@iiiparbox#1#2[#3]#4#5{%
  \leavevmode
  \@pboxswfalse
  \setlength\@tempdima{#4}%
  \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@@par}%
    \ifx\relax#2\else
      \setlength\@tempdimb{#2}%
      \edef\@parboxto{to\the\@tempdimb}%
    \fi
```

```
    \if#1b\vbox
    \else\if #1t\vtop
    \else\ifmmode\vcenter
    \else\@pboxswtrue $\vcenter
    \fi\fi\fi
    \@parboxto{\let\hss\vss\let\unhbox\unvbox
     \expandafter\ifx\csname bm@#3\endcsname\relax
       \bm@c
       \@latex@warning{Unexpected alignment #3}%
     \else
       \csname bm@#3\endcsname
     \fi}%
    \if@pboxsw \m@th$\fi
  \@end@tempboxa}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@iiiparbox}{Unknown alignment warning}%
⟨latexrelease⟩\long\def\@iiiparbox#1#2[#3]#4#5{%
⟨latexrelease⟩  \leavevmode
⟨latexrelease⟩  \@pboxswfalse
⟨latexrelease⟩  \setlength\@tempdima{#4}%
⟨latexrelease⟩  \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@@par}%
⟨latexrelease⟩    \ifx\relax#2\else
⟨latexrelease⟩      \setlength\@tempdimb{#2}%
⟨latexrelease⟩      \edef\@parboxto{to\the\@tempdimb}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩    \if#1b\vbox
⟨latexrelease⟩    \else\if #1t\vtop
⟨latexrelease⟩    \else\ifmmode\vcenter
⟨latexrelease⟩    \else\@pboxswtrue $\vcenter
⟨latexrelease⟩    \fi\fi\fi
⟨latexrelease⟩    \@parboxto{\let\hss\vss\let\unhbox\unvbox
⟨latexrelease⟩       \csname bm@#3\endcsname}%
⟨latexrelease⟩    \if@pboxsw \m@th$\fi
⟨latexrelease⟩  \@end@tempboxa}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@iiiparbox` *and* `\@parboxto`.)

`\@arrayparboxrestore` Restore various paragraph parameters.

The rational for allowing two normally global flags to be set locally here was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within boxes or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in `\@setnobreak`; otherwise this command will be redundant.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017-04-15}%
⟨latexrelease⟩                 {\normallineskiplimit}
⟨latexrelease⟩                 {reset \lineskiplimit}%
⟨*2ekernel | latexrelease⟩
```

```
\def\@arrayparboxrestore{%
  \let\if@nobreak\iffalse
  \let\if@noskipsec\iffalse
  \let\par\@@par
  \let\-\@dischyph
```

Redefined accents to allow changes in font encoding

```
  \let\'\@acci\let\`\@accii\let\=\@acciii
  \parindent\z@ \parskip\z@skip
  \everypar{}%
  \linewidth\hsize
  \@totalleftmargin\z@
  \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
  \parfillskip\@flushglue
  \lineskip\normallineskip
  \lineskiplimit\normallineskiplimit
  \baselineskip\normalbaselineskip
  \sloppy}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000-00-00}%
⟨latexrelease⟩                 {\normallineskiplimit}
⟨latexrelease⟩                 {reset \lineskiplimit}%
⟨latexrelease⟩\def\@arrayparboxrestore{%
⟨latexrelease⟩  \let\if@nobreak\iffalse
⟨latexrelease⟩  \let\if@noskipsec\iffalse
⟨latexrelease⟩  \let\par\@@par
⟨latexrelease⟩  \let\-\@dischyph
⟨latexrelease⟩  \let\'\@acci\let\`\@accii\let\=\@acciii
⟨latexrelease⟩  \parindent\z@ \parskip\z@skip
⟨latexrelease⟩  \everypar{}%
⟨latexrelease⟩  \linewidth\hsize
⟨latexrelease⟩  \@totalleftmargin\z@
⟨latexrelease⟩  \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
⟨latexrelease⟩  \parfillskip\@flushglue \lineskip\normallineskip
⟨latexrelease⟩  \baselineskip\normalbaselineskip
⟨latexrelease⟩  \sloppy}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@arrayparboxrestore`.)

`\@parboxrestore` Restore various paragraph parameters, and also `\\`.

```
\def\@parboxrestore{\@arrayparboxrestore\let\\\@normalcr}
```

(*End of definition for* `\@parboxrestore`.)

`\if@minipage` Switch that is true at the start of a minipage.

```
\def\@minipagefalse{\global\let\if@minipage\iffalse}
\def\@minipagetrue {\global\let\if@minipage\iftrue}
\@minipagefalse
```

(*End of definition for* `\if@minipage`.)

\if@in@minipage@env

```
\newif\if@in@minipage@env
```

(*End of definition for* `\if@in@minipage@env`.)

\minipage Essentially an environment form of `\parbox`.

```
\def\minipage{%
  \@ifnextchar[%]
    \@iminipage
    {\@iiiminipage c\relax[s]}}
```

(*End of definition for* `\minipage`.)

\@iminipage Optional argument handling.

```
\def\@iminipage[#1]{%
  \@ifnextchar[%]
    {\@iiminipage{#1}}%
    {\@iiiminipage{#1}\relax[s]}}
```

(*End of definition for* `\@iminipage`.)

\@iiminipage Optional argument handling.

```
\def\@iiminipage#1[#2]{%
  \@ifnextchar[%]
    {\@iiiminipage{#1}{#2}}%
    {\@iiiminipage{#1}{#2}[#1]}}
```

(*End of definition for* `\@iiminipage`.)

\@iiiminipage Internal form of `minipage`.

```
\def\@iiiminipage#1#2[#3]#4{%
  \leavevmode
  \@pboxswfalse
  \setlength\@tempdima{#4}%
  \def\@mpargs{{#1}{#2}[#3]{#4}}%
  \setbox\@tempboxa\vbox\bgroup
    \color@begingroup
      \hsize\@tempdima
      \textwidth\hsize \columnwidth\hsize
```

We check for nested minipages inside the box so that there is always a group resetting the switch even if the code does not use `\begin` to start the minipage.

```
      \if@in@minipage@env
```

We only issue a warning if the outer minipage contained footnotes because that is the problematical case.

```
        \ifvoid\@mpfootins\else
          \@latex@warning{Nested minipage:
            footnotes may be misplaced}%
        \fi
      \else
        \@in@minipage@envtrue
      \fi
```

```
      \@parboxrestore
      \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
      \let\@footnotetext\@mpfootnotetext
      \let\@listdepth\@mplistdepth \@mplistdepth\z@
      \@minipagerestore
      \@setminipage}
```

(*End of definition for* `\@iiiminipage`.)

`\@minipagerestore` Hook so that other styles can reset other commands in a minipage.

```
\let\@minipagerestore=\relax
```

(*End of definition for* `\@minipagerestore`.)

`\endminipage`

```
\def\endminipage{%
    \par
    \unskip
    \ifvoid\@mpfootins\else
      \vskip\skip\@mpfootins
      \normalcolor
      \footnoterule
      \unvbox\@mpfootins
    \fi
    \@minipagefalse   %% added 24 May 89
  \color@endgroup
  \egroup
  \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}
```

(*End of definition for* `\endminipage`.)

`\@mplistdepth` `\@mpfootins` Versions of `\@listdepth` and `\footins` local to minipage.

```
\newcount\@mplistdepth
\newinsert\@mpfootins
```

(*End of definition for* `\@mplistdepth` *and* `\@mpfootins`.)

`\@mpfootnotetext` Minipage version of `\@footnotetext`.

Final `\strut` added 27 Mar 89, on suggestion by Don Hosek

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                  {\@mpfootnotetext}{footnotetext tagging}%
\long\def\@mpfootnotetext#1{%
  \global\setbox\@mpfootins\vbox{%
    \unvbox\@mpfootins
    \reset@font\footnotesize
    \hsize\columnwidth
    \@parboxrestore
    \def\@currentcounter{mpfootnote}%
    \protected@edef\@currentlabel
         {\csname p@mpfootnote\endcsname\@thefnmark}%
    \color@begingroup
      \@makefntext{%
        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
```

```
    \par
    \color@endgroup}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                 {\@mpfootnotetext}{footnotetext tagging}%
⟨latexrelease⟩\long\def\@mpfootnotetext#1{%
⟨latexrelease⟩  \global\setbox\@mpfootins\vbox{%
⟨latexrelease⟩    \unvbox\@mpfootins
⟨latexrelease⟩    \reset@font\footnotesize
⟨latexrelease⟩    \hsize\columnwidth
⟨latexrelease⟩    \@parboxrestore
⟨latexrelease⟩    \protected@edef\@currentlabel
⟨latexrelease⟩         {\csname p@mpfootnote\endcsname\@thefnmark}%
⟨latexrelease⟩    \color@begingroup
⟨latexrelease⟩      \@makefntext{%
⟨latexrelease⟩        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
⟨latexrelease⟩    \par
⟨latexrelease⟩    \color@endgroup}}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@mpfootnotetext}{footnotetext tagging}%
⟨latexrelease⟩
⟨latexrelease⟩\long\def\@mpfootnotetext#1{%
⟨latexrelease⟩  \global\setbox\@mpfootins\vbox{%
⟨latexrelease⟩    \unvbox\@mpfootins
⟨latexrelease⟩    \reset@font\footnotesize
⟨latexrelease⟩    \hsize\columnwidth
⟨latexrelease⟩    \@parboxrestore
⟨latexrelease⟩    \protected@edef\@currentlabel
⟨latexrelease⟩         {\csname p@mpfootnote\endcsname\@thefnmark}%
⟨latexrelease⟩    \color@begingroup
⟨latexrelease⟩      \@makefntext{%
⟨latexrelease⟩        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
⟨latexrelease⟩    \color@endgroup}}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@mpfootnotetext`.)

```
\newif\if@pboxsw
```

`\rule` Draw a rule of the specified size.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                 {\rule}{Make \rule robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\rule}{Make \rule robust}%
⟨latexrelease⟩\def\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
```

```
⟨latexrelease⟩\expandafter\let\csname rule \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\rule`.)

`\@rule` Internal form of `\rule`.

```
\def\@rule[#1]#2#3{%
  \leavevmode
  \hbox{%
    \setlength\@tempdima{#1}%
    \setlength\@tempdimb{#2}%
    \setlength\@tempdimc{#3}%
    \advance\@tempdimc\@tempdima
    \vrule\@width\@tempdimb\@height\@tempdimc\@depth-\@tempdima}}
```

(*End of definition for* `\@rule`.)

`\@@underline` Saved primitive `\underline`.

```
\let\@@underline\underline
```

(*End of definition for* `\@@underline`.)

`\underline` LaTeX version works outside math.

```
\DeclareRobustCommand\underline[1]{%
  \relax
  \ifmmode\@@underline{#1}%
  \else $\@@underline{\hbox{#1}}\m@th$\relax\fi}
```

(*End of definition for* `\underline`.)

`\raisebox` Raise a box, and change its vertical dimensions.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\raisebox}{Make \raisebox robust}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand\raisebox[1]{%
  \leavevmode
  \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\raisebox}{Make \raisebox robust}%
⟨latexrelease⟩\def\raisebox#1{%
⟨latexrelease⟩  \leavevmode
⟨latexrelease⟩  \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}}
⟨latexrelease⟩\expandafter\let\csname raisebox \endcsname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\raisebox`.)

`\@rsbox` Optional argument handling.

```
\def\@rsbox#1[#2]{%
  \@ifnextchar[{\@iirsbox{#1}[#2]}{\@irsbox{#1}[#2]}}
```

(*End of definition for* `\@rsbox`.)

`\@argrsbox` ...

(*End of definition for* `\@argrsbox`.)

`\@irsbox` Internal version of `\raisebox` (less than two optional args).

```
\long\def\@irsbox#1[#2]#3{%
  \@begin@tempboxa\hbox{#3}%
    \setlength\@tempdima{#1}%
    \ifx\\#2\\\else\setlength\@tempdimb{#2}\fi
    \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
    \ifx\\#2\\\else\ht\@tempboxa\@tempdimb\fi
    \box\@tempboxa
  \@end@tempboxa}
```

(*End of definition for* `\@irsbox`.)

`\@iirsbox` Internal version of `\raisebox` (two optional args).

```
\long\def\@iirsbox#1[#2][#3]#4{%
  \@begin@tempboxa\hbox{#4}%
    \setlength\@tempdima{#1}%
    \setlength\@tempdimb{#2}%
    \setlength\dimen@{#3}%
    \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
    \ht\@tempboxa\@tempdimb
    \dp\@tempboxa\dimen@
    \box\@tempboxa
  \@end@tempboxa}
```

(*End of definition for* `\@iirsbox`.)

`\@finalstrut` This macro adds a special strut the *depth* of the box given as `#1`, and height and width 0pt. It is used for ensuring that the last line of a paragraph has the correct depth in 'p' columns of tables and in footnotes. In vertical mode nothing is done, as adding the strut (as done in 2.09) would start a new paragraph. It would be possible to inspect `\prevdepth` to check the depth of the just-completed paragraph, but we do not do that here. Actually we do even less now, skip the vmode test as it broke tabular 'p' columns. .

The `\nobreak` was added (1995/10/31) to allow hyphenation of the final word of the paragraph.

In 2024 we changed the macro to account for vertical mode. In that case we use a strut produced with `\hrule` to avoid starting a new paragraph (resulting in spurious extra line) and also account for the `\prevdepth` of the previous line.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/06/01}%
⟨latexrelease⟩                  {\@finalstrut}{final strut correction}%
\def\@finalstrut#1{%
  \unskip
  \ifhmode \nobreak
  \else
```

If we are in vmode we now back up by a baseline.

```
    \vskip-\baselineskip
  \fi
```

Finally we unconditionally use `\vrule`.

```
  \vrule\@width\z@\@height\z@\@depth\dp#1}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@finalstrut}{final strut correction}%
⟨latexrelease⟩\def\@finalstrut#1{%
⟨latexrelease⟩   \unskip\ifhmode\nobreak\fi
⟨latexrelease⟩   \vrule\@width\z@\@height\z@\@depth\dp#1}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@finalstrut`.)

1.1 Some low-level constructs

The following commands are basically inherited from plain TEX.

`\leftline` `\rightline` `\centerline` `\@@line`

These macros place text on a full line either centred or left or right adjusted.

```
\def\@@line{\hb@xt@\hsize}
\DeclareRobustCommand\leftline[1]{\@@line{#1\hss}}
\DeclareRobustCommand\rightline[1]{\@@line{\hss#1}}
\DeclareRobustCommand\centerline[1]{\@@line{\hss#1\hss}}
```

(*End of definition for* `\leftline` *and others.*)

`\rlap` `\llap` `\clap`

These macros place text to the left or right of the current reference point without taking up space.

```
\DeclareRobustCommand\rlap[1]{\hb@xt@\z@{#1\hss}}
\DeclareRobustCommand\llap[1]{\hb@xt@\z@{\hss#1}}
```

And here is the version that centers, it was initially introduced by `mathtools`.

```
\DeclareRobustCommand\clap[1]{\hb@xt@\z@{\hss#1\hss}}
```

(*End of definition for* `\rlap`, `\llap`, *and* `\clap`.)

```
⟨/2ekernel⟩
```

File 41
lttab.dtx

1 Tabbing, Tabular and Array Environments

This section deals with 'Lining It Up in Columns'. First the `tabbing` environment is defined, and then in second part, `tabular` together with its variants, `tabular*` and `array`.

Note that the `tabular` defined here is essentially the original LaTeX 2.09 version, not the extended version described in *The LaTeX Companion*. Use the `array` package to obtain the extended version.

1.1 tabbing

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\dimen(\@firsttab + i) = distance of tab stop i from left margin
       0 <= i <= 15 (?).

\dimen\@firsttab is initialized to \@totalleftmargin, so it starts
    at the prevailing left margin.

\@maxtab          = number of highest defined tab register
                      probably = \@firsttab + 12
\@nxttabmar = tab stop number of next line's left margin
\@curtabmar = tab stop number of current line's left margin
\@curtab    = number of the current tab. At start of line,
                    it equals \@curtabmar
\@hightab   = largest tab number currently defined.
\@tabpush   = depth of \pushtab's

\box\@curline     = contents of current line, excluding left margin
                      skip, and excluding contents of current field
\box\@curfield    = contents of current field

@rjfield          = switch: T iff the last field of the line should
                      be right-justified at the right margin.

\tabbingsep           = distance left by the \' command between the
                      current position and the field that is
                      "left-shifted".

UTILITY MACROS
 \@stopfield  : closes the current field
 \@addfield   : adds the current field to the current line.
 \@contfield  : continues the current field
 \@startfield : begins the next field
 \@stopline   : closes the current line and outputs it
```

```
  \@startline  : starts the next line
  \@ifatmargin : an \if that is true iff the current line.
                 has width zero

\@startline ==
 BEGIN
  \@curtabmar :=G \@nxttabmar
  \@curtab :=G \@curtabmar
  \box\@curline :=G null
  \@startfield
  \strut
 END

\@stopline ==
 BEGIN
  \unskip
  \@stopfield
  if @rjfield = T
    then  @rjfield :=G F
          \@tempdima := \@totalleftmargin + \linewidth
          \hb@xt@ \@tempdima{\@itemfudge
                             \hskip \dimen\@curtabmar
                             \box\@curline
                             \hfil
                             \box\@curfield}
    else \@addfield
         \hbox {\@itemfudge
                \hskip \dimen\@curtabmar
                \box\@curline}
  fi
 END

\@startfield ==
 BEGIN
   \box\@curfield :=G \hbox {
 END

\@stopfield ==
 BEGIN
    }
 END

\@contfield ==
 BEGIN
  \box\@curfield :=G \hbox { \unhbox\@currfield  %%} brace matching
 END
\@addfield ==
 BEGIN
  \box\@curline :=G \unbox\@curline * \unbox\@curfield
 END
```

```
\@ifatmargin ==
 BEGIN
  if  dim of box\@curline = 0pt  then
 END


\tabbing ==
 BEGIN
  \lineskip :=L 0pt
  \> == \@rtab
  \< == \@ltab
  \= == \@settab
  \+ == \@tabplus
  \- == \@tabminus
  \` == \@tabrj
  \' == \@tablab
  \\ == BEGIN \@stopline \@startline END
  \\[DIST] == BEGIN
              \@stopline \vskip DIST \@startline\ignorespaces END
  \\* == BEGIN \@stopline \penalty 10000 \@startline END
  \\*[DIST] == BEGIN \@stopline \penalty 10000 \vskip DIST
                    \@startline\ignorespaces              END
  \@hightab :=  \@nxttabmar :=G \@firsttab
  \@tabpush :=G 0
  \dimen\@firsttab := \@totalleftmargin
  @rjfield :=G F
  \trivlist  \item\relax
  if @minipage = F then \vskip \parskip fi
  \box\@tabfbox = \rlap{\indent\the\everypar}
                        % note: \the\everypar sets @inlabel :=G F
  \@itemfudge == BEGIN \box\@tabfbox END
  \@startline
  \ignorespaces
 END

\@endtabbing ==
 BEGIN
  \@stopline
  if \@tabpush > 0 then error message: "unmatched \poptabs'' fi
  \endtrivlist
 END

\@rtab ==
 BEGIN
  \@stopfield
  \@addfield
  if \@curtab < \@hightab
    then \@curtab :=G \@curtab + 1
    else error message "Undefined Tab"   fi
```

```
  \@tempdima := \dimen\@curtab - \dimen\@curtabmar
                        - width of box \@curline
  \box\@curline :=G \hbox{\unhbox\@curline + \hskip\@tempdima}
  \@startfield
 END

\@settab ==
 BEGIN
  \@stopfield
  \@addfield
  if \@curtab < \@maxtab
    then \@curtab :=G \@curtab+1
    else error message: "Too many tabs"     fi
  if \@curtab > \@hightab
    then \@hightab :=L \@curtab     fi
  \dimen\@curtab :=L \dimen\@curtabmar + width of \box\@curline
  \@startfield
 END

\@ltab ==
 BEGIN
  \@ifatmargin
    then if \@curtabmar > \@firsttab
            then \@curtab :=G \@curtab - 1
                 \@curtabmar :=G \@curtabmar - 1
            else error message "Too many untabs"       fi
    else error message "Left tab in middle of line"
  fi
 END

\@tabplus ==
 BEGIN
       if  \@nxttabmar < \@hightab
           then \@nxttabmar :=G \@nxttabmar+1
           else error message "Undefined tab"
       fi
 END

\@tabminus ==
 BEGIN
       if \@nxttabmar > \@firsttab
           then \@nxttabmar :=G \@nxttabmar-1
           else error message "Too many untabs"
       fi
 END

\@tabrj ==
 BEGIN \@stopfield
       \@addfield
       @rjfield :=G T
```

```
        \@startfield
 END

\@tablab ==
 BEGIN \@stopfield
     \box\@curline G:= \hbox{\box\@curline %% 'G' added 17 Jun 86
                              \hskip - width of \box\@curfield
                              \hskip -\tabbingsep
                              \box\@curfield
                              \hskip \tabbingsep }
        \@startfield
 END

\pushtabs ==
  BEGIN
    \@stopfield
    \@tabpush :=G \@tabpush + 1
    \begingroup
    \@contfield
  END

\poptabs ==
 BEGIN
   \@stopfield
   if \@tabpush > 0
     then \endgroup
          \@tabpush :=G \@tabpush - 1
     else error message: "Too many \poptabs''
   fi
   \@contfield
 END
```

End of historical LaTeX 2.09 comments.

\a The accents \‘ , \’ , and \= that have been redefined inside a tabbing environment can be called by typing \a‘ , \a’ , and \a=. The macro \a is defined in ltoutenc.dtx.

(*End of definition for* \a.)

The ‘2ekernel’ code ensures that a \usepackage{autotabg} is essentially ignored if a ‘full’ format is being used that has picture mode already in the format.

```
⟨2ekernel⟩\expandafter\let\csname ver@autotabg.sty\endcsname\fmtversion
```

\@firsttab
\@maxtab

```
⟨*2ekernel⟩
\newdimen\@gtempa
\chardef\@firsttab=\the\allocationnumber
\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
\newdimen\@gtempa
\chardef\@maxtab=\the\allocationnumber
\dimen\@firsttab=0pt
```

(*End of definition for* `\@firsttab` *and* `\@maxtab`.)

`\@nxttabmar`
`\@curtabmar`
`\@curtab`
`\@hightab`
`\@tabpush`

```
\newcount\@nxttabmar
\newcount\@curtabmar
\newcount\@curtab
\newcount\@hightab
\newcount\@tabpush
```

(*End of definition for* `\@nxttabmar` *and others.*)

`\@curline`
`\@curfield`
`\@tabfbox`

```
\newbox\@curline
\newbox\@curfield
\newbox\@tabfbox
```

(*End of definition for* `\@curline`, `\@curfield`, *and* `\@tabfbox`.)

`\if@rjfield`

```
\newif\if@rjfield
```

(*End of definition for* `\if@rjfield`.)

`\@startline` It is, in some sense, an error if the current margin tab setting is higher than the value of `\@hightab` (which is a local variable). That this is allowed is a fundamental design flaw which is not going to be corrected now.

```
\def\@startline{%
     \ifnum \@nxttabmar >\@hightab
       \@badtab
       \global\@nxttabmar \@hightab
     \fi
     \global\@curtabmar \@nxttabmar
     \global\@curtab \@curtabmar
     \global\setbox\@curline \hbox {}%
     \@startfield
     \strut}
```

(*End of definition for* `\@startline`.)

`\@stopline`

```
\def\@stopline{%
  \unskip
  \@stopfield
  \if@rjfield
    \global\@rjfieldfalse
    \@tempdima\@totalleftmargin
    \advance\@tempdima\linewidth
    \hb@xt@\@tempdima{%
      \@itemfudge\hskip\dimen\@curtabmar
      \box\@curline
      \hfil
      \box\@curfield}%
  \else
    \@addfield
    \hbox{\@itemfudge\hskip\dimen\@curtabmar\box\@curline}%
  \fi}
```

(*End of definition for* `\@stopline`.)

`\@startfield`

```
\def\@startfield{%
  \global\setbox\@curfield\hbox\bgroup\color@begingroup}
```

(*End of definition for* `\@startfield`.)

`\@stopfield`

```
\def\@stopfield{%
  \color@endgroup\egroup}
```

(*End of definition for* `\@stopfield`.)

`\@contfield`

```
\def\@contfield{%
  \global\setbox\@curfield\hbox\bgroup\color@begingroup
  \unhbox\@curfield}
```

(*End of definition for* `\@contfield`.)

`\@addfield`

```
\def\@addfield{\global\setbox\@curline\hbox{\unhbox
     \@curline\unhbox\@curfield}}
```

(*End of definition for* `\@addfield`.)

`\@ifatmargin`

```
\def\@ifatmargin{\ifdim \wd\@curline =\z@}
```

(*End of definition for* `\@ifatmargin`.)

`\@tabcr`

```
\protected\def\@tabcr{\@stopline \@ifstar{\penalty \@M \@xtabcr}\@xtabcr}
```

(*End of definition for* `\@tabcr`.)

`\@xtabcr`

```
\def\@xtabcr{\@ifnextchar[\@itabcr{\@startline\ignorespaces}}
```

(*End of definition for* `\@xtabcr`.)

`\@itabcr`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@itabcr}{Tabbing calc syntax}%
\def\@itabcr[#1]{\@vspace@calcify{#1}\@startline\ignorespaces}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@itabcr}{Tabbing calc syntax}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@itabcr[#1]{\vskip #1\@startline\ignorespaces}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`tabbing` (*env.*) We use `\relax` to prevent `\item` from scanning too far.

`\tabbing`

```
\def\tabbing{\lineskip \z@skip\let\>\@rtab\let\<\@ltab\let\=\@settab
     \let\+\@tabplus\let\-\@tabminus\let\`\@tabrj\let\'\@tablab
     \let\\=\@tabcr
     \@hightab\@firsttab
     \global\@nxttabmar\@firsttab
     \dimen\@firsttab\@totalleftmargin
     \global\@tabpush\z@ \global\@rjfieldfalse
     \trivlist \item\relax
     \if@minipage\else\vskip\parskip\fi

     \setbox\@tabfbox\hbox{%
       \rlap{\hskip\@totalleftmargin\indent\the\everypar}}%
     \def\@itemfudge{\box\@tabfbox}%
     \@startline\ignorespaces}
```

`\endtabbing`

```
\def\endtabbing{%
  \@stopline\ifnum\@tabpush >\z@ \@badpoptabs \fi\endtrivlist}
```

Omitted `\global` added to `\@rtab` 17 Jun 86

`\@rtab`

```
\def\@rtab{\@stopfield\@addfield\ifnum \@curtab<\@hightab
      \global\advance\@curtab \@ne \else\@badtab\fi
      \@tempdima\dimen\@curtab
      \advance\@tempdima -\dimen\@curtabmar
      \advance\@tempdima -\wd\@curline
      \global\setbox\@curline\hbox{\unhbox\@curline\hskip\@tempdima}%
      \@startfield\ignorespaces}
```

`\@settab`

```
\def\@settab{\@stopfield\@addfield
  \ifnum \@curtab <\@maxtab
    \ifnum\@curtab =\@hightab
      \advance\@hightab \@ne
    \fi
    \global\advance\@curtab \@ne
  \else
    \@latex@error{Tab overflow}\@ehd
  \fi
  \dimen\@curtab \dimen\@curtabmar
  \advance\dimen\@curtab \wd\@curline
  \@startfield
  \ignorespaces}
```

`\@ltab`

```
\def\@ltab{\@ifatmargin\ifnum\@curtabmar >\@firsttab
      \global\advance\@curtab \m@ne \global\advance\@curtabmar\m@ne\else
      \@badtab\fi\else
      \@latex@error{\string\<\space in mid line}\@ehd\fi\ignorespaces}
```

`\@tabplus`

```
\def\@tabplus{%
  \ifnum\@nxttabmar<\@hightab
```

```
    \global\advance\@nxttabmar\@ne
  \else
    \@badtab
  \fi
  \ignorespaces}
```

\@tabminus

```
\def\@tabminus{%
  \ifnum\@nxttabmar>\@firsttab
    \global\advance\@nxttabmar\m@ne
  \else
    \@badtab
  \fi
  \ignorespaces}
```

\@tabrj

```
\def\@tabrj{%
  \@stopfield\@addfield\global\@rjfieldtrue\@startfield\ignorespaces}
```

\setbox\@curline made \global in \@tablab. 17 Jun 86

\@tablab

```
\def\@tablab{%
  \@stopfield
  \global\setbox\@curline\hbox{%
    \box\@curline
    \hskip-\wd\@curfield \hskip-\tabbingsep
    \box\@curfield
    \hskip\tabbingsep}%
  \@startfield
  \ignorespaces}
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\pushtabs}{Make commands robust}%
```

\pushtabs

```
\DeclareRobustCommand\pushtabs{%
  \@stopfield\@addfield\global\advance\@tabpush \@ne \begingroup
       \@contfield}
```

It is, in some sense, an error if, after the endgroup, the current tab setting is higher than the new value of \@hightab (which is a local variable). That this is allowed is a fundamental design flaw which is not going to be corrected now.

\poptabs

```
\DeclareRobustCommand\poptabs{\@stopfield\@addfield
  \ifnum \@tabpush >\z@
    \endgroup
    \global\advance\@tabpush \m@ne
    \ifnum \@curtab >\@hightab
      \global \@curtab \@hightab
      \@badtab
    \fi
  \else
    \@badpoptabs
  \fi
  \@contfield}
```

```
\DeclareRobustCommand\kill{\@stopfield\@startline\ignorespaces}
```

(*End of definition for* `\@itabcr` *and others.*)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\pushtabs}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\pushtabs
⟨latexrelease⟩\kernel@make@fragile\poptabs
⟨latexrelease⟩\kernel@make@fragile\kill
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\tabbingsep`

```
\newdimen\tabbingsep
```

(*End of definition for* `\tabbingsep`.)

1.2 array and tabular environments

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
ARRAY PARAMETERS:
 \arraycolsep
      : half the width separating columns in an array environment
 \tabcolsep
      : half the width separating columns in a tabular environment
 \arrayrulewidth
      : width of rules
 \doublerulesep
      : space between adjacent rules in array or tabular
 \arraystretch
      : line spacing in array and tabular environments is done by
        placing a strut in every row of height and depth
        \arraystretch times the height and depth of the strut
        produced by an ordinary \strut command.

PREAMBLE:
 The PREAMBLE argument of an array or tabular environment can
 contain the following:
   l,r,c   : indicate where entry is to be placed.
   |       : for vertical rule
   @{EXP} : inserts the text EXP in every column.
            \arraycolsep or \tabcolsep  spacing is suppressed.
   *{N}{PRE} : equivalent to writing N copies of PRE in the preamble.
               PRE may contain *{N'}{EXP'} expressions.
   p{LEN} : makes entry in parbox of width LEN.

SPECIAL ARRAY COMMANDS:
```

```
\multicolumn{N}{FORMAT}{ITEM} : replaces the next N column
 items by ITEM, formatted according to FORMAT.
 FORMAT should contain at most one l,r or c.
 If it contains none, then ITEM is ignored.

\vline : draws a vertical line the height of the current row.  May
         appear in an array element entry.
\hline : draws a horizontal line between rows.  Must appear either
         before the first entry (to appear above the first row) or
          right after a \\ command.  If followed by another \hline,
          then adds a \vskip of \doublerulesep.

\cline{i-j} : draws horizontal lines between rows covering columns
              i through j, inclusive.  Multiple commands may follow
              one another to provide lines covering several disjoint
              columns
\extracolsep{WIDTH} : for use inside an @ in the preamble.  Causes
            a WIDTH space to be added between columns for the rest
             of the columns.  This is in addition to the ordinary
             intercolumn space.

\array ==
  BEGIN
    \@acol    == \@arrayacol
    \@classz  == \@arrayclassz
    \@classiv == \@arrayclassiv
    \\        == \@arraycr
    \@halignto == NULL
    \@tabarray
  END

\endarray{NAME} ==  BEGIN  \crcr }}  END

\tabular  ==
  BEGIN
    \@halignto == NULL
    \@tabular
  END

\tabular*{WIDTH} ==
  BEGIN
    \@halignto == to WIDTH
    \@tabular
  END

\@tabular ==
  BEGIN
    \leavevmode
    \hbox { $
       \@acol    == \@tabacol
```

```
        \@classz  == \@tabclassz
        \@classiv == \@tabclassiv
        \\        == \@tabularcr
        \@tabarray
  END

\endtabular == BEGIN \crcr}} $} END

\@tabarray == if next char = [ then \@array else \@array[c] fi

\@array[POS]{PREAMBLE} ==
  BEGIN
    define \@arstrutbox to make \@arstrut produce strut of height
      and depth \arraystretch times the height and
      depth of a normal strut.
    \@mkpream{PREAMBLE}
    \@preamble == \halign \@halignto {\tabskip=0pt\@arstrut
                            eval{\@preamble}\tabskip = 0pt\cr %%}
    \@startpbox == \@@startpbox
    \@endpbox == \@@endpbox
    if POS = t then \vtop
               else if POS = b then \vbox
                                   else \vcenter
    fi              fi
   {
    \par          ==L {} % changed 92/09/18
    \@sharp       == #
    \protect      == \relax
    \lineskip     :=L 0pt
    \baselineskip :=L 0pt
    \@preamble
  END


\@arraycr ==
 BEGIN
   $              %% Prevents extra space at end of row's last entry.
   if next char = [
    then  \@argarraycr
    else  $ \cr          %% Needed to balance $
 END

\@argarraycr[LENGTH] ==
 BEGIN
   $                    %% Needed to balance $ of \@arraycr
   if LENGTH > 0
     then  \@tempdima := depth of \@arstrutbox + LENGTH
           \vrule height 0pt width 0pt depth \@tempdima
           \cr
     else  \cr \noalign{\vskip LENGTH}
```

END

\@tabularcr and \@argtabularcr same as \@arraycr and \@argarraycr except without the extra $'s.

End of historical LaTeX 2.09 comments.

\extracolsep This command needs to expand during the tabular preamble construction so can't be robust.

```
\def\extracolsep#1{\tabskip #1\relax}
```

(*End of definition for* \extracolsep.)

array \array(array.)

```
\def\array{\let\@acol\@arrayacol \let\@classz\@arrayclassz
 \let\@classiv\@arrayclassiv
 \let\\\@arraycr\let\@halignto\@empty\@tabarray}
```

(*End of definition for* \array.)

\endarray
\endtabular
\endtabular*

```
\def\endarray{\crcr\egroup\egroup}
\def\endtabular{\crcr\egroup\egroup $\egroup}
\expandafter \let \csname endtabular*\endcsname = \endtabular
```

(*End of definition for* \endarray, \endtabular, *and* \endtabular*.)

tabular \tabular(ular.)

```
\def\tabular{\let\@halignto\@empty\@tabular}
```

(*End of definition for* \tabular.)

\tabular* Note that the change to use \setlength slightly alters the timing of the expansion and use of the length in #1 but this is very unlikely to have any practical effect.

```
\@namedef{tabular*}#1{%
 \setlength\dimen@{#1}%
   \edef\@halignto{to\the\dimen@}\@tabular}
```

(*End of definition for* \tabular*.)

\@tabular

```
\def\@tabular{\leavevmode \hbox \bgroup $\let\@acol\@tabacol
   \let\@classz\@tabclassz
   \let\@classiv\@tabclassiv \let\\\@tabularcr\@tabarray}
```

(*End of definition for* \@tabular.)

\@tabarray RmS 91/11/04 added \m@th.

```
\def\@tabarray{\m@th\@ifnextchar[\@array{\@array[c]}}
```

(*End of definition for* \@tabarray.)

RmS 1993/11/03 changed \halign to \ialign and removed superfluous \tabskip assignment

\@array

```
\def\@array[#1]#2{%
  \if #1t\vtop \else \if#1b\vbox \else \vcenter \fi\fi
```

```
\bgroup
```

This next bit of code sets up the strut and then builds the halign and its preamble according to the specification in the second argument.

This code has been moved inside the box. A side effect of this has been to expose what was a buglet in the previous version: since the `\@arstrut` below is expanded and contains an `\ifmmode` then it could produce an unnecessary extra box in every row, thus wasting 'lots of' main memory.

```
\setbox\@arstrutbox\hbox{%
  \vrule \@height\arraystretch\ht\strutbox
         \@depth\arraystretch \dp\strutbox
         \@width\z@}%
\@mkpream{#2}%
\edef\@preamble{%
  \ialign \noexpand\@halignto
    \bgroup \@arstrut \@preamble \tabskip\z@skip \cr}%
```

That is the end of setting up the preamble; now we reset things before executing the halign built-up in `\@preamble`. The restorations could be done by introducing an extra group, thus saving tokens.

```
\let\@startpbox\@@startpbox \let\@endpbox\@@endpbox
\let\tabularnewline\\%
  \let\par\@empty
  \let\@sharp##%
  \set@typeset@protect
  \lineskip\z@skip\baselineskip\z@skip
```

If the parsing of the preamble goes wrong there my be some characters left which TEX then tries to typeset, i.e., we would be in horizontal mode. That would produce an endless loop because the `\halign` expects vertical mode thus issues a `\par` but that is a no-op at this point. So we better test this case issue some error message and make a crude recovery by ending that horizontal mode with force. A better fix would be to ensure that we never pick up more than a single character token (not done).

```
  \ifhmode \@preamerr\z@ \@@par\fi
  \@preamble}
```

(*End of definition for* `\@array`.)

`\@arraycr` Array version of `\\`.

```
\protected\def\@arraycr{%
  ${\ifnum0=`}\fi\@ifstar\@xarraycr\@xarraycr}
```

(*End of definition for* `\@arraycr`.)

`\@arraycr`

```
\def\@xarraycr{\@ifnextchar[\@argarraycr{\ifnum0=`{\fi}${}\cr}}
```

(*End of definition for* `\@arraycr`.)

`\@argarraycr`

```
\def\@argarraycr[#1]{%
  \ifnum0=`{\fi}${}\ifdim #1>\z@ \@xargarraycr{#1}\else
   \@yargarraycr{#1}\fi}
```

(*End of definition for* `\@argarraycr`.)

\tabularnewline Tabular version of \\.

```
\let\tabularnewline\relax
```

(*End of definition for* \tabularnewline.)

\@tabularcr

```
\protected\def\@tabularcr{%
  {\ifnum0=`}\fi\@ifstar\@xtabularcr\@xtabularcr}
```

(*End of definition for* \@tabularcr.)

\@xtabularcr

```
\def\@xtabularcr{\@ifnextchar[\@argtabularcr{\ifnum0=`{\fi}\cr}}
```

(*End of definition for* \@xtabularcr.)

\@argtabularcr

```
\def\@argtabularcr[#1]{%
  \ifnum0=`{\fi}%
    \ifdim #1>\z@
      \unskip\@xargarraycr{#1}%
    \else
      \@yargarraycr{#1}%
    \fi}
```

(*End of definition for* \@argtabularcr.)

\@xargarraycr

```
\def\@xargarraycr#1{\@tempdima #1\advance\@tempdima \dp \@arstrutbox
   \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr}
```

(*End of definition for* \@xargarraycr.)

\@yargarraycr

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@yargarraycr}{tabular support calc syntax}%
\def\@yargarraycr#1{\cr\noalign{\@vspace@calcify{#1}}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@yargarraycr}{tabular support calc syntax}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@yargarraycr#1{\cr\noalign{\vskip #1}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \@yargarraycr.)

`\multicolumn` *Historical LaTeX 2.09 comments (not necessarily accurate any more):*

```
\multicolumn{NUMBER}{FORMAT}{ITEM} ==
  BEGIN
  \multispan{NUMBER}
  \begingroup
  \@addamp == null
  \@mkpream{FORMAT}
  \@sharp == ITEM
  \protect == \relax
  \@startpbox == \@@startpbox
  \@endpbox == \@@endpbox
  \@arstrut
  \@preamble
  \endgroup
  END
```

End of historical LaTeX 2.09 comments.

The command `\def\@addamp{}` was removed from `\multicolumn` on 6 Dec 86 because it caused embedded array environments not to work. I think that it was included originally to prevent an error message if the 2nd argument to the `\multicolumn` command had two column specifiers.

8 Feb 89 — `\hbox{}` added after `\@preamble` to correct bug that occurred if `\multicolumn` preceded `\\[D]` with `D > 0`, caused by `\\[]` command doing an `\unskip`, which removed `\tabcolsep` glue inserted by `\multicolumn`.

This has been made long so that, for example, a `p`-column can contain multiple paragraphs; maybe the arguments of `@`-expressions should also be able to contain multiple paragraphs.

```
\long\def\multicolumn#1#2#3{\multispan{#1}\begingroup
  \@mkpream{#2}%
  \def\@sharp{#3}\set@typeset@protect
  \let\@startpbox\@@startpbox\let\@endpbox\@@endpbox
  \@arstrut \@preamble\hbox{}\endgroup\ignorespaces}
```

(*End of definition for* `\multicolumn`.)

Historical LaTeX 2.09 comments (not necessarily accurate any more):

Codes for classes and character numbers of array, tabular and multicolumn arguments.

Character	Class	Number
c	0	0
l	0	1
r	0	2
|	1	-
@	2	-
p	3	-
{@-exp}	4	-
{p-arg}	5	-

`\@testpach \foo` : expands `\foo`, which should be an array parameter

token, and sets \@chclass and \@chnum to its class and number. Uses \@lastchclass to distinguish 4 and 5

```
Preamble error codes
   0: 'illegal character'
   1: 'Missing @-exp'
   2: 'Missing p-arg'

\@addamp ==
  BEGIN if @firstamp = true then @firstamp := false
                                else &                        fi
  END

\@mkpream TOKENLIST ==
  BEGIN
   @firstamp      := T
   \@lastchclass := 6
   \@preamble     == null
   \@sharp        == \relax
   \protect       == BEGIN \noexpand\protect\noexpand END
   \@startpbox    == \relax
   \@endpbox      == \relax
   \@expast{TOKENLIST}
   for \@nextchar := expand(\reserved@a)
     do  \@testpach{\@nextchar}
         case of \@chclass
           0 -> \@classz
           1 -> \@classi
              ...
           5 -> \@classv
         end case
         \@lastchclass := \@chclass
     od
     case of \@lastchclass
        0 -> \hskip \arraycolsep              % lrc
        1 ->                                   % |
        2 -> \@preamerr1 % 'Missing @-exp'    % @
        3 -> \@preamerr2 % 'Missing p-arg'    % p
        4 ->                                   % @-exp
        5 -> \hskip \arraycolsep              % p-exp
     end case
  END

 \@arrayclassz ==
   BEGIN
     \@preamble := \@preamble *
               case of \@lastchclass
                  0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
                  1 -> \@addamp \hskip \arraycolsep
                  2 ->  % impossible
```

```
                3 ->  % impossible
                4 -> \@addamp
                5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
                6 -> \@addamp \hskip \arraycolsep
              end case
            * case of \@chnum
                0 -> \hfil$\relax\@sharp$\hfil
                1 -> $\relax\@sharp$\hfil
                2 -> \hfil$\relax\@sharp$
              end case
  END

\@tabclassz == similar to \@arrayclassz

\@classi ==
 BEGIN
   \@preamble := \@preamble *
                case of \@lastchclass
                   0 -> \hskip \arraycolsep \@arrayrule
                   1 -> \hskip \doublerulesep \@arrayrule
                   2 -> % impossible
                   3 -> % impossible
                   4 -> \@arrayrule
                   5 -> \hskip \arraycolsep \@arrayrule
                   6 -> \@arrayrule
                end case
 END

\@classii ==
 BEGIN
   \@preamble := \@preamble *
                case of \@lastchclass
                   0     ->
                   1     -> \hskip .5\arrayrulewidth
                   2     -> % impossible
                   else ->
                end case
 END

\@classiii ==
 BEGIN
   \@preamble := \@preamble *
              case of \@lastchclass
                0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
                1 -> \@addamp \hskip \arraycolsep
                2 -> % impossible
                3 -> % impossible
                4 -> \@addamp
                5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
                6 -> \@addamp \hskip \arraycolsep
```

```
                end case
 END

\@arrayclassiv  ==
     BEGIN  \@preamble := \@preamble * $ \@nextchar$  END

\@tabclassiv   == same as \@arrayclassv except without the $ ... $

\@classv ==
  BEGIN
   \@preamble :=
       \@preamble * \@startpbox{\@nextchar}\ignorespaces\@sharp
                                \@endpbox
  END

\@expast{S}:
 Sets \reserved@a := S with all instances of *{N}{STRING}
 replaced by N copies of STRING, where N > 0.  An *
 appearing inside braces is ignored, but *-expressions
 inside STRING are expanded, so nested *-expressions are
 handled properly.

\@expast{S} == BEGIN  \@xexpast S *0x\@@  END

\@xexpast S1 *{N}{S2} S3 \@@ ==
 BEGIN
   \reserved@a   := S1
   \@tempcnta := N
   if \@tempcnta > 0
     then  while \@tempcnta > 0 do \reserved@a   := \reserved@a S2
                                   \@tempcnta := \@tempcnta - 1 od
           \reserved@b == \@xexpast
     else  \reserved@b == \@xexnoop
   fi
   \expandafter \reserved@b \reserved@a S3 \@@
 END
```

End of historical LaTeX 2.09 comments.

\@xexnoop

```
\def\@xexnoop #1\@@{}
```

(*End of definition for* \@xexnoop.)

\@expast

```
\def\@expast#1{\@xexpast #1*0x\@@}
```

(*End of definition for* \@expast.)

\@xexpast

```
\def\@xexpast#1*#2#3#4\@@{%
  \edef\reserved@a{#1}%
  \@tempcnta#2\relax
  \ifnum\@tempcnta>\z@
    \@whilenum\@tempcnta>\z@\do
       {\edef\reserved@a{\reserved@a#3}\advance\@tempcnta \m@ne}%
    \let\reserved@b\@xexpast
  \else
    \let\reserved@b\@xexnoop
  \fi
  \expandafter\reserved@b\reserved@a #4\@@}
```

(*End of definition for* `\@xexpast`.)

\if@firstamp
\@addamp

```
\newif\if@firstamp
```

```
\def\@addamp{%
  \if@firstamp
    \@firstampfalse
  \else
    \edef\@preamble{\@preamble &}%
  \fi}
```

(*End of definition for* `\if@firstamp` *and* `\@addamp`.)

\@arrayacol
\@tabacol
\@ampacol
\@acolampacol

```
\def\@arrayacol{\edef\@preamble{\@preamble \hskip \arraycolsep}}
\def\@tabacol{\edef\@preamble{\@preamble \hskip \tabcolsep}}
\def\@ampacol{\@addamp \@acol}
\def\@acolampacol{\@acol\@addamp\@acol}
```

(*End of definition for* `\@arrayacol` *and others.*)

\@mkpream

```
\def\@mkpream#1{\@firstamptrue\@lastchclass6
  \let\@preamble\@empty
  \let\protect\@unexpandable@protect
  \let\@sharp\relax
  \let\@startpbox\relax\let\@endpbox\relax
  \@expast{#1}%
  \expandafter\@tfor \expandafter
    \@nextchar \expandafter:\expandafter=\reserved@a\do
       {\@testpach\@nextchar
    \ifcase \@chclass \@classz \or \@classi \or \@classii \or \@classiii
      \or \@classiv \or\@classv \fi\@lastchclass\@chclass}%
  \ifcase \@lastchclass \@acol
      \or \or \@preamerr \@ne\or \@preamerr \tw@\or \or \@acol \fi}
```

(*End of definition for* `\@mkpream`.)

\@arrayclassz

```
\def\@arrayclassz{\ifcase \@lastchclass \@acolampacol \or \@ampacol \or
   \or \or \@addamp \or
   \@acolampacol \or \@firstampfalse \@acol \fi
\edef\@preamble{\@preamble
  \ifcase \@chnum
     \hfil$\relax\@sharp$\hfil \or $\relax\@sharp$\hfil
    \or \hfil$\relax\@sharp$\fi}}
```

(*End of definition for* `\@arrayclassz`.)

\@tabclassz RmS 91/08/14 inserted extra braces around entry for NFSS

```
\def\@tabclassz{%
  \ifcase\@lastchclass
    \@acolampacol
  \or
    \@ampacol
  \or
  \or
  \or
    \@addamp
  \or
    \@acolampacol
  \or
    \@firstampfalse\@acol
  \fi
  \edef\@preamble{%
    \@preamble{%
      \ifcase\@chnum
        \hfil

        \hskip1sp%
        \ignorespaces\@sharp\unskip\hfil
      \or

        \hskip1sp\ignorespaces\@sharp\unskip\hfil
      \or
        \hfil\hskip1sp\ignorespaces\@sharp\unskip
      \fi}}}
```

(*End of definition for* `\@tabclassz`.)

\@classi

```
\def\@classi{%
  \ifcase\@lastchclass
    \@acol\@arrayrule
  \or
    \@addtopreamble{\hskip \doublerulesep}\@arrayrule
  \or
  \or
  \or
    \@arrayrule
  \or
    \@acol\@arrayrule
  \or
```

```
    \@arrayrule
  \fi}
```

(*End of definition for* `\@classi`.)

`\@classii`

```
\def\@classii{%
  \ifcase\@lastchclass
  \or
    \@addtopreamble{\hskip .5\arrayrulewidth}%
  \fi}
```

(*End of definition for* `\@classii`.)

`\@classiii`

```
\def\@classiii{\ifcase \@lastchclass \@acolampacol \or
   \@addamp\@acol \or
   \or \or \@addamp \or
   \@acolampacol \or \@ampacol \fi}
```

(*End of definition for* `\@classiii`.)

`\@tabclassiv`

```
\def\@tabclassiv{\@addtopreamble\@nextchar}
```

(*End of definition for* `\@tabclassiv`.)

`\@arrayclassiv`

```
\def\@arrayclassiv{\@addtopreamble{$\@nextchar$}}
```

(*End of definition for* `\@arrayclassiv`.)

`\@classv`

```
\def\@classv{\@addtopreamble{\@startpbox{\@nextchar}\ignorespaces
\@sharp\@endpbox}}
```

(*End of definition for* `\@classv`.)

`\@addtopreamble`

```
\def\@addtopreamble#1{\edef\@preamble{\@preamble #1}}
```

(*End of definition for* `\@addtopreamble`.)

`\@chclass`
`\@lastchclass`
`\@chnum`

```
\newcount\@chclass
\newcount\@lastchclass
\newcount\@chnum
```

(*End of definition for* `\@chclass`, `\@lastchclass`, *and* `\@chnum`.)

`\arraycolsep`
`\tabcolsep`
`\arrayrulewidth`
`\doublerulesep`

```
\newdimen\arraycolsep
\newdimen\tabcolsep
\newdimen\arrayrulewidth
\newdimen\doublerulesep
```

(*End of definition for* `\arraycolsep` *and others.*)

`\arraystretch`

```
\def\arraystretch{1}    % Default value.
```

(*End of definition for* `\arraystretch`.)

`\@arstrutbox`
`\@arstrut`

```
\newbox\@arstrutbox
\def\@arstrut{%
  \relax\ifmmode\copy\@arstrutbox\else\unhcopy\@arstrutbox\fi}
```

(*End of definition for* `\@arstrutbox` *and* `\@arstrut`.)

`\@arrayrule`

```
\def\@arrayrule{\@addtopreamble{\hskip -.5\arrayrulewidth
   \vrule \@width \arrayrulewidth\hskip -.5\arrayrulewidth}}
```

(*End of definition for* `\@arrayrule`.)

`\@testpach`

```
\def\@testpach#1{\@chclass \ifnum \@lastchclass=\tw@ 4 \else
    \ifnum \@lastchclass=3 5 \else
     \z@ \if #1c\@chnum \z@ \else
                                \if #1l\@chnum \@ne \else
                                \if #1r\@chnum \tw@ \else
          \@chclass \if #1|\@ne \else
                    \if #1@\tw@ \else
                    \if #1p3 \else \z@ \@preamerr 0\fi
  \fi  \fi  \fi  \fi  \fi  \fi
\fi}
```

(*End of definition for* `\@testpach`.)

`\hline`

```
\def\hline{%
  \noalign{\ifnum0=`}\fi\hrule \@height \arrayrulewidth \futurelet
   \reserved@a\@xhline}
```

(*End of definition for* `\hline`.)

`\@xhline`

```
\def\@xhline{\ifx\reserved@a\hline
               \vskip\doublerulesep
```

Measure from the middle of the rules.

```
               \vskip-\arrayrulewidth
             \fi
      \ifnum0=`{\fi}}
```

(*End of definition for* `\@xhline`.)

`\vline`

```
\def\vline{\vrule \@width \arrayrulewidth}
```

(*End of definition for* `\vline`.)

\cline
\@cline

The old LaTeX2.09 implementation of \cline used up quite a lot of memory and two precious count registers. This new (1995/09/14) implementation does not use any count registers. It is coded in a way that depends heavily on the definition of \multispan so that command has been moved here from the file ltplain.dtx.

These counters are no longer declared.

```
\newcount\@cla
\newcount\@clb
```

```
\def\cline#1{\@cline#1\@nil}
```

```
\def\@cline#1-#2\@nil{%
  \omit
```

Use the counter from \multispan.

```
  \@multicnt#1%
  \advance\@multispan\m@ne
  \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
  \@multicnt#2%
  \advance\@multicnt-#1%
  \advance\@multispan\@ne
```

The original had \unskip at this point, but how could a skip get here ???

```
  \leaders\hrule\@height\arrayrulewidth\hfill
  \cr
```

This is back spacing is fairly horrible, but it is what happened in the old version... An alternative would be to make \cline look ahead for a following \cline as does \hline. This would alter the spacing in existing documents so keep the old version in the kernel. Perhaps a package should do this differently.

```
  \noalign{\vskip-\arrayrulewidth}}
```

(*End of definition for* \cline *and* \@cline.)

\mscount

The \mscount counter is no longer declared, saving a csname and a register. It is declared in compatibility mode.

(*End of definition for* \mscount.)

\multispan
\@multispan
\sp@n

Modify \multispan slightly from its plain TeX definition to allow more efficient code sharing with \multicolumn. Also share a count register with \multiput.

```
\def\multispan{\omit\@multispan}
```

```
\def\@multispan#1{%
  \@multicnt#1\relax
  \loop\ifnum\@multicnt>\@ne \sp@n\repeat}
```

```
\def\sp@n{\span\omit\advance\@multicnt\m@ne}
```

(*End of definition for* \multispan, \@multispan, *and* \sp@n.)

\@startpbox
\@endpbox

Helper macros for 'p' columns.

\@startpbox{⟨*width*⟩} *text* \egroup is essentially \parbox{⟨*width*⟩}{⟨*text*⟩}

\@endpbox is essentially \unskip \strut \par \egroup\hfil (Changed 14 Jan 89) (changed again 1994/05/13)

```
\def\@startpbox#1{\vtop\bgroup \setlength\hsize{#1}\@arrayparboxrestore}
```

```
\def\@endpbox{\@finalstrut\@arstrutbox\par\egroup\hfil}
```

14 Jan 89: Def of `\@endpbox` changed from
`\def\@endpbox{\par\vskip\dp\@arstrutbox\egroup\hfil}`
so vertical spacing works out right if the last line of a 'p' entry has a descender.

(*End of definition for* `\@startpbox` *and* `\@endpbox`.)

`\@@startpbox`
`\@@endpbox`

```
\let\@@startpbox=\@startpbox
\let\@@endpbox=\@endpbox
```

(*End of definition for* `\@@startpbox` *and* `\@@endpbox`.)

⟨/2ekernel⟩

File 42
ltpictur.dtx

1 Picture Mode

Picture mode commands. In addition to the commands available in LaTeX2.09, This section adds the new `\qbezier` command for drawing curves.

`\qbezier` `\qbezier[`⟨*N*⟩`](`⟨*AX,AY*⟩`)(`⟨*BX,BY*⟩`)(`⟨*CX,CY*⟩`)` plots a quadratic Bezier curve from `(`⟨*AX,AY*⟩`)` to `(`⟨*CX,CY*⟩`)`, with `(`⟨*BX,BY*⟩`)` as the third Bezier point, using $N+1$ points equally spaced parametrically. If $N = 0$ (the default value), then a sufficient number of points are used to draw a connected curve–except that at most `\qbeziermax`+1 points are drawn. A "point" is a square of side `\@wholewidth`.

`\bezier` In addition, to be compatible with the old `bezier` package, a variant of this command, `\bezier`, is defined, in which the first argument is not optional.

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\unitlength      = value of dimension argument
\@wholewidth     = current line width
\@halfwidth      = half of current line width
\@linefnt        = font for drawing lines
\@circlefnt      = font for drawing circles

\linethickness{DIM} : Sets the width of horizontal and vertical lines
    in a picture to DIM.  Does not change width of slanted lines
    or circles.   Width of all lines reset by \thinlines and
    \thicklines

\picture(XSIZE,YSIZE)(XORG,YORG)
  BEGIN
    \@picht :=L YSIZE * \unitlength
    box \@picbox :=
         \hb@xt@ XSIZE * \unitlength
           {\hskip -XORG * \unitlength
            \lower YORG * \unitlength
            \hbox{
            \ignorespaces    %% added 13 June 89
  END

\endpicture ==
  BEGIN
                  } \hss }
                  height of \@picbox := \@picht
                  depth  of \@picbox := 0
                  \mbox{\box\@picbox}   %% change 26 Aug 91
  END

\put(X, Y){OBJ} ==
  BEGIN
```

```
    \@killglue
    \raise Y * \unitlength  \hb@xt@ 0pt { \hskip X * \unitlength
                                          OBJ \hss            }
    \ignorespaces
  END

\multiput(X,Y)(DELX,DELY){N}{OBJ} ==
  BEGIN
   \@killglue
   \@multicnt := N
   \@xdim  := X * \unitlength
   \@ydim  := Y * \unitlength
   while \@multicnt > 0
     do \raise \@ydim \hb@xt@ 0pt { \hskip \@xdim
                                          OBJ \hss   }
        \@multicnt := \@multicnt - 1
        \@xdim     := \@xdim + DELX * \unitlength
        \@ydim     := \@ydim + DELY * \unitlength
     od
   \ignorespaces
  END
```

`\shortstack[POS]{TEXT}` : Makes a `\vbox` containing TEXT stacked as a one-column array, positioned l, r or c as indicated by POS.

End of historical LaTeX 2.09 comments.

The '2ekernel' code ensures that a `\usepackage{autopict}` is essentially ignored if a 'full' format is being used that has picture mode already in the format.

```
⟨2ekernel⟩\expandafter\let\csname ver@autopict.sty\endcsname\fmtversion
```

`\@wholewidth`
`\@halfwidth`

```
⟨*2ekernel⟩
\newdimen\@wholewidth
\newdimen\@halfwidth
```

(*End of definition for* `\@wholewidth` *and* `\@halfwidth`.)

`\unitlength`

```
\newdimen\unitlength \unitlength =1pt
```

(*End of definition for* `\unitlength`.)

`\@picbox`
`\@picht`

```
\newbox\@picbox
\newdimen\@picht
```

(*End of definition for* `\@picbox` *and* `\@picht`.)

`\@defaultunitsset` Set a length register, `#1`, accepting number or an etex length expression, `#2`, with default unit, `#3`.

The register name in `#1` can be prefixed by `\advance` so that the register is incremented by the supplied value.

`\@defaultunitsset{\advance\@vxx}{\textwidth-15pt}\unitlength`

`#3` can be a literal unit such as `cm` or a length register such as `\unitlength`.

This is used in all `picture` commands that take picture coordinates. So `\put(2,2)` as previously but now `\put(\textwidth-5cm,0.4\texteight)` Note that you can only use expressions with lengths, `\put(1+2,0)` is not supported.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@defaultunitsset}{default units}%
\def\@defaultunitsset#1#2#3{%
  \@defaultunits#1\dimexpr#2#3\relax\relax\@nnil}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@defaultunitsset}{default units}%
⟨latexrelease⟩\let\@defaultunitsset\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@defaultunitsset`.)

`picture` `\picture` `#1` should be white space.

`#1` should be a `(` (eating any white space before the bracket),

`\pictur@`

```
\long\def\picture#1{\pictur@#1}
\def\pictur@(#1){%
  \@ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)}}
```

(*End of definition for* `\picture` *and* `\pictur@`.)

`\@picture`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@picture}{default units}%
\def\@picture(#1,#2)(#3,#4){%
  \@defaultunitsset\@picht{#2}\unitlength
  \@defaultunitsset\@tempdimc{#1}\unitlength
  \setbox\@picbox\hb@xt@\@tempdimc\bgroup
    \@defaultunitsset\@tempdimc{#3}\unitlength
    \hskip -\@tempdimc
    \@defaultunitsset\@tempdimc{#4}\unitlength
    \lower\@tempdimc\hbox\bgroup
      \ignorespaces}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@picture}{default units}%
⟨latexrelease⟩\def\@picture(#1,#2)(#3,#4){%
⟨latexrelease⟩  \@picht#2\unitlength
⟨latexrelease⟩  \setbox\@picbox\hb@xt@#1\unitlength\bgroup
⟨latexrelease⟩    \hskip -#3\unitlength
⟨latexrelease⟩    \lower #4\unitlength\hbox\bgroup
⟨latexrelease⟩      \ignorespaces}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@picture`.)

`\endpicture`

```
\def\endpicture{%
  \egroup\hss\egroup
    \ht\@picbox\@picht\dp\@picbox\z@
    \mbox{\box\@picbox}}
```

(*End of definition for* `\endpicture`.)

In the definitions of `\put` and `\multiput`, `\hskip` was replaced by `\kern` just in case arg `#3` = "plus". (Bug detected by Don Knuth. changed 20 Jul 87).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\put}{default units}%
⟨latexrelease⟩\expandafter\let\csname put \endcsname\@undefind
\long\def\put(#1,#2)#3{%
  \@killglue
  \@defaultunitsset\@tempdimc{#2}\unitlength
  \raise\@tempdimc
  \hb@xt@\z@{%
    \@defaultunitsset\@tempdimc{#1}\unitlength
    \kern\@tempdimc
    #3\hss}%
  \ignorespaces}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\put}{default units}%
⟨latexrelease⟩\expandafter\let\csname put \endcsname\@undefind
⟨latexrelease⟩\long\def\put(#1,#2)#3{%
⟨latexrelease⟩  \@killglue\raise#2\unitlength
⟨latexrelease⟩  \hb@xt@\z@{\kern#1\unitlength #3\hss}%
⟨latexrelease⟩  \ignorespaces}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\multiput` `#3` had better be a `(`.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
```

```
⟨latexrelease⟩                  {\multiput}{default units}%
⟨latexrelease⟩\expandafter\let\csname multiput \endcsname\@undefind
\def\multiput(#1,#2)#3{%
  \@defaultunitsset\@xdim{#1}\unitlength
  \@defaultunitsset\@ydim{#2}\unitlength
   \@multiput(}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\multiput}{default units}%
⟨latexrelease⟩\expandafter\let\csname multiput \endcsname\@undefind
⟨latexrelease⟩\def\multiput(#1,#2)#3{%
⟨latexrelease⟩  \@xdim #1\unitlength
⟨latexrelease⟩  \@ydim #2\unitlength
⟨latexrelease⟩   \@multiput(}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\multiput`.)

`\@multiput`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@multiput}{default units}%
\long\def\@multiput(#1,#2)#3#4{%
  \@killglue\@multicnt #3\relax
  \@whilenum \@multicnt >\z@\do
    {\raise\@ydim\hb@xt@\z@{\kern\@xdim #4\hss}%
     \advance\@multicnt\m@ne
     \@defaultunitsset{\advance\@xdim}{#1}\unitlength
     \@defaultunitsset{\advance\@ydim}{#2}\unitlength}%
  \ignorespaces}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@multiput}{default units}%
⟨latexrelease⟩\long\def\@multiput(#1,#2)#3#4{%
⟨latexrelease⟩  \@killglue\@multicnt #3\relax
⟨latexrelease⟩  \@whilenum \@multicnt >\z@\do
⟨latexrelease⟩    {\raise\@ydim\hb@xt@\z@{\kern\@xdim #4\hss}%
⟨latexrelease⟩     \advance\@multicnt\m@ne
⟨latexrelease⟩     \advance\@xdim#1\unitlength\advance\@ydim#2\unitlength}%
⟨latexrelease⟩  \ignorespaces}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@multiput`.)

`\@killglue`

```
\def\@killglue{\unskip\@whiledim \lastskip >\z@\do{\unskip}}
```

(*End of definition for* `\@killglue`.)

\thinlines
\thicklines

```
\DeclareRobustCommand\thinlines{\let\@linefnt\tenln
  \let\@circlefnt\tencirc
  \@wholewidth\fontdimen8\tenln \@halfwidth .5\@wholewidth}
\DeclareRobustCommand\thicklines{\let\@linefnt\tenlnw
  \let\@circlefnt\tencircw
  \@wholewidth\fontdimen8\tenlnw \@halfwidth .5\@wholewidth}
```

(*End of definition for* \thinlines *and* \thicklines.)

\linethickness

```
\DeclareRobustCommand*\linethickness[1]
   {\@wholewidth #1\relax \@halfwidth .5\@wholewidth \ignorespaces}
```

(*End of definition for* \linethickness.)

\ishortstack

```
\def\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}
```

(*End of definition for* \ishortstack.)

\@ishortstack

```
\def\@shortstack[#1]{%
  \leavevmode
  \vbox\bgroup
    \baselineskip-\p@\lineskip 3\p@
    \let\mb@l\hss\let\mb@r\hss
    \expandafter\let\csname mb@#1\endcsname\relax
    \let\\\@stackcr
    \@ishortstack}
```

(*End of definition for* \@ishortstack.)

\@ishortstack

```
\def\@ishortstack#1{\ialign{\mb@l {##}\unskip\mb@r\cr #1\crcr}\egroup}
```

(*End of definition for* \@ishortstack.)

\@stackcr
\@ixstackcr

```
\protected\def\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
\def\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}}
```

(*End of definition for* \@stackcr *and* \@ixstackcr.)

\@istackcr

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@istackcr}{\shortstack calc support}%
\def\@istackcr[#1]{\cr\noalign{\@vspace@calcify{#1}}\ignorespaces}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@istackcr}{\shortstack calc support}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@istackcr[#1]{\cr\noalign{\vskip #1}\ignorespaces}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@istackcr`.)

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\line(X,Y){LEN} ==
BEGIN
 \@xarg    := X
 \@yarg    := Y
 \@linelen := LEN * \unitlength
 if \@xarg = 0
     then \@vline
     else if \@yarg = 0
             then \@hline
             else \@sline
          if
 if
END

\@sline ==
 BEGIN
   if \@xarg < 0
     then @negarg := T
          \@xarg  := -\@xarg
          \@yyarg := -\@yarg
     else @negarg := F
          \@yyarg := \@yarg
   fi
   \@tempcnta := |\@yyarg|
   if \@tempcnta > 6
     then error: 'LATEX ERROR: Illegal \line or \vector argument.'
          \@tempcnta := 0
   fi
   \box\@linechar := \hbox{\@linefnt \@getlinechar(\@xarg,\@yyarg) }
    if \@yarg > 0 then \@upordown = \raise
                         \@clnht := 0
                   else \@upordown = \lower
                        \@clnht := height of \box\@linechar
    fi
    \@clnwd  := width of \box\@linechar
    if @negarg
      then \hskip - width of \box\@linechar
           \reserved@a == \hskip - 2* width of box \@linechar
      else \reserved@a == \relax
    fi
 %% Put out integral number of line segments
```

```
    while \@clnwd <  \@linelen
      do  \@upordown \@clnht \copy\@linechar
          \reserved@a
          \@clnht := \@clnht + ht of \box\@linechar
          \@clnwd := \@clnwd + width of \box\@linechar
      od

 %% Put out last segment
    \@clnht := \@clnht - height of \box\@linechar
    \@clnwd := \@clnwd - width of \box\@linechar
    \@tempdima   := \@linelen - \@clnwd
    \@tempdimb   := \@tempdima - width of \box\@linechar
    if @negarg  then \hskip -\@tempdimb
                 else \hskip  \@tempdimb
    fi
    \@tempdima   := 1000 * \@tempdima
    \@tempcnta   := \@tempdima / width of \box\@linechar
    \@tempdima   := (\@tempcnta * ht of \box\@linechar)/1000
    \@clnht := \@clnht + \@tempdima
    if \@linelen < width of box\@linechar
        then \hskip width of box\@linechar
        else \hbox{\@upordown \@clnht \copy\@linechar}
    fi
END

\@hline ==
  BEGIN
    if \@xarg < 0 then  \hskip -\@linelen \fi
    \vrule height \@halfwidth depth \@halfwidth width \@linelen
    if \@xarg < 0 then  \hskip -\@linelen \fi
 END

\@vline == if \@yarg < 0 \@downline else \@upline  fi


\@getlinechar(X,Y) ==
  BEGIN
    \@tempcnta := 8*X - 9
    if Y > 0
      then \@tempcnta := \@tempcnta + Y
      else \@tempcnta := \@tempcnta - Y + 64
    fi
    \char\@tempcnta
  END

\vector(X,Y){LEN} ==
BEGIN
 \@xarg    := X
 \@yarg    := Y
 \@linelen := LEN * \unitlength
```

```
 if \@xarg = 0
    then \@vvector
    else if \@yarg = 0
            then \@hvector
            else \@svector
         if
 if
END

\@hvector ==
  BEGIN
    \@hline
    {\@linefnt if \@xarg < 0 then  \@getlarrow(1,0)
                                else  \@getrarrow(1,0)
                fi}
  END

\@vvector == if \@yarg < 0 \@downvector else \@upvector  fi

\@svector ==
 BEGIN
  \@sline
  \@tempcnta := |\@yarg|
    if  \@tempcnta < 5
       then  \hskip - width of \box\@linechar
             \@upordown \@clnht \hbox
                      {\@linefnt
                       if @negarg then \@getlarrow(\@xarg,\@yyarg)
                                  else \@getrarrow(\@xarg,\@yyarg)
                       fi }
       else  error: 'LaTeX ERROR: Illegal \line or \vector argument.'
    fi
 END

\@getlarrow(X,Y) ==
 BEGIN
  if Y = 0
    then \@tempcnta := '33
    else \@tempcnta := 16 * X  -  9
         \@tempcntb := 2 * Y
         if \@tempcntb > 0
           then  \@tempcnta := \@tempcnta  +  \@tempcntb
           else  \@tempcnta := \@tempcnta  -  \@tempcntb +  64
         fi
  fi
  \char\@tempcnta
 END

\@getrarrow(X,Y) ==
 BEGIN
```

```
\@tempcntb := |Y|
case of \@tempcntb
  0 : \@tempcnta := '55
  1 : if X < 3
        then \@tempcnta :=  24*X - 6
        else if X = 3
               then \@tempcnta := 49
               else \@tempcnta := 58  fi
      fi
  2 : if X < 3
        then \@tempcnta :=  24*X - 3
        else \@tempcnta := 51      % X must = 3
      fi
  3 : \@tempcnta := 16*X - 2
  4 : \@tempcnta := 16*X + 7
endcase
if Y < 0
  then \@tempcnta := \@tempcnta + 64
fi
\char\@tempcnta
END
```

End of historical LaTeX 2.09 comments.

`\if@negarg`

```
\newif\if@negarg
```

(*End of definition for* `\if@negarg`.)

`\line`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\line}{default units}%
⟨latexrelease⟩\expandafter\let\csname line \endcsname\@undefind
\def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
  \@defaultunitsset\@linelen{#3}\unitlength
  \ifdim\@linelen<\z@\@badlinearg\else
    \ifnum\@xarg =\z@ \@vline
      \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
    \fi
  \fi}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\line}{default units}%
⟨latexrelease⟩\expandafter\let\csname line \endcsname\@undefind
⟨latexrelease⟩\def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
⟨latexrelease⟩  \@linelen #3\unitlength
⟨latexrelease⟩  \ifdim\@linelen<\z@\@badlinearg\else
⟨latexrelease⟩    \ifnum\@xarg =\z@ \@vline
⟨latexrelease⟩      \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
⟨latexrelease⟩    \fi
```

```
⟨latexrelease⟩   \fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\line`.)

`\@sline`

```
\def\@sline{%
  \ifnum\@xarg<\z@ \@negargtrue \@xarg -\@xarg \@yyarg -\@yarg
  \else \@negargfalse \@yyarg \@yarg \fi
\ifnum \@yyarg >\z@ \@tempcnta\@yyarg \else \@tempcnta -\@yyarg \fi
\ifnum\@tempcnta>6 \@badlinearg\@tempcnta\z@ \fi
\ifnum\@xarg>6 \@badlinearg\@xarg \@ne \fi
\setbox\@linechar\hbox{\@linefnt\@getlinechar(\@xarg,\@yyarg)}%
```

If we have something like `\line(5,5){30}` the `\@linechar` will not contain a char and later on we will end in an infinite loop. So we check the width of the box and put in something as an emergency fix if necessary.

```
\ifdim\wd\@linechar=\z@
   \setbox\@linechar\hbox{.}%
   \@badlinearg
\fi
\ifnum \@yarg >\z@ \let\@upordown\raise \@clnht\z@
   \else\let\@upordown\lower \@clnht \ht\@linechar\fi
\@clnwd \wd\@linechar
\if@negarg
  \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
\else
     \let\reserved@a\relax
\fi
\@whiledim \@clnwd <\@linelen \do
  {\@upordown\@clnht\copy\@linechar
   \reserved@a
   \advance\@clnht \ht\@linechar
   \advance\@clnwd \wd\@linechar}%
\advance\@clnht -\ht\@linechar
\advance\@clnwd -\wd\@linechar
\@tempdima\@linelen\advance\@tempdima -\@clnwd
\@tempdimb\@tempdima\advance\@tempdimb -\wd\@linechar
\if@negarg \hskip -\@tempdimb \else \hskip \@tempdimb \fi
\multiply\@tempdima \@m
\@tempcnta \@tempdima
\@tempdima \wd\@linechar \divide\@tempcnta \@tempdima
\@tempdima \ht\@linechar \multiply\@tempdima \@tempcnta
\divide\@tempdima \@m
\advance\@clnht \@tempdima
\ifdim \@linelen <\wd\@linechar
   \hskip \wd\@linechar
```

Warn if line gets so short that it can't be printed. But don't warn if it is exactly zero since that was probably deliberate (e.g., to get a vector head only).

```
   \ifdim \@linelen = \z@
   \else
     \@picture@warn
   \fi
```

```
  \else\@upordown\@clnht\copy\@linechar\fi}
```

(*End of definition for* `\@sline`.)

`\@hline`

```
\def\@hline{\ifnum \@xarg <\z@ \hskip -\@linelen \fi
\vrule \@height \@halfwidth \@depth \@halfwidth \@width \@linelen
\ifnum \@xarg <\z@ \hskip -\@linelen \fi}
```

(*End of definition for* `\@hline`.)

`\@getlinechar`

```
\def\@getlinechar(#1,#2){\@tempcnta#1\relax\multiply\@tempcnta 8%
  \advance\@tempcnta -9\ifnum #2>\z@ \advance\@tempcnta #2\relax\else
  \advance\@tempcnta -#2\relax\advance\@tempcnta 64 \fi
  \char\@tempcnta}
```

(*End of definition for* `\@getlinechar`.)

`\vector`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\vector}{default units}%
⟨latexrelease⟩\expandafter\let\csname vector \endcsname\@undefind
\def\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
  \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
  \ifnum\@tempcnta<5\relax
  \@defaultunitsset\@linelen{#3}\unitlength
  \ifdim\@linelen<\z@\@badlinearg\else
    \ifnum\@xarg =\z@ \@vvector
      \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
    \fi
  \fi
  \else\@badlinearg\fi}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\vector}{default units}%
⟨latexrelease⟩\expandafter\let\csname vector \endcsname\@undefind
⟨latexrelease⟩\def\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
⟨latexrelease⟩  \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
⟨latexrelease⟩  \ifnum\@tempcnta<5\relax
⟨latexrelease⟩  \@linelen #3\unitlength
⟨latexrelease⟩  \ifdim\@linelen<\z@\@badlinearg\else
⟨latexrelease⟩    \ifnum\@xarg =\z@ \@vvector
⟨latexrelease⟩      \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \else\@badlinearg\fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\vector`.)

\@hvector

```
\def\@hvector{\@hline\hb@xt@\z@{\@linefnt
 \ifnum \@xarg <\z@ \@getlarrow(1,0)\hss\else
    \hss\@getrarrow(1,0)\fi}}
```

(*End of definition for* \@hvector.)

\@vvector

```
\def\@vvector{\ifnum \@yarg <\z@ \@downvector \else \@upvector \fi}
```

(*End of definition for* \@vvector.)

\@svector

```
\def\@svector{\@sline
  \@tempcnta\@yarg \ifnum\@tempcnta <\z@ \@tempcnta -\@tempcnta\fi
  \ifnum\@tempcnta <5%
    \hskip -\wd\@linechar
    \@upordown\@clnht \hbox{\@linefnt  \if@negarg
    \@getlarrow(\@xarg,\@yyarg)\else \@getrarrow(\@xarg,\@yyarg)\fi}%
  \else\@badlinearg\fi}
```

(*End of definition for* \@svector.)

\@getlarrow

```
\def\@getlarrow(#1,#2){\ifnum #2=\z@ \@tempcnta 27 % '33
  \else
  \@tempcnta #1\relax\multiply\@tempcnta \sixt@@n
  \advance\@tempcnta -9 \@tempcntb #2\relax\multiply\@tempcntb \tw@
  \ifnum \@tempcntb >\z@ \advance\@tempcnta \@tempcntb
  \else\advance\@tempcnta -\@tempcntb\advance\@tempcnta 64
  \fi\fi\char\@tempcnta}
```

(*End of definition for* \@getlarrow.)

\@getrarrow

```
\def\@getrarrow(#1,#2){\@tempcntb #2\relax
\ifnum\@tempcntb <\z@ \@tempcntb -\@tempcntb\relax\fi
\ifcase \@tempcntb\relax \@tempcnta 45  % '55
\or
\ifnum #1<\thr@@ \@tempcnta #1\relax\multiply\@tempcnta
24\advance\@tempcnta -6 \else \ifnum #1=\thr@@ \@tempcnta 49
\else\@tempcnta 58 \fi\fi\or
\ifnum #1<\thr@@ \@tempcnta=#1\relax\multiply\@tempcnta
24\advance\@tempcnta -\thr@@ \else \@tempcnta 51 \fi\or
\@tempcnta #1\relax\multiply\@tempcnta
\sixt@@n \advance\@tempcnta -\tw@ \else
\@tempcnta #1\relax\multiply\@tempcnta
\sixt@@n \advance\@tempcnta 7 \fi\ifnum #2<\z@ \advance\@tempcnta 64 \fi
\char\@tempcnta}
```

(*End of definition for* \@getrarrow.)

\@vline

```
\def\@vline{\ifnum \@yarg <\z@ \@downline \else \@upline\fi}
```

(*End of definition for* `\@vline`.)

`\@upline`

```
\def\@upline{%
  \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
   \@height \@linelen \@depth \z@\hss}}
```

(*End of definition for* `\@upline`.)

`\@downline`

```
\def\@downline{%
  \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
   \@height \z@ \@depth \@linelen \hss}}
```

(*End of definition for* `\@downline`.)

`\@upvector`

```
\def\@upvector{\@upline\setbox\@tempboxa\hbox{\@linefnt\char 54}% '66
  \raise \@linelen \hb@xt@\z@{\lower \ht\@tempboxa\box\@tempboxa\hss}}
```

(*End of definition for* `\@upvector`.)

`\@downvector`

```
\def\@downvector{\@downline\lower \@linelen
      \hb@xt@\z@{\@linefnt\char 63  % '77
      \hss}}
```

(*End of definition for* `\@downvector`.)

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\dashbox{D}(X,Y) ==
  BEGIN
  leave vertical mode
  \hb@xt@ 0pt {
       \baselineskip := 0pt
       \lineskip     := 0pt
  %% HORIZONTAL DASHES
       \@dashdim := X * \unitlength
       \@dashcnt := \@dashdim + 200 % to prevent roundoff error
       \@dashdim := D * \unitlength
       \@dashcnt := \@dashcnt / \@dashdim
       if \@dashcnt is odd
         then \@dashdim := 0pt
              \@dashcnt := (\@dashcnt + 1) / 2
         else \@dashdim := \@dashdim / 2
              \@dashcnt := \@dashcnt / 2 - 1
              \box\@dashbox   := \hbox{\vrule height \@halfwidth
                                 depth \@halfwidth width \@dashdim}
              \put(0,0){\copy\@dashbox}
              \put(0,Y){\copy\@dashbox}
              \put(X,0){\hskip -\@dashdim\copy\@dashbox}
              \put(X,Y){\hskip -\@dashdim\box\@dashbox}
              \@dashdim := 3 * \@dashdim
       fi
```

```
\box\@dashbox := \hbox{\vrule height \@halfwidth
                   depth \@halfwidth width D * \unitlength
                   \hskip D * \unitlength}
\@tempcnta := 0
\put(0,0){\hskip \@dashdim
         while \@tempcnta < \@dascnt
           do \copy\@dashbox
              \@tempcnta := \@tempcnta + 1
           od
        }
\@tempcnta := 0
put(0,Y){\hskip \@dashdim
         while \@tempcnta < \@dascnt
           do \copy\@dashbox
              \@tempcnta := \@tempcnta + 1
           od
        }

%% vertical dashes
\@dashdim := Y * \unitlength
\@dashcnt := \@dashdim + 200 % to prevent roundoff error
\@dashdim := D * \unitlength
\@dashcnt := \@dashcnt / \@dashdim
if \@dashcnt is odd
  then \@dashdim := 0pt
       \@dashcnt := (\@dashcnt + 1) / 2
  else \@dashdim := \@dashdim / 2
       \@dashcnt := \@dashcnt / 2 - 1
       \box\@dashbox   := \hbox{\hskip -\@halfwidth
                                 \vrule width \@wholewidth
                                          height \@dashdim  }
       \put(0,0){\copy\@dashbox}
       \put(X,0){\copy\@dashbox}
       \put(0,Y){\lower\@dashdim\copy\@dashbox}
       \put(X,Y){\lower\@dashdim\copy\@dashbox}
       \@dashdim := 3 * \@dashdim
fi
\box\@dashbox := \hbox{\vrule width \@wholewidth
                          height D * \unitlength       }
\@tempcnta := 0
put(0,0){\hskip -\halfwidth
         \vbox{while \@tempcnta < \@dashcnt
                 do \vskip D*\unitlength
                    \copy\@dashbox
                    \@tempcnta := \@tempcnta + 1
                 od
                \vskip \@dashdim
               } }
\@tempcnta := 0
put(X,0){\hskip -\halfwidth
```

```
                    \vbox{while \@tempcnta < \@dashcnt
                            do \vskip D*\unitlength
                               \copy\@dashbox
                               \@tempcnta := \@tempcnta + 1
                            od
                           \vskip \@dashdim
                          }
                  }
    }     % END DASHES

  \@imakepicbox(X,Y)
 END
```

End of historical LaTeX 2.09 comments.

\dashbox

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\dashbox}{default units}%
⟨latexrelease⟩\expandafter\let\csname dashbox \endcsname\@undefind
\def\dashbox#1(#2,#3){\leavevmode\hb@xt@\z@{\baselineskip \z@skip
\lineskip \z@skip
\@defaultunitsset\@dashdim{#2}\unitlength
\@dashcnt \@dashdim \advance\@dashcnt 200
\@defaultunitsset\@dashdim{#1}\unitlength
\divide\@dashcnt \@dashdim
\ifodd\@dashcnt\@dashdim \z@
\advance\@dashcnt \@ne \divide\@dashcnt \tw@
\else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
\advance\@dashcnt \m@ne
\setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
\@width \@dashdim}\put(0,0){\copy\@dashbox}%
\put(0,#3){\copy\@dashbox}%
\put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
\put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
\multiply\@dashdim \thr@@
\fi
\setbox\@dashbox \hbox{%
  \@defaultunitsset\@tempdimc{#1}\unitlength
  \vrule \@height \@halfwidth \@depth \@halfwidth \@width \@tempdimc
  \hskip\@tempdimc}%
\@tempcnta\z@
\put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
\do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
\put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
\do{\copy\@dashbox\advance\@tempcnta \@ne }}%
\@defaultunitsset\@dashdim{#3}\unitlength
\@dashcnt \@dashdim \advance\@dashcnt 200
\@defaultunitsset\@dashdim{#1}\unitlength
\divide\@dashcnt \@dashdim
\ifodd\@dashcnt \@dashdim \z@
\advance\@dashcnt \@ne \divide\@dashcnt \tw@
\else
```

```
\divide\@dashdim \tw@ \divide\@dashcnt \tw@
\advance\@dashcnt \m@ne
\setbox\@dashbox\hbox{\hskip -\@halfwidth
\vrule \@width \@wholewidth
\@height \@dashdim}\put(0,0){\copy\@dashbox}%
\put(#2,0){\copy\@dashbox}%
\put(0,#3){\lower\@dashdim\copy\@dashbox}%
\put(#2,#3){\lower\@dashdim\copy\@dashbox}%
\multiply\@dashdim \thr@@
\fi
\@defaultunitsset\@tempdimb{#1}\unitlength
\setbox\@dashbox\hbox{%
  \vrule \@width \@wholewidth \@height\@tempdimb}%
\@tempcnta\z@
\put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
\do{\vskip\@tempdimb\copy\@dashbox\advance\@tempcnta \@ne }%
\vskip\@dashdim}}\@tempcnta\z@
\put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
\do{\vskip\@tempdimb\copy\@dashbox\advance\@tempcnta \@ne }%
\vskip\@dashdim}}}\@makepicbox(#2,#3)}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\dashbox}{default units}%
⟨latexrelease⟩\expandafter\let\csname dashbox \endcsname\@undefind
⟨latexrelease⟩\def\dashbox#1(#2,#3){%
⟨latexrelease⟩\leavevmode\hb@xt@\z@{\baselineskip \z@skip
⟨latexrelease⟩\lineskip \z@skip
⟨latexrelease⟩\@dashdim #2\unitlength
⟨latexrelease⟩\@dashcnt \@dashdim \advance\@dashcnt 200
⟨latexrelease⟩\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
⟨latexrelease⟩\ifodd\@dashcnt\@dashdim \z@
⟨latexrelease⟩\advance\@dashcnt \@ne \divide\@dashcnt \tw@
⟨latexrelease⟩\else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
⟨latexrelease⟩\advance\@dashcnt \m@ne
⟨latexrelease⟩\setbox\@dashbox \hbox{%
⟨latexrelease⟩  \vrule \@height \@halfwidth \@depth \@halfwidth
⟨latexrelease⟩  \@width \@dashdim}\put(0,0){\copy\@dashbox}%
⟨latexrelease⟩\put(0,#3){\copy\@dashbox}%
⟨latexrelease⟩\put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
⟨latexrelease⟩\put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
⟨latexrelease⟩\multiply\@dashdim \thr@@
⟨latexrelease⟩\fi
⟨latexrelease⟩\setbox\@dashbox \hbox{%
⟨latexrelease⟩  \vrule \@height \@halfwidth \@depth \@halfwidth
⟨latexrelease⟩  \@width #1\unitlength\hskip #1\unitlength}\@tempcnta\z@
⟨latexrelease⟩\put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
⟨latexrelease⟩\do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
⟨latexrelease⟩\put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
⟨latexrelease⟩\do{\copy\@dashbox\advance\@tempcnta \@ne }}%
⟨latexrelease⟩\@dashdim #3\unitlength
⟨latexrelease⟩\@dashcnt \@dashdim \advance\@dashcnt 200
⟨latexrelease⟩\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
⟨latexrelease⟩\ifodd\@dashcnt \@dashdim \z@
```

```
⟨latexrelease⟩\advance\@dashcnt \@ne \divide\@dashcnt \tw@
⟨latexrelease⟩\else
⟨latexrelease⟩\divide\@dashdim \tw@ \divide\@dashcnt \tw@
⟨latexrelease⟩\advance\@dashcnt \m@ne
⟨latexrelease⟩\setbox\@dashbox\hbox{\hskip -\@halfwidth
⟨latexrelease⟩\vrule \@width \@wholewidth
⟨latexrelease⟩\@height \@dashdim}\put(0,0){\copy\@dashbox}%
⟨latexrelease⟩\put(#2,0){\copy\@dashbox}%
⟨latexrelease⟩\put(0,#3){\lower\@dashdim\copy\@dashbox}%
⟨latexrelease⟩\put(#2,#3){\lower\@dashdim\copy\@dashbox}%
⟨latexrelease⟩\multiply\@dashdim \thr@@
⟨latexrelease⟩\fi
⟨latexrelease⟩\setbox\@dashbox\hbox{\vrule \@width \@wholewidth
⟨latexrelease⟩\@height #1\unitlength}\@tempcnta\z@
⟨latexrelease⟩\put(0,0){%
⟨latexrelease⟩  \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
⟨latexrelease⟩  \do{\vskip #1\unitlength\copy\@dashbox
⟨latexrelease⟩      \advance\@tempcnta\@ne }%
⟨latexrelease⟩  \vskip\@dashdim}}\@tempcnta\z@
⟨latexrelease⟩\put(#2,0){%
⟨latexrelease⟩  \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
⟨latexrelease⟩  \do{\vskip #1\unitlength\copy\@dashbox
⟨latexrelease⟩      \advance\@tempcnta \@ne }%
⟨latexrelease⟩  \vskip\@dashdim}}}\@makepicbox(#2,#3)}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\dashbox`.)

Historical LaTeX 2.09 comments (not necessarily accurate any more):

CIRCLES AND OVALS

USER COMMANDS:

`\circle{D}` : Produces the circle with the diameter as close as possible to D * `\unitlength`. `\put(X,Y){\circle{D}}` puts the circle with its center at (X,Y).

`\oval(X,Y)` : Makes an oval as round as possible that fits in the rectangle of width X * `\unitlength` and height Y * `\unitlength`. The reference point is the center.

`\oval(X,Y)[POS]` : Save as `\oval(X,Y)` except it draws only the half or quadrant of the oval indicated by POS. E.G., `\oval(X,Y)[t]` draws just the top half and `\oval(X,Y)[br]` draws just the bottom right quadrant. In all cases, the reference point is the same as the unqualified `\oval(X,Y)` command.

`\@ovvert {DELTA1} {DELTA2}` : Makes a vbox containing either the left side or the right side of the oval being constructed. The baseline will coincide with the outside bottom edge of the oval; the left side of the box will coincide with the left edge of the vertical

rule. The width of the box will be `\@tempdima`.
DELTA1 and DELTA2 are added to the character number in `\@tempcnta` to get the characters for the top and bottom quarter circle pieces.

`\@ovhorz` : Makes an hbox containing the straight rule for either the top or the bottom of the oval being constructed. The baseline will coincide with bottom edge of the rule; the left side of the box will coincide with the left side of the oval. The width of the box will be `\@ovxx`.

`\@getcirc {DIAM}` : Sets `\@tempcnta` to the character number of the top-right quarter circle with the largest diameter less than or equal to DIAM.
Sets `\@tempboxa` to an hbox containing that character.
Sets `\@tempdima` to `\wd \@tempboxa`, which is the distance from the circle's left outside edge to its right inside edge.
(These characters are like those described in the TeXbook, pp. 389-90.)

```
\@getcirc {DIAM} ==
  BEGIN
    \@tempcnta       := integer coercion of (DIAM + 2pt)
                                          + 2pt added 1 Nov 88
    \@tempcnta       := \@tempcnta / integer coercion of 4pt
    if \@tempcnta > 10
      then \@tempcnta := 10 fi
    if \@tempcnta > 0
      then \@tempcnta := \@tempcnta-1
      else LaTeX Warning: Oval too small.
    fi
    \@tempcnta       := 4 * \@tempcnta
    \@tempboxa       := \hbox{\@circlefnt \char \@tempcnta}
    \@tempdima       := \wd \@tempboxa
  END

\@put{X}{Y}{OBJ} ==
  BEGIN
    \raise Y \hb@xt@ 0pt{\hskip X OBJ \hss}
  END

\@oval(X,Y)[POS] ==
  BEGIN
    \begingroup
      \boxmaxdepth := \maxdimen
      @ovt := @ovb := @ovl := @ovr := true
      for all E in POS
        do  @ovE := false od
      \@ovxx      := X * \unitlength
      \@ovyy      := Y * \unitlength
```

```
      \@tempdimb := min(\@ovxx,\@ovyy)
      \@getcirc{\@tempdimb-2pt}   %% "-2pt" added 7 Dec 89
      \@ovro     := \ht \@tempboxa
      \@ovri     := \dp \@tempboxa
      \@ovdx     := \@ovxx - \@tempdima
      \@ovdx     := \@ovdx/2
      \@ovdy     := \@ovyy - \@tempdima
      \@ovdy     := \@ovyy/2
      \@circlefnt
      \@tempboxa :=
          \hbox{
                if @ovr
                  then \@ovvert{3}{2} \kern -\@tempdima
                fi
                if @ovl
                  then \kern \@ovxx \@ovvert{0}{1} \kern -\@tempdima
                       \kern -\@ovxx
                fi
                if @ovt
                  then \@ovhorz \kern -\@ovxx
                fi
                if @ovb
                  then \raise \@ovyy \@ovhorz
                fi
               }
      \@ovdx    := \@ovdx + \@ovro
      \@ovdy    := \@ovdy + \@ovro
     \ht\@tempboxa := \dp\@tempboxa := 0
      \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}
   \endgroup
  END

\@ovvert {DELTA1} {DELTA2} ==
  BEGIN
     \vbox to \@ovyy {
                      if @ovb
                        then \@tempcntb := \@tempcnta + DELTA1
                             \kern -\@ovro
                             \hbox { \char \@tempcntb }
                             \nointerlineskip
                        else \kern \@ovri \kern \@ovdy
                      fi
                      \leaders \vrule width \@wholewidth \vfil
                      \nointerlineskip
                      if @ovt
                        then \@tempcntb := \@tempcnta + DELTA2
                             \hbox { \char \@tempcntb }
                        else \kern \@ovdy \kern \@ovro
                      fi
                     }
```

```
  END

\@ovhorz ==
  BEGIN
   \hb@xt@ \@ovxx{
                  \kern \@ovro
                  if @ovr
                    then
                    else \kern \@ovdx
                  fi
                  \leaders \hrule height \@wholewidth \hfil
                  if @ovl
                    then
                    else \kern \@ovdx
                  fi
                  \kern \@ovri
                 }
  END

\circle{DIAM} ==
  BEGIN
   \begingroup
   \boxmaxdepth := maxdimen
   \@tempdimb := DIAM *\unitlength
   if \@tempdimb > 15.5pt
     then \@getcirc{\@tempdimb}
          \@ovro := \ht \@tempboxa
          \@tempboxa := \hbox{
                  \@circlefnt
                  \@tempcnta := \@tempcnta + 2
                  \char \@tempcnta
                  \@tempcnta := \@tempcnta - 1
                  \char \@tempcnta
                  \kern -2\@tempdima
                  \@tempcnta := \@tempcnta + 2
                  \raise \@tempdima \hbox { \char \@tempcnta }
                  \raise \@tempdima \box\@tempboxa
                 }
          \ht\@tempboxa := \dp\@tempboxa := 0
          \@put{-\@ovro}{-\@ovro}{\@tempboxa}
     else
          \@circ{\@tempdimb}{96}
   fi
  \endgroup
  END

\circle*{DIAM}  ==  \@dot{DIAM} == \@circ{DIAM*\unitlength}{112}

\@circ{DIAM}{CHAR} ==
 BEGIN
```

```
\@tempcnta := integer coercion of (DIAM + .5pt)/1pt.
if \@tempcnta > 15 then \@tempcnta := 15 fi
if \@tempcnta > 1  then \@tempcnta := \@tempcnta - 1 fi
\@tempcnta := \@tempcnta + CHAR
\@circlefnt
\char \@tempcnta
END
```

End of historical LaTeX 2.09 comments.

`\if@ovt` `\if@ovb` `\if@ovl` `\if@ovr`

If producing the Top Bottom Left or Right of an oval.

```
\newif\if@ovt
\newif\if@ovb
\newif\if@ovl
\newif\if@ovr
```

(*End of definition for* `\if@ovt` *and others.*)

`\@ovxx` `\@ovyy` `\@ovdx` `\@ovdy` `\@ovro` `\@ovri`

```
\newdimen\@ovxx
\newdimen\@ovyy
\newdimen\@ovdx
\newdimen\@ovdy
\newdimen\@ovro
\newdimen\@ovri
```

(*End of definition for* `\@ovxx` *and others.*)

`\advance\@tempdima 2pt\relax` added 1 Nov 88 to fix bug in which size of drawn circle not monotonic function of argument of `\circle`, caused by different rounding for dimensions of large and small circles.

`\@getcirc`

```
\def\@getcirc#1{\@tempdima #1\relax \advance\@tempdima 2\p@
  \@tempcnta\@tempdima
  \@tempdima 4\p@ \divide\@tempcnta\@tempdima
  \ifnum \@tempcnta >10\relax
      \@picture@warn
      \@tempcnta 10\relax
  \fi
  \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne
```

Warn if requirements for oval or circle can't be met.

```
    \else \@picture@warn \fi
  \multiply\@tempcnta 4\relax
  \setbox \@tempboxa \hbox{\@circlefnt
  \char \@tempcnta}\@tempdima \wd \@tempboxa}
```

(*End of definition for* `\@getcirc`*.*)

`\@picture@warn`

Generic warning for lines, vectors (used in `\@sline`) and oval or circle (used in `\@getcirc`) are not available at right size.

```
\def\@picture@warn{\@latex@warning{%
     \string\oval, \string\circle, or \string\line\space
     size unavailable}}
```

(*End of definition for* `\@picture@warn`.)

`\@put`

```
\def\@put#1#2#3{\raise #2\hb@xt@\z@{\hskip #1#3\hss}}
```

(*End of definition for* `\@put`.)

`\oval`

```
\def\oval(#1,#2){\@ifnextchar[{\@oval(#1,#2)}{\@oval(#1,#2)[]}}
```

(*End of definition for* `\oval`.)

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2016/03/31}%
⟨latexrelease⟩                  {\@ovhlinetrue}%
⟨latexrelease⟩                  {Avoid almost zero length leaders}%
⟨*2ekernel | latexrelease⟩
```

`\if@ovvline` `\if@ovhline` Tests whether horizontal or vertical lines are needed.

```
\newif\if@ovvline \@ovvlinetrue
\newif\if@ovhline \@ovhlinetrue
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@ovhlinetrue}%
⟨latexrelease⟩                  {Avoid almost zero length leaders}%
⟨latexrelease⟩\let\if@ovvline\@undefined
⟨latexrelease⟩\let\if@ovhline\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\if@ovvline` *and* `\if@ovhline`.)

`\@oval`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@oval}{default units}%
\def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
  \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue

  \@ovvlinefalse \@ovhlinefalse

  \@tfor\reserved@a :=#3\do{%
    \csname @ov\reserved@a false\endcsname}%
  \@defaultunitsset\@ovxx{#1}\unitlength
  \@defaultunitsset\@ovyy{#2}\unitlength

  \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx \@ovvlinetrue
  \else \@ovyy \ifdim \@ovyy =\@ovxx \else \@ovhlinetrue \fi\fi

  \advance \@tempdimb -2\p@
  \@getcirc \@tempdimb
  \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
  \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
  \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
```

```
  \ifdim \@ovdx >\z@ \@ovhlinetrue \fi
  \ifdim \@ovdy >\z@ \@ovvlinetrue \fi

  \@circlefnt \setbox\@tempboxa
  \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
  \if@ovl \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx \fi
  \if@ovt \@ovhorz \kern -\@ovxx \fi
  \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
  \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
  \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
  \endgroup}
⟨/2ekernel | latexrelease⟩

⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2016/03/31}%
⟨latexrelease⟩                  {\@oval}{default units}%
⟨latexrelease⟩\def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
⟨latexrelease⟩  \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue
⟨latexrelease⟩  \@ovvlinefalse \@ovhlinefalse
⟨latexrelease⟩  \@tfor\reserved@a :=#3\do{%
⟨latexrelease⟩    \csname @ov\reserved@a false\endcsname}%
⟨latexrelease⟩  \@ovxx #1\unitlength
⟨latexrelease⟩  \@ovyy #2\unitlength
⟨latexrelease⟩  \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx \@ovvlinetrue
⟨latexrelease⟩  \else \@ovyy \ifdim \@ovyy =\@ovxx \else \@ovhlinetrue
⟨latexrelease⟩   \fi\fi
⟨latexrelease⟩  \advance \@tempdimb -2\p@
⟨latexrelease⟩  \@getcirc \@tempdimb
⟨latexrelease⟩  \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
⟨latexrelease⟩  \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
⟨latexrelease⟩  \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
⟨latexrelease⟩  \ifdim \@ovdx >\z@ \@ovhlinetrue \fi
⟨latexrelease⟩  \ifdim \@ovdy >\z@ \@ovvlinetrue \fi
⟨latexrelease⟩  \@circlefnt \setbox\@tempboxa
⟨latexrelease⟩  \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
⟨latexrelease⟩  \if@ovl
⟨latexrelease⟩   \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@ovt \@ovhorz \kern -\@ovxx \fi
⟨latexrelease⟩  \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
⟨latexrelease⟩  \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
⟨latexrelease⟩  \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
⟨latexrelease⟩  \endgroup}
⟨latexrelease⟩\EndIncludeInRelease

⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@oval}{default units}%
⟨latexrelease⟩\def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
⟨latexrelease⟩  \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue
⟨latexrelease⟩  \@tfor\reserved@a :=#3\do
⟨latexrelease⟩                 {\csname @ov\reserved@a false\endcsname}%
⟨latexrelease⟩  \@ovxx #1\unitlength
⟨latexrelease⟩  \@ovyy #2\unitlength
⟨latexrelease⟩  \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx\else \@ovyy \fi
⟨latexrelease⟩  \advance \@tempdimb -2\p@
⟨latexrelease⟩  \@getcirc \@tempdimb
```

```
⟨latexrelease⟩  \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
⟨latexrelease⟩  \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
⟨latexrelease⟩  \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
⟨latexrelease⟩  \@circlefnt \setbox\@tempboxa
⟨latexrelease⟩  \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
⟨latexrelease⟩  \if@ovl
⟨latexrelease⟩   \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@ovt \@ovhorz \kern -\@ovxx \fi
⟨latexrelease⟩  \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
⟨latexrelease⟩  \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
⟨latexrelease⟩  \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
⟨latexrelease⟩  \endgroup}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@oval`.)

`\@ovvert`

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2016/03/31}%
⟨latexrelease⟩                 {\@ovvert}{Avoid almost zero length leaders}%
⟨*2ekernel | latexrelease⟩
\def\@ovvert#1#2{\vbox to\@ovyy{%
    \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
      \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
    \else \kern \@ovri \kern \@ovdy \fi
    \if@ovvline \leaders\vrule \@width \@wholewidth \fi
    \vfil \nointerlineskip
    \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
      \hbox{\char \@tempcntb}%
    \else \kern \@ovdy \kern \@ovro \fi}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@ovvert}{Avoid almost zero length leaders}%
⟨latexrelease⟩\def\@ovvert#1#2{\vbox to\@ovyy{%
⟨latexrelease⟩    \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
⟨latexrelease⟩      \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
⟨latexrelease⟩    \else \kern \@ovri \kern \@ovdy \fi
⟨latexrelease⟩    \leaders\vrule \@width \@wholewidth\vfil \nointerlineskip
⟨latexrelease⟩    \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
⟨latexrelease⟩      \hbox{\char \@tempcntb}%
⟨latexrelease⟩    \else \kern \@ovdy \kern \@ovro \fi}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@ovvert`.)

`\@ovhorz`

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2016/03/31}%
⟨latexrelease⟩                 {\@ovhorz}{Avoid almost zero length leaders}%
```

```
⟨*2ekernel | latexrelease⟩
\def\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
    \if@ovr \else \kern \@ovdx \fi
    \if@ovhline \leaders \hrule \@height \@wholewidth \fi
    \hfil
    \if@ovl \else \kern \@ovdx \fi
    \kern \@ovri}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@ovhorz}{Avoid almost zero length leaders}%
⟨latexrelease⟩\def\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
⟨latexrelease⟩    \if@ovr \else \kern \@ovdx \fi
⟨latexrelease⟩    \leaders \hrule \@height \@wholewidth \hfil
⟨latexrelease⟩    \if@ovl \else \kern \@ovdx \fi
⟨latexrelease⟩    \kern \@ovri}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@ovhorz`.)

`\circle`

```
\def\circle{\@inmatherr\circle\@ifstar\@dot\@circle}
```

(*End of definition for* `\circle`.)

`\@circle`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                 {\@circle}{default units}%
\def\@circle#1{%
  \begingroup \boxmaxdepth \maxdimen
   \@defaultunitsset\@tempdimb{#1}\unitlength
   \ifdim \@tempdimb >15.5\p@ \@getcirc\@tempdimb
      \@ovro\ht\@tempboxa
     \setbox\@tempboxa\hbox{\@circlefnt
      \advance\@tempcnta\tw@ \char \@tempcnta
      \advance\@tempcnta\m@ne \char \@tempcnta \kern -2\@tempdima
      \advance\@tempcnta\tw@
      \raise \@tempdima \hbox{\char\@tempcnta}\raise \@tempdima
        \box\@tempboxa}\ht\@tempboxa\z@ \dp\@tempboxa\z@
      \@put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
   \else  \@circ\@tempdimb{96}\fi\endgroup}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@circle}{default units}%
⟨latexrelease⟩\def\@circle#1{%
⟨latexrelease⟩  \begingroup \boxmaxdepth \maxdimen \@tempdimb #1\unitlength
⟨latexrelease⟩   \ifdim \@tempdimb >15.5\p@ \@getcirc\@tempdimb
⟨latexrelease⟩      \@ovro\ht\@tempboxa
```

```
⟨latexrelease⟩      \setbox\@tempboxa\hbox{\@circlefnt
⟨latexrelease⟩       \advance\@tempcnta\tw@ \char \@tempcnta
⟨latexrelease⟩       \advance\@tempcnta\m@ne \char \@tempcnta
⟨latexrelease⟩       \kern -2\@tempdima
⟨latexrelease⟩       \advance\@tempcnta\tw@
⟨latexrelease⟩       \raise \@tempdima \hbox{\char\@tempcnta}%
⟨latexrelease⟩       \raise \@tempdima
⟨latexrelease⟩         \box\@tempboxa}\ht\@tempboxa\z@ \dp\@tempboxa\z@
⟨latexrelease⟩       \@put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
⟨latexrelease⟩   \else  \@circ\@tempdimb{96}\fi\endgroup}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@circle`.)

`\@dot` Internal form of `\circle*`.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                 {\@dot}{default units}%
\def\@dot#1{%
  \@defaultunitsset\@tempdimb{#1}\unitlength
  \@circ\@tempdimb{112}}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@dot}{default units}%
⟨latexrelease⟩\def\@dot#1{\@tempdimb #1\unitlength \@circ\@tempdimb{112}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@dot`.)

`\@circ`

```
\def\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
   \@tempcnta\@tempdima \@tempdima \p@
   \divide\@tempcnta\@tempdima
   \ifnum\@tempcnta >15\relax \@tempcnta 15\relax \fi
   \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne\fi
   \advance\@tempcnta #2\relax
   \@circlefnt \char\@tempcnta}
```

(*End of definition for* `\@circ`.)

`\@xarg` `\@yarg` `\@yyarg` Counters used for manipulating the 'slope' arguments.

```
\newcount\@xarg
\newcount\@yarg
\newcount\@yyarg
```

(*End of definition for* `\@xarg`, `\@yarg`, *and* `\@yyarg`.)

`\@multicnt` Counter used in `\multiput`, and also `\multicolumn`.

```
\newcount\@multicnt
```

(*End of definition for* `\@multicnt`.)

\@xdim
\@ydim

Length registers.

```
\newdimen\@xdim
\newdimen\@ydim
```

(*End of definition for* `\@xdim` *and* `\@ydim`.)

\@linechar

Box for holding a line segment character, for sloping lines.

```
\newbox\@linechar
```

(*End of definition for* `\@linechar`.)

\@linelen

Length of the line currently being built.

```
\newdimen\@linelen
```

(*End of definition for* `\@linelen`.)

\@clnwd
\@clnht

Height and width of current line segment.

```
\newdimen\@clnwd
\newdimen\@clnht
```

(*End of definition for* `\@clnwd` *and* `\@clnht`.)

\@dashdim
\@dashbox
\@dashcnt

`\dashbox` internal registers.

```
\newdimen\@dashdim
\newbox\@dashbox
\newcount\@dashcnt
```

(*End of definition for* `\@dashdim`, `\@dashbox`, *and* `\@dashcnt`.)

Initialization: "`\thinlines`"

```
\let\@linefnt\tenln
\let\@circlefnt\tencirc
\@wholewidth\fontdimen8\tenln
\@halfwidth .5\@wholewidth
```

1.1 Curves

The new `\qbezier` command, based on the old `\bezier` defined in `bezier.sty`. *Historical LaTeX 2.09 comments (not necessarily accurate any more):*

```
\qbezier[N] == \bezier{N}

\bezier{N}(AX,AY)(BX,BY)(CX,CY) ==
  BEGIN
    IF N = 0
       THEN \@xdima := |BX - AX|
            \@xb := |CX - BX|
            \@xa := Max(\@xa, \@xb)
            \@ya := |BY - AY|
            \@yb := |CY - BY|
            \@ya := Max(\@ya, \@yb)
            @sc := Max(\@xa, \@ya)
            %% The coefficient .5 below is the degree of overlap of
            %% successive points, where 1 is no overlap and 0 is
```

```
        %% complete overlap.  A coefficient of C multiplies
        %% the number of points plotted by 1/C.
        %%
        \@xa := .5 * \@halfwidth
        @sc := @sc / \@halfwidth
        @sc := Max(@sc, qbeziermax)
    ELSE @sc := N
@scp := @sc+1
\@xb := 2 * (BX - AX) * \unitlength
\@xa := ((CX-AX)*\unitlength - \@xb)/@sc
\@yb := 2 * (BY - AY) * \unitlength
\@ya := ((CY-AY)*\unitlength - \@yb)/@sc
\@pictdot := square rule of width \@wholewidth
\count@ := 0
WHILE \count@ < @scp
  DO  \@xdim := ((\count@*\@xa + @xb) / @sc) * \count@
      \@ydim := ((\count@*\@ya + @yb) / @sc) * \count@
      plot pt with relative coords (\@xdim,\@ydim)
      \count@ := \count@+1
  OD
```

End of historical LaTeX 2.09 comments.

`\qbeziermax` The maximum number of points to plot.

```
\def\qbeziermax{500}
```

(*End of definition for* `\qbeziermax`.)

In the code below, to save registers `\@a` ... are not used. Instead other registers are reused.

```
\newcounter{@sc} -> \c@multicnt
\newcounter{@scp} -> \@tempcnta
\newdimen\@xa ->  \@ovxx
\newdimen\@xb ->  \@ovdx
\newdimen\@ya ->  \@ovyy
\newdimen\@yb ->  \@ovdy
\newsavebox{\@pictdot} -> \@tempboxa
```

`\qbezier` Main user-level command to plot quadratic bezier curves. `#2` should be `(`.

```
\newcommand\qbezier[2][0]{\bezier{#1}#2}
```

(*End of definition for* `\qbezier`.)

`\bezier` Form of `\bezier` compatible with 2.09 `bezier.sty`, but modified to ignore spaces between its arguments. `#2` should be white space, and `#4` should be `(`.

```
\def\bezier#1)#2(#3)#4({\@bezier#1)(#3)(}
```

\@bezier

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@bezier}{default units}%
\def\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
  \ifnum #1=\z@
      \@defaultunitsset\@ovxx{#4}\unitlength
        \@defaultunitsset{\advance\@ovxx}{-#2}\unitlength
        \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
      \@defaultunitsset\@ovdx{#6}\unitlength
        \@defaultunitsset{\advance\@ovdx}{-#4}\unitlength
        \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
        \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
      \@defaultunitsset\@ovyy{#5}\unitlength
        \@defaultunitsset{\advance\@ovyy}{-#3}\unitlength
        \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
      \@defaultunitsset\@ovdy{#7}\unitlength
        \@defaultunitsset{\advance\@ovdy}{-#5}\unitlength
        \ifdim \@ovdy<\z@  \@ovdy -\@ovdy \fi
        \ifdim \@ovyy<\@ovdy \@ovyy  \@ovdy \fi
      \@multicnt
         \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
      \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
      \ifnum \qbeziermax<\@multicnt
        \@multicnt\qbeziermax\relax
      \fi
  \else \@multicnt#1\relax \fi
  \@tempcnta\@multicnt \advance\@tempcnta\@ne
  \@defaultunitsset\@ovdx{#4}\unitlength
  \@defaultunitsset{\advance\@ovdx}{-#2}\unitlength
      \multiply\@ovdx \tw@
  \@defaultunitsset\@ovxx{#6}\unitlength
  \@defaultunitsset{\advance\@ovxx}{-#2}\unitlength
      \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
  \@defaultunitsset\@ovdy{#5}\unitlength
  \@defaultunitsset{\advance\@ovdy}{-#3}\unitlength
       \multiply\@ovdy \tw@
  \@defaultunitsset\@ovyy{#7}\unitlength
  \@defaultunitsset{\advance\@ovyy}{-#3}\unitlength
      \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt

  \setbox\@tempboxa\hbox{%
            \hskip -\@halfwidth
            \vrule \@height\@halfwidth
                   \@depth \@halfwidth
                   \@width \@wholewidth}%
   \put(#2,#3){%
     \count@\z@
     \@whilenum{\count@<\@tempcnta}\do
        {\@xdim\count@\@ovxx
           \advance\@xdim\@ovdx
           \divide\@xdim\@multicnt
           \multiply\@xdim\count@
```

```
          \@ydim\count@\@ovyy
             \advance\@ydim\@ovdy
             \divide\@ydim\@multicnt
             \multiply\@ydim\count@
          \raise \@ydim
             \hb@xt@\z@{\kern\@xdim
                          \unhcopy\@tempboxa\hss}%
          \advance\count@\@ne}}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@bezier}{default units}%
⟨latexrelease⟩\def\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
⟨latexrelease⟩  \ifnum #1=\z@
⟨latexrelease⟩      \@ovxx #4\unitlength
⟨latexrelease⟩        \advance\@ovxx -#2\unitlength
⟨latexrelease⟩        \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
⟨latexrelease⟩      \@ovdx #6\unitlength
⟨latexrelease⟩        \advance\@ovdx -#4\unitlength
⟨latexrelease⟩        \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
⟨latexrelease⟩        \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
⟨latexrelease⟩      \@ovyy #5\unitlength
⟨latexrelease⟩        \advance\@ovyy -#3\unitlength
⟨latexrelease⟩        \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
⟨latexrelease⟩      \@ovdy #7\unitlength
⟨latexrelease⟩        \advance\@ovdy -#5\unitlength
⟨latexrelease⟩        \ifdim \@ovdy<\z@  \@ovdy -\@ovdy \fi
⟨latexrelease⟩        \ifdim \@ovyy<\@ovdy \@ovyy  \@ovdy \fi
⟨latexrelease⟩      \@multicnt
⟨latexrelease⟩         \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
⟨latexrelease⟩      \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
⟨latexrelease⟩      \ifnum
⟨latexrelease⟩        \qbeziermax<\@multicnt \@multicnt\qbeziermax\relax
⟨latexrelease⟩      \fi
⟨latexrelease⟩  \else \@multicnt#1\relax \fi
⟨latexrelease⟩  \@tempcnta\@multicnt \advance\@tempcnta\@ne
⟨latexrelease⟩  \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
⟨latexrelease⟩      \multiply\@ovdx \tw@
⟨latexrelease⟩  \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
⟨latexrelease⟩      \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
⟨latexrelease⟩  \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength
⟨latexrelease⟩       \multiply\@ovdy \tw@
⟨latexrelease⟩  \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
⟨latexrelease⟩      \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt
⟨latexrelease⟩  \setbox\@tempboxa\hbox{%
⟨latexrelease⟩            \hskip -\@halfwidth
⟨latexrelease⟩            \vrule \@height\@halfwidth
⟨latexrelease⟩                   \@depth \@halfwidth
⟨latexrelease⟩                   \@width \@wholewidth}%
⟨latexrelease⟩   \put(#2,#3){%
⟨latexrelease⟩     \count@\z@
⟨latexrelease⟩     \@whilenum{\count@<\@tempcnta}\do
⟨latexrelease⟩        {\@xdim\count@\@ovxx
⟨latexrelease⟩           \advance\@xdim\@ovdx
```

```
⟨latexrelease⟩            \divide\@xdim\@multicnt
⟨latexrelease⟩            \multiply\@xdim\count@
⟨latexrelease⟩          \@ydim\count@\@ovyy
⟨latexrelease⟩             \advance\@ydim\@ovdy
⟨latexrelease⟩             \divide\@ydim\@multicnt
⟨latexrelease⟩             \multiply\@ydim\count@
⟨latexrelease⟩          \raise \@ydim
⟨latexrelease⟩             \hb@xt@\z@{\kern\@xdim
⟨latexrelease⟩                          \unhcopy\@tempboxa\hss}%
⟨latexrelease⟩          \advance\count@\@ne}}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\bezier` *and* `\@bezier`.)

As the commands above all use "picture" interface we couldn't define them with `\DeclareRobustCommand` so we do that now.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                 {\bezier}{Make commands robust}%
\MakeRobust\bezier
\MakeRobust\circle
\MakeRobust\dashbox
\MakeRobust\line
\MakeRobust\linethickness
\MakeRobust\multiput
\MakeRobust\oval
\MakeRobust\put
\MakeRobust\qbezier
\MakeRobust\shortstack
\MakeRobust\thinlines
\MakeRobust\vector
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\bezier}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\bezier
⟨latexrelease⟩\kernel@make@fragile\circle
⟨latexrelease⟩\kernel@make@fragile\dashbox
⟨latexrelease⟩\kernel@make@fragile\line
⟨latexrelease⟩\kernel@make@fragile\linethickness
⟨latexrelease⟩\kernel@make@fragile\multiput
⟨latexrelease⟩\kernel@make@fragile\oval
⟨latexrelease⟩\kernel@make@fragile\put
⟨latexrelease⟩\kernel@make@fragile\qbezier
⟨latexrelease⟩\kernel@make@fragile\shortstack
⟨latexrelease⟩\kernel@make@fragile\thinlines
⟨latexrelease⟩\kernel@make@fragile\vector
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
⟨/2ekernel⟩
```

File 43
ltthm.dtx

1 Theorem Environments

The user creates his own theorem-like environments with the command
`\newtheorem{`⟨*name*⟩`}{`⟨*text*⟩`}[`⟨*counter*⟩`]` or
`\newtheorem{`⟨*name*⟩`}[`⟨*oldname*⟩`]{`⟨*text*⟩`}`
This defines the environment ⟨*name*⟩ to be just as one would expect a theorem environment to be, except that it prints ⟨*text*⟩ instead of "Theorem".

If ⟨*oldname*⟩ is given, then environments ⟨*name*⟩ and ⟨*oldname*⟩ use the same counter, so using a ⟨*name*⟩ environment advances the number of the next ⟨*name*⟩ environment, and vice-versa.

If ⟨*counter*⟩ is given, then environment ⟨*name*⟩ is numbered within ⟨*counter*⟩.

E.g., if ⟨*counter*⟩ = `subsection`, then the first ⟨*name*⟩ in subsection 7.2 is numbered ⟨*text*⟩ 7.2.1.

The way ⟨*name*⟩ environments are numbered can be changed by redefining `\the`⟨*name*⟩.
Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
DOCUMENT STYLE PARAMETERS

\@thmcounter{COUNTER} : A command such that
              \edef\theCOUNTER{\@thmcounter{COUNTER}}
   defines \theCOUNTER to produce a number for a theorem environment.
   The default is:
          BEGIN \noexpand\arabic{COUNTER} END

\@thmcountersep : A separator placed between a theorem number and
        the number of the counter within which it is numbered.
        E.g., to make the third theorem of section 7.2 be numbered
        7.2-3, \@thmcountersep should be \def'ed to '-'.  Its
        default is '.'.

\@begintheorem{NAME}{NUMBER} : A command that begins a theorem
        environment for a 'theorem' named 'NAME NUMBER' –
        e.g., \@begintheorem{Lemma}{3.7} starts Lemma 3.7.

\@opargbegintheorem{NAME}{NUMBER}{OPARG} :
        A command that begins a theorem
        environment for a 'theorem' named 'NAME NUMBER' with optional
        argument OPARG – e.g., \@begintheorem{Lemma}{3.7}{Jones}
        starts 'Lemma 3.7 (Jones):'.

\@endtheorem : A command that ends a theorem environment.

\newtheorem{NAME}{TEXT}[COUNTER] ==
  BEGIN
    if \NAME is definable
```

```
        then \@definecounter{NAME}
             if COUNTER present
               then \@newctr{NAME}[COUNTER] fi
                    \theNAME ==  BEGIN \theCOUNTER \@thmcountersep
                                       eval\@thmcounter{NAME}      END
               else \theNAME ==  BEGIN eval\@thmcounter{NAME} END
             \NAME == \@thm{NAME}{TEXT}
             \endNAME == \@endtheorem
        else error
      fi
    END

  \newtheorem{NAME}[OLDNAME]{TEXT}==
    BEGIN
      if counter OLDNAME nonexistent
        then ERROR
        else
             if \NAME is definable
               then BEGIN
                    \theNAME == \theOLDNAME
                    \NAME == \@thm{OLDNAME}{TEXT}
                    \endNAME == \@endtheorem
                    END
               else error
             fi
      fi
    END

  \@thm{NAME}{TEXT} ==
    BEGIN
     \refstepcounter{NAME}
     if next char = [
        then \@ythm{NAME}{TEXT}
        else \@xthm{NAME}{TEXT}
     fi
    END

  \@xthm{NAME}{TEXT} ==
    BEGIN
     \@begintheorem{TEXT}{\theNAME}
     \ignorespaces
    END

  \@ythm{NAME}{TEXT}[OPARG] ==
    BEGIN
     \@opargbegintheorem{TEXT}{\theNAME}{OPARG}
     \ignorespaces
    END
```

End of historical LaTeX 2.09 comments.

\newtheorem

\newtheorem ought really be allowed only in the preamble Which would be good document style, and allow some main memory to be saved by declaring these commands to be \@onlypreamble. Unfortunately the LaTeX book indicates that \newtheorem may be used anywhere in the document...

```
⟨*2ekernel⟩
\def\newtheorem#1{%
  \@ifnextchar[{\@othm{#1}}{\@nthm{#1}}}
```

(*End of definition for* \newtheorem.)

\@nthm

```
\def\@nthm#1#2{%
  \@ifnextchar[{\@xnthm{#1}{#2}}{\@ynthm{#1}{#2}}}
```

(*End of definition for* \@nthm.)

\@xnthm

92/09/18 RmS: Changed \@addtoreset to \@newctr to produce error message if counter #3 does not exist (to be consistent with behaviour of \newcounter)

```
\def\@xnthm#1#2[#3]{%
  \expandafter\@ifdefinable\csname #1\endcsname
    {\@definecounter{#1}\@newctr{#1}[#3]%
     \expandafter\xdef\csname the#1\endcsname{%
       \expandafter\noexpand\csname the#3\endcsname \@thmcountersep
          \@thmcounter{#1}}%
     \global\@namedef{#1}{\@thm{#1}{#2}}%
     \global\@namedef{end#1}{\@endtheorem}}}
```

(*End of definition for* \@xnthm.)

\@ynthm

```
\def\@ynthm#1#2{%
  \expandafter\@ifdefinable\csname #1\endcsname
    {\@definecounter{#1}%
     \expandafter\xdef\csname the#1\endcsname{\@thmcounter{#1}}%
     \global\@namedef{#1}{\@thm{#1}{#2}}%
     \global\@namedef{end#1}{\@endtheorem}}}
```

(*End of definition for* \@ynthm.)

\@othm

```
\def\@othm#1[#2]#3{%
  \@ifundefined{c@#2}{\@nocounterr{#2}}%
    {\expandafter\@ifdefinable\csname #1\endcsname
    {\global\@namedef{the#1}{\@nameuse{the#2}}%
  \global\@namedef{#1}{\@thm{#2}{#3}}%
  \global\@namedef{end#1}{\@endtheorem}}}}
```

(*End of definition for* \@othm.)

\@thm

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/03/18}%
⟨latexrelease⟩                  {\@thm}{no link target}%
\def\@thm#1#2{%
```

```
  \@kernel@refstepcounter{#1}%
  \@ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@thm}{no link target}%
⟨latexrelease⟩\def\@thm#1#2{%
⟨latexrelease⟩   \refstepcounter{#1}%
⟨latexrelease⟩   \@ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
```

(*End of definition for* `\@thm`.)

`\@xthm`
`\@ythm`

```
⟨*2ekernel⟩
\def\@xthm#1#2{%
  \@begintheorem{#2}{\csname the#1\endcsname}\ignorespaces}
\def\@ythm#1#2[#3]{%
  \@opargbegintheorem{#2}{\csname the#1\endcsname}{#3}\ignorespaces}
```

(*End of definition for* `\@xthm` *and* `\@ythm`.)

Default values

`\@thmcounter`
`\@thmcountersep`

```
\def\@thmcounter#1{\noexpand\arabic{#1}}
\def\@thmcountersep{.}
```

(*End of definition for* `\@thmcounter` *and* `\@thmcountersep`.)

`\@begintheorem`
`\@opargbegintheorem`
`\@endtheorem`

Providing theorem defaults.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/03/18}%
⟨latexrelease⟩                 {\@begintheorem}{add link targets}%
\def\@begintheorem#1#2{\trivlist
   \item[\MakeLinkTarget{\@currentcounter}\hskip \labelsep{\bfseries #1\ #2}]\itshape}
\def\@opargbegintheorem#1#2#3{\trivlist
      \item[\MakeLinkTarget{\@currentcounter}\hskip \labelsep{\bfseries #1\ #2\ (#3)}]\itshap
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@begintheorem}{add link targets}%
⟨latexrelease⟩\def\@begintheorem#1#2{\trivlist
⟨latexrelease⟩   \item[\hskip \labelsep{\bfseries #1\ #2}]\itshape}
⟨latexrelease⟩\def\@opargbegintheorem#1#2#3{\trivlist
⟨latexrelease⟩   \item[\hskip \labelsep{\bfseries #1\ #2\ (#3)}]\itshape}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
\def\@endtheorem{\endtrivlist}
⟨/2ekernel⟩
```

(*End of definition for* `\@begintheorem`, `\@opargbegintheorem`, *and* `\@endtheorem`.)

File 44
ltsect.dtx

1 Sectioning Commands

This file defines the declarations such as `\author` which are used by `\maketitle`. `\maketitle` itself is defined by each class, not in the LaTeX kernel.

The second part of the file defines the generic commands used for defining sectioning commands such as `\chapter`. Again the actual document level commands are defined in the class files, in terms of these commands.

```
⟨*2ekernel⟩
\message{title,}
```

1.1 The Title

`\title` `\author` `\date` `\thanks` `\and` `\maketitle`

The user defines the title and author by the declarations `\title{`⟨*name*⟩`}`, `\author{`⟨*name*⟩`}`

Similarly the date is declared with `\date{`⟨*date*⟩`}`.

Inside these, the `\thanks{`⟨*footnote text*⟩`}` command may be used to make acknowledgements, notice of address, etc. in a footnote. If there are multiple authors, they have to be separated with the `\and` command.

And finally, the `\maketitle` command produces the actual title, using the information previously saved with the other commands.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\title}{Make commands robust}%
```

`\title` `\title` for use in `\maketitle`. If not given `\maketitle` will produce an error message.

```
\DeclareRobustCommand\title[1]{\gdef\@title{#1}}
```

(*End of definition for* `\title`. *This function is documented on page* 751.)

`\author` `\author` for use in `\maketitle`. If not given `\maketitle` will produce a warning message.

```
\DeclareRobustCommand*\author[1]{\gdef\@author{#1}}
```

(*End of definition for* `\author`.)

`\date` `\date` for use in `\maketitle`. If not given `\maketitle` will produce `\today` as the default.

```
\DeclareRobustCommand*\date[1]{\gdef\@date{#1}}
```

(*End of definition for* `\date`.)

`\thanks`

```
\DeclareRobustCommand\thanks[1]{\footnotemark
    \protected@xdef\@thanks{\@thanks
        \protect\footnotetext[\the\c@footnote]{#1}}%
}
```

(*End of definition for* `\thanks`.)

\and

```
\DeclareRobustCommand\and{%   % \begin{tabular}
  \end{tabular}%
  \hskip 1em \@plus.17fil%
  \begin{tabular}[t]{c}}%     % \end{tabular}
```

(*End of definition for* \and.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\title}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\title
⟨latexrelease⟩\kernel@make@fragile\author
⟨latexrelease⟩\kernel@make@fragile\date
⟨latexrelease⟩\kernel@make@fragile\thanks
⟨latexrelease⟩\kernel@make@fragile\and
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@title

```
\def\@title{\@latex@error{No \noexpand\title given}\@ehc}
```

(*End of definition for* \@title.)

\@author

```
\def\@author{\@latex@warning@no@line{No \noexpand\author given}}
```

(*End of definition for* \@author.)

\@date

```
\gdef\@date{\today}
```

(*End of definition for* \@date.)

\@thanks

```
\let\@thanks\@empty
```

(*End of definition for* \@thanks.)

```
\message{sectioning,}
```

1.2 Sectioning

\@secpenalty

```
\newcount\@secpenalty
\@secpenalty = -300
```

(*End of definition for* \@secpenalty.)

\if@noskipsec
\@noskipsectrue

Way back in 1991 (08/26) FMi & RmS set the \@noskipsec switch to true for the preamble and to false in \document. This was done to trap lists and related text in the preamble but it does not catch everything.

```
\newif\if@noskipsec \@noskipsectrue
```

(*End of definition for* `\if@noskipsec` *and* `\@noskipsectrue`.)

`\@startsection` The `\@startsection{`⟨*name*⟩`}{`⟨*level*⟩`}{ `⟨*indent*⟩`}{`⟨*beforeskip*⟩`}{`⟨*afterskip*⟩`}{`⟨*style*⟩`}*[`⟨*altheading*⟩` ]{`⟨*heading*⟩`}` command is the mother of all the user level sectioning commands. The part after the `*`, including the `*` is optional.

name: e.g., 'subsection'

level: a number, denoting depth of section – e.g., chapter = 0, section = 1, etc.

indent: Indentation of heading from left margin

beforeskip: Absolute value = skip to leave above the heading. If negative, then paragraph indent of text following heading is suppressed.

afterskip: if positive, then skip to leave below heading, else negative of skip to leave to right of run-in heading.

style: Commands to set style. Since June 1996 release the *last* command in this argument may be a command such as `\MakeUppercase` or `\fbox` that takes an argument. The section heading will be supplied as the argument to this command. So setting `#6` to, say, `\bfseries\MakeUppercase` would produce bold, uppercase headings.

If '`*`' is missing, then increment the counter. If it is present, then there should be no `[`⟨*altheading*⟩`]` argument. The command uses the counter 'secnumdepth'. It contains a pointer to the highest section level that is to be numbered.

Warning: The `\@startsection` command should be at the same or higher grouping level as the text that follows it. For example, you should *not* do something like

```
\def\foo{ \begingroup ...
            \paragraph{...}
          \endgroup}
```

Pseudocode for the `\@startsection` command *Historical LaTeX 2.09 comments (not necessarily accurate any more):*

```
 \@startsection
{NAME}{LEVEL}{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE} ==
    BEGIN
     IF  @noskipsec = T  THEN  \leavevmode  FI
                               % true if previous section had no body.

     \par
     \@tempskipa  := BEFORESKIP
     @afterindent := T
     IF \@tempskipa < 0  THEN  \@tempskipa  := -\@tempskipa
                               @afterindent := F
     FI
     IF @nobreak = true
       THEN \everypar == null
       ELSE \addpenalty{\@secpenalty}
            \addvspace{\@tempskipa}
     FI
     IF * next
```

```
        THEN \@ssect{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE}
        ELSE \@dblarg{\@sect
                 {NAME}{LEVEL}{INDENT}
                 {BEFORESKIP}{AFTERSKIP}{STYLE}}
     FI
 END
```

End of historical LaTeX 2.09 comments.

```
\def\@startsection#1#2#3#4#5#6{%
  \if@noskipsec \leavevmode \fi
  \par
  \@tempskipa #4\relax
  \@afterindenttrue
  \ifdim \@tempskipa <\z@
    \@tempskipa -\@tempskipa \@afterindentfalse
  \fi
  \if@nobreak
    \everypar{}%
  \else
    \addpenalty\@secpenalty\addvspace\@tempskipa
  \fi
  \@ifstar
    {\@ssect{#3}{#4}{#5}{#6}}%
    {\@dblarg{\@sect{#1}{#2}{#3}{#4}{#5}{#6}}}}
```

(*End of definition for* `\@startsection`.)

`\@sect` Pseudocode for the `\@sect` command *Historical LaTeX 2.09 comments (not necessarily accurate any more):*

```
 \@sect{NAME}{LEVEL}
        {INDENT}{BEFORESKIP}{AFTERSKIP}
        {STYLE}[ARG1]{ARG2}
            ==
   BEGIN
    IF LEVEL > \c@secnumdepth
      THEN \@svsec :=L null
      ELSE \refstepcounter{NAME}
           \@svsec :=L BEGIN \@seccntformat{#1}\relax END
    FI
    IF AFTERSKIP > 0
      THEN \begingroup
               STYLE
               \@hangfrom{\hskip INDENT\@svsec}
               {\interlinepenalty 10000 ARG2\par}
           \endgroup
           \NAMEmark{ARG1}
           \addcontentsline{toc}{NAME}
              { IF   LEVEL > \c@secnumdepth
                  ELSE \protect\numberline{\theNAME}  FI
                ARG1 }
      ELSE \@svsechd == BEGIN   STYLE
                                \hskip INDENT\@svsec
```

```
                                    ARG2
                                    \NAMEmark{ARG1}
                                    \addcontentsline{toc}{NAME}
                                       { IF   LEVEL > \c@secnumdepth
                                           ELSE
                                             \protect\numberline{\theNAME}
                                           FI
                                         ARG1 }
                            END
    FI
    \@xsect{AFTERSKIP}
 END
```

End of historical LaTeX 2.09 comments.

```
\def\@sect#1#2#3#4#5#6[#7]#8{%
  \ifnum #2>\c@secnumdepth
    \let\@svsec\@empty
  \else
    \refstepcounter{#1}%
```

Since `\@seccntformat` might end with an improper `\hskip` which is scanning forward for `plus` or `minus` we end the definition of `\@svsec` with `\relax` as a precaution.

```
    \protected@edef\@svsec{\@seccntformat{#1}\relax}%
  \fi
  \@tempskipa #5\relax
  \ifdim \@tempskipa>\z@
    \begingroup
```

This `{` used to be after the argument to `\@hangfrom` but was moved here to allow commands such as `\MakeUppercase` to be used at the end of `#6`.

```
      #6{%
        \@hangfrom{\hskip #3\relax\@svsec}%
          \interlinepenalty \@M #8\@@par}%
    \endgroup
    \csname #1mark\endcsname{#7}%
    \addcontentsline{toc}{#1}{%
      \ifnum #2>\c@secnumdepth \else
        \protect\numberline{\csname the#1\endcsname}%
      \fi
      #7}%
  \else
```

`\relax` added 2 May 90

```
    \def\@svsechd{%
      #6{\hskip #3\relax
      \@svsec #8}%
      \csname #1mark\endcsname{#7}%
      \addcontentsline{toc}{#1}{%
        \ifnum #2>\c@secnumdepth \else
          \protect\numberline{\csname the#1\endcsname}%
        \fi
        #7}}%
  \fi
  \@xsect{#5}}
```

(*End of definition for* `\@sect`.)

`\@xsect` Pseudocode for the `\@xsect` command *Historical LaTeX 2.09 comments (not necessarily accurate any more):*

```
\@xsect{AFTERSKIP} ==
 BEGIN
   IF AFTERSKIP > 0
     THEN \par \nobreak
          \vskip AFTERSKIP
          \@afterheading
     ELSE @nobreak :=G F
          @noskipsec :=G T
          \everypar{ IF @noskipsec = T
                       THEN @noskipsec :=G F
                            \clubpenalty := 10000 % local
                            \hskip -\parindent
                            \begingroup
                              \@svsechd
                            \endgroup
                            \unskip
                            \hskip -AFTERSKIP \relax
                                          %% relax added 14 Jan 91
                       ELSE \clubpenalty := \@clubpenalty % local
                            \everypar := NULL
                     FI
                   }
   FI

  END
```

End of historical LaTeX 2.09 comments.

```
\def\@xsect#1{%
  \@tempskipa #1\relax
  \ifdim \@tempskipa>\z@
```

Why not combine `\@sect` and `\@xsect` and save doing the same test twice? It is not possible to change this now as these have become hooks!

This `\par` seems unnecessary.

```
    \par \nobreak
    \vskip \@tempskipa
    \@afterheading
  \else

    \@nobreakfalse
    \global\@noskipsectrue
    \everypar{%
      \if@noskipsec
        \global\@noskipsecfalse
       {\setbox\z@\lastbox}%
        \clubpenalty\@M
        \begingroup \@svsechd \endgroup
        \unskip
        \@tempskipa #1\relax
```

```
        \hskip -\@tempskipa
      \else
        \clubpenalty \@clubpenalty
        \everypar{}%
      \fi}%
  \fi
  \ignorespaces}
```

(*End of definition for* `\@xsect`.)

`\@seccntformat` This command formats the section number including the space following it.

```
\def\@seccntformat#1{\csname the#1\endcsname\quad}
```

(*End of definition for* `\@seccntformat`.)

Pseudocode for the `\@ssect` command *Historical LaTeX 2.09 comments (not necessarily accurate any more):*

```
\@ssect{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE}{ARG} ==
  BEGIN
   IF AFTERSKIP > 0
     THEN \begingroup
            STYLE
            \@hangfrom{\hskip INDENT}
                      {\interlinepenalty 10000 ARG\par}
          \endgroup
     ELSE \@svsechd == BEGIN STYLE
                           \hskip INDENT
                           ARG
                       END
   FI
   \@xsect{AFTERSKIP}
  END
```

End of historical LaTeX 2.09 comments.

Pseudocode for the `\@afterheading` command *Historical LaTeX 2.09 comments (not necessarily accurate any more):*

```
\@afterheading ==
 BEGIN
   @nobreak :=G true
   \everypar := BEGIN IF @nobreak = T
                      THEN @nobreak  :=G false
                           \clubpenalty := 10000 % local
                           IF @afterindent = F
                             THEN remove \lastbox
                           FI
                      ELSE \clubpenalty := \@clubpenalty % local
                           \everypar := NULL
                   FI
              END
 END
```

End of historical LaTeX 2.09 comments.

\@ssect

```
\def\@ssect#1#2#3#4#5{%
  \@tempskipa #3\relax
  \ifdim \@tempskipa>\z@
    \begingroup
```

This { used to be after the argument to \@hangfrom but was moved here to allow commands such as \MakeUppercase to be used at the end of #4.

```
      #4{%
        \@hangfrom{\hskip #1}%
          \interlinepenalty \@M #5\@@par}%
    \endgroup
  \else
    \def\@svsechd{#4{\hskip #1\relax #5}}%
  \fi
  \@xsect{#3}}
```

(*End of definition for* \@ssect.)

\if@afterindent
\@afterindenttrue

```
\newif\if@afterindent \@afterindenttrue
```

(*End of definition for* \if@afterindent *and* \@afterindenttrue.)

\@afterheading

This hook is used in setting up custom-built headings in classes.dtx.

```
\def\@afterheading{%
  \@nobreaktrue
  \everypar{%
    \if@nobreak
      \@nobreakfalse
      \clubpenalty \@M
      \if@afterindent \else
        {\setbox\z@\lastbox}%
      \fi
    \else
      \clubpenalty \@clubpenalty
      \everypar{}%
    \fi}}
```

(*End of definition for* \@afterheading.)

\@hangfrom

\@hangfrom{⟨*text*⟩} : Puts ⟨*text*⟩ in a box, and makes a hanging indentation of the following material up to the first \par. Should be used in vertical mode.

```
\def\@hangfrom#1{\setbox\@tempboxa\hbox{{#1}}%
      \hangindent \wd\@tempboxa\noindent\box\@tempboxa}
```

(*End of definition for* \@hangfrom.)

\c@secnumdepth
\c@tocdepth

```
\newcount\c@secnumdepth
\newcount\c@tocdepth
```

(*End of definition for* \c@secnumdepth *and* \c@tocdepth.)

\secdef

\secdef{⟨*unstarcmds*⟩}{⟨*unstarcmds*⟩}{ ⟨*starcmds*⟩}

When defining a \chapter or \section command without using \@startsection, you can use \secdef as follows:

1. \def\chapter{ ...\secdef \⟨*starcmd*⟩ \⟨*unstarcmd*⟩ }
2. \def\⟨*starcmd*⟩[#1]#2{ ...} % Command to define \chapter[...]{...}
3. \def\⟨*unstarcmd*⟩#1{ ...} % Command to define \chapter*{...}

```
\def\secdef#1#2{\@ifstar{#2}{\@dblarg{#1}}}
```

(*End of definition for* \secdef.)

1.2.1 Initializations

\sectionmark
\subsectionmark
\subsubsectionmark
\paragraphmark
\subparagraphmark

```
\let\sectionmark\@gobble
\let\subsectionmark\@gobble
\let\subsubsectionmark\@gobble
\let\paragraphmark\@gobble
\let\subparagraphmark\@gobble
```

(*End of definition for* \sectionmark *and others.*)

```
\message{contents,}
```

1.3 Table of Contents etc.

1.3.1 Convention

\tf@⟨*foo*⟩ = file number for output for table foo. The file is opened only if @filesw = true.

1.3.2 Commands

A \l@⟨*type*⟩{⟨*entry*⟩}{⟨*page*⟩} Macro needs to defined by document style for making an entry of type ⟨*type*⟩ in a table of contents, etc. E.g., the document style should define \l@chapter, \l@section, etc.

Note: When the \protect command is used in the ⟨*entry*⟩ or ⟨*text*⟩ of one of the commands below, it causes the following control sequence to be written on the file without being expanded. The sequence will be expanded when the table of contents entry is processed.

Surprise: Inside an \addcontentsline or \addtocontents command argument, the commands: \index, \glossary, and \label are no-ops . This could cause a problem if the user puts an \index or \label into one of the commands he writes, or into the optional 'short version' argument of a \section or \caption command.

\@starttoc

The \@starttoc{⟨*ext*⟩} command is used to define the commands: \tableofcontents, \listoffigures, etc.

For example: \@starttoc{lof} is used in \listoffigures. This command reads the .⟨*ext*⟩ file and sets up to write the new .⟨*ext*⟩ file.

Historical LaTeX 2.09 comments (not necessarily accurate any more):

\@starttoc{EXT} ==

```
BEGIN
  \begingroup
     \makeatletter
     read file \jobname.EXT
     IF @filesw = true
       THEN  open \jobname.EXT as file \tf@EXT
     FI
     @nobreak :=G FALSE  %% added 24 May 89
  \endgroup
END
```

End of historical LaTeX 2.09 comments.

```
\def\@starttoc#1{%
  \begingroup
    \makeatletter
    \@input{\jobname.#1}%
    \if@filesw
      \expandafter\newwrite\csname tf@#1\endcsname
      \immediate\openout \csname tf@#1\endcsname \jobname.#1\relax
    \fi
    \@nobreakfalse
  \endgroup}
```

(*End of definition for* `\@starttoc`.)

`\addcontentsline` The `\addcontentsline{`⟨*table*⟩`}{`⟨*type*⟩`}{ `⟨*entry*⟩`}` command allows the user to add his/her own entry to a table of contents, etc. The command adds the entry `\contentsline{`⟨*type*⟩`}{`⟨*entry*⟩`}{`⟨*page*⟩`}{}` to the `.`⟨*table*⟩ file.

This macro is implemented as an application of `\addtocontents`. Note that `\thepage` is not expandable during `\protected@write` therefore one gets the page number at the time of the `\shipout`.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\addcontentsline}{fourth argument}%
\def\addcontentsline#1#2#3{%
```

We add an empty brace pair at the end of `\contentsline` so that the number of argument is identical in documents with and without hyperref.

```
  \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}{}%
                     \protected@file@percent}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                  {\addcontentsline}{Mask line endings}%
⟨latexrelease⟩ \def\addcontentsline#1#2#3{%
⟨latexrelease⟩  \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}%
```

We add `\protected@file@percent` at the end which is turned inside `\@writefile` into a percent character to mask the newline after the closing argument brace.

```
⟨latexrelease⟩             \protected@file@percent}}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\addcontentsline}{Mask line endings}%
```

```
⟨latexrelease⟩\def\addcontentsline#1#2#3{%
⟨latexrelease⟩   \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\addcontentsline`.)

`\addtocontents` The `\addtocontents{⟨table⟩}{⟨text⟩}` command adds ⟨*text*⟩ to the `.⟨table⟩` file, with no page number.

```
\long\def\addtocontents#1#2{%
  \protected@write\@auxout
      {\let\label\@gobble \let\index\@gobble \let\glossary\@gobble}%
      {\string\@writefile{#1}{#2}}}
```

(*End of definition for* `\addtocontents`.)

`\contentsline` The `\contentsline{⟨type⟩}{⟨entry⟩}{⟨page⟩}{}` macro produces a ⟨*type*⟩ entry in a table of contents, etc. It will appear in the `.toc` or other file. For example, the entry for subsection 1.4.3 in the table of contents, might be produced by:

```
\contentsline{subsection}
    {\numberline{1.4.3}Gnats and Gnus}{22}{}
```

The `\protect` command causes command sequences to be written without expanding them.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                  {\contentsline}{Four arguments}%
```

In the toc file `\contentsline` is followed by 4 arguments these days, but only the first 3 are used in the old interface. The fourth was by default empty and only used when hyperref was loaded. We now pick up all 4 arguments, save the last one away in `\@contentsline@destination` and then call the old interface. This is done to simplify the interface to hyperref and to prepare for future changes.

```
\def\contentsline#1#2#3#4{\gdef\@contentsline@destination{#4}%
                          \csname l@#1\endcsname{#2}{#3}}
```

Default definition.

```
\let\@contentsline@destination\@empty
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\contentsline}{Four arguments}%
⟨latexrelease⟩
⟨latexrelease⟩\def\contentsline#1{\csname l@#1\endcsname}
⟨latexrelease⟩\let\@contentsline@destination\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\contentsline`.)

`\@dottedtocline{⟨level⟩}{⟨indent⟩}{⟨numwidth⟩ }{⟨title⟩}{⟨page⟩}`: Macro to produce a table of contents line with the following parameters:

level If ⟨*level*⟩ > `\c@tocdepth`, then no line produced.

indent Total indentation from the left margin.

numwidth Width of box for number if the ⟨*title*⟩ has a `\numberline` command. As of 25 Jan 1988, this is also the amount of extra indentation added to second and later lines of a multiple line entry.

title Contents of entry.

page Page number.

Uses the following parameters, which must be set by the document style. They should be defined with `\def`'s.

`pnumwidth` Width of box in which page number is set.

`tocrmarg` Right margin indentation for all but last line of multiple-line entries.

`dotsep` Separation between dots, in mu units. Should be `\def`'d to a number like 2 or 1.7

`\@dottedtocline`

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2018/12/01}%
⟨latexrelease⟩                  {\@dottedtocline}{Prevent protrusion}%
\def\@dottedtocline#1#2#3#4#5{%
  \ifnum #1>\c@tocdepth \else
    \vskip \z@ \@plus.2\p@
    {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
     \parindent #2\relax\@afterindenttrue
     \interlinepenalty\@M
     \leavevmode
     \@tempdima #3\relax
```

If a document uses fonts other than computer modern, the use of a dot from math can be very disturbing despite the fact that this might be the only place in a document that then uses computer modern. Therefore we surround the dot with an `\hbox` to escape to the surrounding text font.

```
     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
     {#4}\nobreak
     \leaders\hbox{$\m@th
```

We finish off by preventing any protrusion if that is enabled. If protrusion happens the number may shift to the right and as a result you may end up with an additional dot in the toc line in some situations.

```
        \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
        mu$}\hfill
     \nobreak
     \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5%
                        \kern-\p@\kern\p@}%
     \par}%
  \fi}
```

(*End of definition for* `\@dottedtocline`.)

`\noprotrusion` This command, if placed directly to the right (or left) of a word, will prevent protrusion of that word into the margin. It is used in the toc entry lines as they shouldn't protrude. It is implemented as to kerns that cancel each other but being there hide the word so that protrusion is not added. Note that a zero kern or an empty box would not work as the protrusion mechanism will skip over those.

```
\DeclareRobustCommand\noprotrusion{\leavevmode\kern-\p@\kern\p@}
```

(*End of definition for* `\noprotrusion`.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@dottedtocline}{Prevent protrusion}%
⟨latexrelease⟩\def\@dottedtocline#1#2#3#4#5{%
⟨latexrelease⟩  \ifnum #1>\c@tocdepth \else
⟨latexrelease⟩    \vskip \z@ \@plus.2\p@
⟨latexrelease⟩    {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
⟨latexrelease⟩     \parindent #2\relax\@afterindenttrue
⟨latexrelease⟩     \interlinepenalty\@M
⟨latexrelease⟩     \leavevmode
⟨latexrelease⟩     \@tempdima #3\relax
⟨latexrelease⟩     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
⟨latexrelease⟩     {#4}\nobreak
⟨latexrelease⟩     \leaders\hbox{$\m@th
⟨latexrelease⟩        \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
⟨latexrelease⟩        mu$}\hfill
⟨latexrelease⟩     \nobreak
⟨latexrelease⟩     \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5}%
⟨latexrelease⟩     \par}%
⟨latexrelease⟩  \fi}
⟨latexrelease⟩
⟨latexrelease⟩\let\noprotrusion\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

Note: `\nobreak`'s added 7 Jan 86 to prevent bad line break that left the page number dangling by itself at left edge of a new line.

Changed 25 Jan 88 to use `\leftskip` instead of `\hangindent` so leaders of multiple-line contents entries would line up properly.

`\numberline` `\numberline{`⟨*number*⟩`}`: For use in a `\contentsline` command. It puts ⟨*number*⟩ flush-left in a box of width `\@tempdima` (Before 25 Jan 88 change, it also added `\@tempdima` to the hanging indentation.)

```
\def\numberline#1{\hb@xt@\@tempdima{#1\hfil}}
⟨/2ekernel⟩
```

(*End of definition for* `\numberline`.)

File 45
ltfloat.dtx

1 Floats

The different types of floats are identified by a ⟨*type*⟩ name, which is the name of the counter for that kind of float. For example, figures are of type 'figure' and tables are of type 'table'. Each ⟨*type*⟩ has associated a positive ⟨*type number*⟩, which is a power of two. E.g.,
figures might be have type number 1, tables type number 2, programs type number 4, etc.

The locations where a float can go are specified by a ⟨*placement specifier*⟩, which is a list of the possible locations, each denoted by a letter as follows:

h : here — at the current location in the text.
t : top — at the top of a text page.
b : bottom — at the bottom of a text page.
p : page — on a separate float page

In addition, in conjunction with these, you can use '!' which means that the current values of the float positioning parameters are ignored for this float. (Has no effect on 'p', float page positioning.) For example, 'pht' specifies that the float can appear in any of three locations: page, here or top.

1.1 Floating Environments

```
⟨*2ekernel⟩
\message{floats,}
```

Historical LaTeX 2.09 comments (not necessarily accurate any more):

Where floats may appear on a page, and how many may appear there are specified by the following float placement parameters. The numbers are named like counters so the user can set them with the ordinary counter-setting commands.

```
\c@topnumber       : Number of floats allowed at the top of a column.
\topfraction       : Fraction of column that can be devoted to floats.
\c@dbltopnumber, \dbltopfraction
                   : Same as above, but for double-column floats.
\c@bottomnumber, \bottomfraction
                   : Same as above for bottom of page.
\c@totalnumber     : Number of floats allowed in a single column,
                         including in-text floats.
\textfraction      :Minimum fraction of column that must contain text.
\floatpagefraction: Minimum fraction of page that must be taken
                         up by float page.
\dblfloatpagefraction
                   : Same as above, for double-column floats.
```

The document style must define the following.

```
\fps@TYPE    : The default placement specifier for floats of type
               TYPE.

\ftype@TYPE  : The type number for floats of type TYPE.

\ext@TYPE    : The file extension indicating the file on which the
               contents list for float type TYPE is stored.
                 For example,  \ext@figure = 'lof'.

\fnum@TYPE   : A macro to generate the figure number for a caption.
               For example, \fnum@TYPE == Figure \thefigure.

\@makecaption{NUM}{TEXT} :
            A macro to make a caption, with NUM the value
            produced by \fnum@... and TEXT the text of the caption.
            It can assume it's in a \parbox of the appropriate width.

\@float{TYPE}[PLACEMENT] : This macro begins a float environment for a
    single-column float of type TYPE with PLACEMENT as the placement
    specifier.  The default value of PLACEMENT is defined by
    \fps@TYPE.  The environment is ended by \end@float.
    E.g., \figure == \@float{figure}, \endfigure == \end@float.

 \@float{TYPE}[PLACEMENT] ==
  BEGIN
    if hmode then \@bsphack
                  \@floatpenalty := -10002
            else \@floatpenalty := -10003
    fi
    \@captype ==L TYPE
    \@dblflset
    \@fps     ==L PLACEMENT
    \@onelevel@sanitize \@fps
    add default PLACEMENT if at most ! in PLACEMENT ==
\@fpsadddefault
    if inner
      then LaTeX Error: 'Not in outer paragraph mode.'
           \@floatpenalty := 0
      else if \@freelist nonempty
            then \@currbox  :=L head of \@freelist
                 \@freelist :=G tail of \@freelist
                 \count\@currbox :=G 32*\ftype@TYPE +
                                       bits determined by PLACEMENT
            else \@floatpenalty := 0
                 LaTeX Error: 'Too many unprocessed floats'
           fi
    fi
    \@currbox :=G   \color@vbox
```

```
                    \normalcolor
                      \vbox{
                       %% 15 Dec 87 –
                       %% removed \boxmaxdepth :=L 0pt
                       %% that made box 0 depth because it screwed
                       %% things up. Instead, added \vskip0pt at end
                            \hsize = \columnwidth
                            \@parboxrestore
                            \@floatboxreset
  END

 \caption ==
   BEGIN
    \refstepcounter{\@captype}
    \@dblarg{\@caption{\@captype}}
   END
```

In following definition, `\par` moved from after `\addcontentsline` to before `\addcontentsline` because the `\write` could cause an extra blank line to be added to the paragraph above the caption. (Change made 12 Jun 87)

```
 \@caption{TYPE}[STEXT]{TEXT} ==
  BEGIN
    \par
    \addcontentsline{\ext@TYPE}{TYPE}{\numberline{\theTYPE}{STEXT}}
    \begingroup
      \@parboxrestore
      \@normalsize
      \@makecaption{\fnum@TYPE}{TEXT}
      \par
    \endgroup
  END
```

`\@dblfloat`{TYPE}[PLACEMENT] : Macro to begin a float environment for a double-column float of type TYPE with PLACEMENT as the placement specifier. The default value of PLACEMENT is 'tp'
The environment is ended by `\end@dblfloat`.
E.g., `\figure*` == `\@dblfloat`{figure},
`\endfigure*` == `\end@dblfloat`.

`\@dblfloat`{TYPE}[PLACEMENT] ==
Identical to `\@float`{TYPE}[PLACEMENT] except `\hsize` and `\linewidth` are set to `\textwidth`.

End of historical LaTeX 2.09 comments.

`\@floatpenalty`

```
3 \newcount\@floatpenalty
```

(*End of definition for* `\@floatpenalty`.)

\caption This is set to be an error message outside a float since no captype is defined there; this may need to be changed by some classes.

```
\def\caption{%
  \ifx\@captype\@undefined
    \@latex@error{\noexpand\caption outside float}\@ehd
    \expandafter\@gobble
  \else
    \refstepcounter\@captype
    \expandafter\@firstofone
  \fi
  {\@dblarg{\@caption\@captype}}%
}
```

(*End of definition for* \caption.)

\@caption

```
\long\def\@caption#1[#2]#3{%
  \par
  \addcontentsline{\csname ext@#1\endcsname}{#1}%
    {\protect\numberline{\csname the#1\endcsname}{\ignorespaces #2}}%
  \begingroup
```

The paragraph setting parameters are normalised at this point, however \@parboxrestore resets \everypar which is not correct in this context so \@setminipage is called if needed.

The float mechanism, like minipage, sets the flag @minipage true before executing the user-supplied text. Many LaTeX constructs test for this flag and do not add vertical space when it is true. The intention is that this emulates TeX's 'top of page' behaviour. The flag must be set false at the start of the first paragraph. This is achieved by a redefinition of \everypar, but the call to \@parboxrestore removes that redefinition, so it is re-inserted if needed. If the flag is already false then the \caption was not the first entry in the float, and so some other paragraph has already activated the special \everypar. In this case no further action is needed.

```
    \@parboxrestore
    \if@minipage
      \@setminipage
    \fi
    \normalsize
    \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
  \endgroup}
```

(*End of definition for* \@caption.)

\@float
\@dblflset

```
\def\@float#1{%
  \@ifnextchar[%
    {\@xfloat{#1}}%
    {\edef\reserved@a{\noexpand\@xfloat{#1}[\csname fps@#1\endcsname]}%
     \reserved@a}}
```

(*End of definition for* \@float *and* \@dblflset.)

\@dblfloat

```
\def\@dblfloat{%
  \if@twocolumn\let\reserved@a\@dbflt\else\let\reserved@a\@float\fi
  \reserved@a}
```

(*End of definition for* `\@dblfloat`.)

`\fps@dbl` Note that all double floats have default fps 'tp'.

(*End of definition for* `\fps@dbl`.)

`\@setfps` This sets the fps, dealing with error conditions by adding the default.

(*End of definition for* `\@setfps`.)

`\@xfloat` The first part of this sets the count register that stores all the information about the type and fps of the float.

We assume here that the default specifiers already contain no active characters.

It may be better to store the defaults as numbers, rather than symbol strings.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\@xfloat}{Check float options}%
⟨*2ekernel | latexrelease⟩
\def\@xfloat #1[#2]{%
  \@nodocument
  \def \@captype {#1}%
   \def \@fps {#2}%
   \@onelevel@sanitize \@fps
   \def \reserved@b {!}%
   \ifx \reserved@b \@fps
     \@fpsadddefault
   \else
     \ifx \@fps \@empty
       \@fpsadddefault
     \fi
   \fi
   \ifhmode
     \@bsphack
     \@floatpenalty -\@Mii
   \else
     \@floatpenalty-\@Miii
   \fi
  \ifinner
     \@parmoderr\@floatpenalty\z@
  \else
    \@next\@currbox\@freelist
      {%
       \@tempcnta \sixt@@n
       \expandafter \@tfor \expandafter \reserved@a
         \expandafter :\expandafter =\@fps
         \do
```

Start of changes, use a nested if structure, ending in an error.

```
          {%
           \if \reserved@a h%
             \ifodd \@tempcnta
             \else
               \advance \@tempcnta \@ne
             \fi
```

```
          \else\if \reserved@a t%
            \@setfpsbit \tw@
          \else\if \reserved@a b%
            \@setfpsbit 4%
          \else\if \reserved@a p%
            \@setfpsbit 8%
          \else\if \reserved@a !%
            \ifnum \@tempcnta>15
              \advance\@tempcnta -\sixt@@n\relax
            \fi
          \else
            \@latex@error{Unknown float option `\reserved@a'}%
            {Option `\reserved@a' ignored and `p' used.}%
            \@setfpsbit 8%
          \fi\fi\fi\fi\fi
          }%
```

End of changes

```
      \@tempcntb \csname ftype@\@captype \endcsname
      \multiply \@tempcntb \@xxxii
      \advance \@tempcnta \@tempcntb
      \global \count\@currbox \@tempcnta
      }%
    \@fltovf
  \fi
```

The remainder sets up the box in which the float is typeset, and the typesetting environment to be used. It is essential to have the extra box to avoid the unwanted space that would otherwise often be put at the top of the float.

It ends with a hook; not sure how useful this is but it is needed at present to deal with double-column floats.

```
  \global \setbox\@currbox
    \color@vbox
      \normalcolor
      \vbox \bgroup
        \hsize\columnwidth
        \@parboxrestore
        \@floatboxreset
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@xfloat}{Check float options}%
⟨latexrelease⟩\def\@xfloat #1[#2]{%
⟨latexrelease⟩  \@nodocument
⟨latexrelease⟩  \def \@captype {#1}%
⟨latexrelease⟩   \def \@fps {#2}%
⟨latexrelease⟩   \@onelevel@sanitize \@fps
⟨latexrelease⟩   \def \reserved@b {!}%
⟨latexrelease⟩   \ifx \reserved@b \@fps
⟨latexrelease⟩     \@fpsadddefault
⟨latexrelease⟩   \else
⟨latexrelease⟩     \ifx \@fps \@empty
⟨latexrelease⟩       \@fpsadddefault
```

```
⟨latexrelease⟩      \fi
⟨latexrelease⟩    \fi
⟨latexrelease⟩    \ifhmode
⟨latexrelease⟩      \@bsphack
⟨latexrelease⟩      \@floatpenalty -\@Mii
⟨latexrelease⟩    \else
⟨latexrelease⟩      \@floatpenalty-\@Miii
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \ifinner
⟨latexrelease⟩      \@parmoderr\@floatpenalty\z@
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@next\@currbox\@freelist
⟨latexrelease⟩      {%
⟨latexrelease⟩       \@tempcnta \sixt@@n
⟨latexrelease⟩       \expandafter \@tfor \expandafter \reserved@a
⟨latexrelease⟩         \expandafter :\expandafter =\@fps
⟨latexrelease⟩         \do
⟨latexrelease⟩          {%
⟨latexrelease⟩           \if \reserved@a h%
⟨latexrelease⟩             \ifodd \@tempcnta
⟨latexrelease⟩             \else
⟨latexrelease⟩               \advance \@tempcnta \@ne
⟨latexrelease⟩             \fi
⟨latexrelease⟩           \fi
⟨latexrelease⟩           \if \reserved@a t%
⟨latexrelease⟩             \@setfpsbit \tw@
⟨latexrelease⟩           \fi
⟨latexrelease⟩           \if \reserved@a b%
⟨latexrelease⟩             \@setfpsbit 4%
⟨latexrelease⟩           \fi
⟨latexrelease⟩           \if \reserved@a p%
⟨latexrelease⟩             \@setfpsbit 8%
⟨latexrelease⟩           \fi
⟨latexrelease⟩           \if \reserved@a !%
⟨latexrelease⟩             \ifnum \@tempcnta>15
⟨latexrelease⟩               \advance\@tempcnta -\sixt@@n\relax
⟨latexrelease⟩             \fi
⟨latexrelease⟩           \fi
⟨latexrelease⟩           }%
⟨latexrelease⟩       \@tempcntb \csname ftype@\@captype \endcsname
⟨latexrelease⟩       \multiply \@tempcntb \@xxxii
⟨latexrelease⟩       \advance \@tempcnta \@tempcntb
⟨latexrelease⟩       \global \count\@currbox \@tempcnta
⟨latexrelease⟩       }%
⟨latexrelease⟩    \@fltovf
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \global \setbox\@currbox
⟨latexrelease⟩    \color@vbox
⟨latexrelease⟩      \normalcolor
⟨latexrelease⟩      \vbox \bgroup
⟨latexrelease⟩        \hsize\columnwidth
⟨latexrelease⟩        \@parboxrestore
⟨latexrelease⟩        \@floatboxreset
⟨latexrelease⟩}%
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@xfloat`.)

`\@floatboxreset` The rational for allowing these normally global flags to be set locally here, via `\@parboxrestore`, was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within marginals or floats or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in `\@setnobreak`; otherwise this command will be redundant.

```
\def \@floatboxreset {%
        \reset@font
        \normalsize
        \@setminipage
}
```

(*End of definition for* `\@floatboxreset`.)

`\@setnobreak`

```
\def \@setnobreak{%
  \if@nobreak
    \let\outer@nobreak\@nobreaktrue
    \@nobreakfalse
  \fi
}
```

(*End of definition for* `\@setnobreak`.)

`\@setminipage`

```
\def \@setminipage{%
  \@minipagetrue
  \everypar{\@minipagefalse\everypar{}}%
}
```

(*End of definition for* `\@setminipage`.)

`\end@float`

```
\def\end@float{%
  \@endfloatbox
  \ifnum\@floatpenalty <\z@
```

We make sure that we never exceed `\textheight`, otherwise float will never get typeset (91/03/15 FMi).

```
    \@largefloatcheck
    \@cons\@currlist\@currbox
    \ifnum\@floatpenalty <-\@Mii
      \penalty -\@Miv
```

Saving and restoring `\prevdepth` added 26 May 87 to prevent extra vertical space when used in vertical mode.

```
      \@tempdima\prevdepth
      \vbox{}%
      \prevdepth\@tempdima
```

```
      \penalty\@floatpenalty
    \else
      \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
    \fi
  \fi
}
```

(*End of definition for* `\end@float`.)

`\end@dblfloat`

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\end@dblfloat}{float order in 2-column}%
⟨*2ekernel | latexrelease⟩
\def\end@dblfloat{%
  \if@twocolumn
    \@endfloatbox
    \ifnum\@floatpenalty <\z@
      \@largefloatcheck
```

Force the depth of two column float boxes.

```
      \global\dp\@currbox1sp %
```

What follows is essentially `\end@float` without a starting `\@endfloatbox`.

```
      \@cons\@currlist\@currbox
      \ifnum\@floatpenalty <-\@Mii
        \penalty -\@Miv
        \@tempdima\prevdepth
        \vbox{}%
        \prevdepth\@tempdima
        \penalty\@floatpenalty
      \else
        \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
      \fi
    \fi
  \else
    \end@float
  \fi
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\end@dblfloat}{float order in 2-column}%
⟨latexrelease⟩\def\end@dblfloat{%
⟨latexrelease⟩\if@twocolumn
⟨latexrelease⟩  \@endfloatbox
⟨latexrelease⟩  \ifnum\@floatpenalty <\z@
```

We make sure that we never exceed `\textheight`, otherwise float will never get typeset (91/03/15 FMi).

```
⟨latexrelease⟩    \@largefloatcheck
⟨latexrelease⟩    \@cons\@dbldeferlist\@currbox
⟨latexrelease⟩  \fi
```

RmS 92/03/18 changed \@esphack to \@Esphack.

```
⟨latexrelease⟩     \ifnum \@floatpenalty =-\@Mii \@Esphack\fi
⟨latexrelease⟩\else
⟨latexrelease⟩   \end@float
⟨latexrelease⟩\fi
⟨latexrelease⟩}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \end@dblfloat.)

\@endfloatbox This macro is not intended to be a hook; it is designed to help maintain the integrity of this code, which is used twice and, as can be seen, is subject to frequent changes.

```
\def \@endfloatbox{%
      \par\vskip\z@skip      %% \par\vskip\z@ added 15 Dec 87

      \@minipagefalse
      \outer@nobreak
    \egroup                  %% end of vbox
  \color@endbox
}
```

(*End of definition for* \@endfloatbox.)

\outer@nobreak

```
\let\outer@nobreak\@empty
```

(*End of definition for* \outer@nobreak.)

\@largefloatcheck This calculates by how much a float is oversize for the page and prints this in a warning message.

```
\def \@largefloatcheck{%
  \ifdim \ht\@currbox>\textheight
    \@tempdima -\textheight
    \advance \@tempdima \ht\@currbox

    \@latex@warning {Float too large for page by \the\@tempdima}%
    \ht\@currbox \textheight
  \fi
}
```

(*End of definition for* \@largefloatcheck.)

\@dbflt
\@xdblfloat

```
\def\@dbflt#1{\@ifnextchar[{\@xdblfloat{#1}}{\@xdblfloat{#1}[tp]}}
\def\@xdblfloat#1[#2]{%
  \@xfloat{#1}[#2]\hsize\textwidth\linewidth\textwidth}
```

(*End of definition for* \@dbflt *and* \@xdblfloat.)

Moved to ltoutput 93/12/16

```
%\newcount\c@topnumber
%\newcount\c@dbltopnumber
%\newcount\c@bottomnumber
%\newcount\c@totalnumber
```

`\@floatplacement` An analysis of `\@floatplacement`:

This should be called whenever `\@colht` has been set.

```
\def\@floatplacement{\global\@topnum\c@topnumber
    % Textpage bit, global:
   \global\@toproom \topfraction\@colht
   \global\@botnum  \c@bottomnumber
   \global\@botroom \bottomfraction\@colht
   \global\@colnum  \c@totalnumber
    % Floatpage bit, local:
   \@fpmin   \floatpagefraction\@colht}
⟨/2ekernel⟩
```

(*End of definition for* `\@floatplacement`.)

`\@dblfloatplacement` This should be called only within a group. Now changed to provide extra checks in `\@addtodblcol`, needed when processing a BANG float.

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩          {\@dblfloatplacement}{float order in 2-column}%
⟨*2ekernel | latexrelease⟩
```

When making two column float area, look for floats with 1sp depth.

```
\def\@dblfloatplacement{\global\@dbltopnum\c@dbltopnumber
   \global\@dbltoproom \dbltopfraction\@colht
   \@textmin \@colht
   \advance \@textmin -\@dbltoproom
   \@fpmin \dblfloatpagefraction\textheight
   \@fptop \@dblfptop
   \@fpsep \@dblfpsep
   \@fpbot \@dblfpbot
```

`\f@depth` is used in `\@testwrongwidth` to look for either column or dbl-column floats. A value of `1sp` signals the latter. Because of this setting here, `\@dblfloatplacment` needs to be called inside a group which is a questionable design.

```
   \def\f@depth{1sp}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩          {\@dblfloatplacement}{float order in 2-column}%
⟨latexrelease⟩\def \@dblfloatplacement {%
```

Textpage bit: global, but need not be.

```
⟨latexrelease⟩  \global \@dbltopnum \c@dbltopnumber
⟨latexrelease⟩  \global \@dbltoproom \dbltopfraction\@colht
```

This new bit uses `\@textmin` to locally store the amount of extra room in the column.

```
⟨latexrelease⟩  \@textmin \@colht
⟨latexrelease⟩  \advance \@textmin -\@dbltoproom
```

Floatpage bit: must be local.

```
⟨latexrelease⟩  \@fpmin \dblfloatpagefraction\textheight
⟨latexrelease⟩  \@fptop \@dblfptop
⟨latexrelease⟩  \@fpsep \@dblfpsep
⟨latexrelease⟩  \@fpbot \@dblfpbot
⟨latexrelease⟩}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@dblfloatplacement`.)

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
MARGINAL NOTES:

Marginal notes use the same mechanism as floats to communicate
with the \output routine.  Marginal notes are distinguished from
floats by having a negative placement specification.  The command
\marginpar [LTEXT]{RTEXT} generates a marginal note in a parbox,
using LTEXT if it's on the left and RTEXT if it's on the right.
(Default is RTEXT = LTEXT.)  It uses the following parameters.

\marginparwidth : Width of marginal notes.
\marginparsep   : Distance between marginal note and text.
      the page layout to determine how to move the marginal
      note into the margin.   E.g., \@leftmarginskip ==
      \hskip -\marginparwidth \hskip -\marginparsep .
\marginparpush  :  Minimum vertical separation between \marginpar's

Marginal notes are normally put on the outside of the page
if @mparswitch = true, and on the right if @mparswitch = false.
The command \reversemarginpar reverses the side where they
are put.  \normalmarginpar undoes \reversemarginpar.
These commands have no effect for two-column output.

SURPRISE: if two marginal notes appear on the same line of
text, then the second one could appear on the next page, in
a funny position.


\marginpar [LTEXT]{RTEXT} ==
 BEGIN
   if hmode then \@bsphack
                 \@floatpenalty := -10002
            else \@floatpenalty := -10003
   fi
   if inner
     then LaTeX Error: 'Not in outer paragraph mode.'
          \@floatpenalty := 0
     else if \@freelist has two elements:
            then get \@marbox, \@currbox  from \@freelist
                 \count\@marbox :=G -1
            else \@floatpenalty := 0
                 LaTeX Error: 'Too many unprocessed floats'
                 \@currbox, \@marbox := \@tempboxa    %%use \def
          fi
   fi
   if optional argument
     then %% \@xmpar ==
          \@savemarbox\@marbox{LTEXT}
          \@savemarbox\@currbox{RTEXT}
```

```
        else %% \@ympar ==
              \@savemarbox\@marbox{RTEXT}
              \box\@currbox :=G \box\@marbox
    fi
    \@xympar
   END

\reversemarginpar == BEGIN \@mparbottom    :=G 0
                           @reversemargin :=G true
                     END

\normalmarginpar  == BEGIN \@mparbottom    :=G 0
                           @reversemargin :=G false
                     END
```

End of historical LaTeX 2.09 comments.

`\marginpar`

```
\def\marginpar{%
  \ifhmode
    \@bsphack
    \@floatpenalty -\@Mii
  \else
    \@floatpenalty-\@Miii
  \fi
  \ifinner
    \@parmoderr
    \@floatpenalty\z@
  \else
    \@next\@currbox\@freelist{}{}%
    \@next\@marbox\@freelist{\global\count\@marbox\m@ne}%
       {\@floatpenalty\z@
        \@fltovf\def\@currbox{\@tempboxa}\def\@marbox{\@tempboxa}}%
  \fi
  \@ifnextchar [\@xmpar\@ympar}
```

(*End of definition for* `\marginpar`.)

`\@xmpar`

```
\long\def\@xmpar[#1]#2{%
  \@savemarbox\@marbox{#1}%
  \@savemarbox\@currbox{#2}%
  \@xympar}
```

(*End of definition for* `\@xmpar`.)

`\@ympar`

```
\long\def\@ympar#1{%
  \@savemarbox\@marbox{#1}%
  \global\setbox\@currbox\copy\@marbox
  \@xympar}
```

(*End of definition for* `\@ympar`.)

\@savemarbox

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                 {\@savemarbox}{Explicit par for marginpar}%
\long\def \@savemarbox #1#2{%
  \global\setbox #1%
    \color@vbox
      \vtop{%
        \hsize\marginparwidth
        \@parboxrestore
        \@marginparreset
        #2\par
        \@minipagefalse
        \outer@nobreak
        }%
    \color@endbox
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@savemarbox}{Explicit par for marginpar}%
⟨latexrelease⟩
⟨latexrelease⟩\long\def \@savemarbox #1#2{%
⟨latexrelease⟩  \global\setbox #1%
⟨latexrelease⟩    \color@vbox
⟨latexrelease⟩      \vtop{%
⟨latexrelease⟩        \hsize\marginparwidth
⟨latexrelease⟩        \@parboxrestore
⟨latexrelease⟩        \@marginparreset
⟨latexrelease⟩        #2%
⟨latexrelease⟩        \@minipagefalse
⟨latexrelease⟩        \outer@nobreak
⟨latexrelease⟩        }%
⟨latexrelease⟩    \color@endbox
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@savemarbox`.)

\@marginparreset The rational for allowing these normally global flags to be set locally here, via `\@parboxrestore` was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within marginals or floats or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in `\@setnobreak`; otherwise this command will be redundant.

```
\def \@marginparreset {%
        \reset@font
        \normalsize
%        \let\if@nobreak\iffalse
%        \let\if@noskipsec\iffalse
```

```
%         \@setnobreak
         \@setminipage
}
```

(*End of definition for* `\@marginparreset`.)

`\@xympar`

Setting the box here is done only because the code uses `\end@float`; it will be empty and gets discarded.

```
\def \@xympar{%
  \ifnum\@floatpenalty <\z@\@cons\@currlist\@marbox\fi
  \setbox\@tempboxa
    \color@vbox
      \vbox \bgroup
  \end@float
  \@ignorefalse
  \@esphack
}
```

(*End of definition for* `\@xympar`.)

`\reversemarginpar`
`\normalmarginpar`

```
\def\reversemarginpar{\global\@mparbottom\z@ \@reversemargintrue}
\def\normalmarginpar{\global\@mparbottom\z@ \@reversemarginfalse}
```

(*End of definition for* `\reversemarginpar` *and* `\normalmarginpar`.)

```
\message{footnotes,}
```

1.2 Footnotes

Historical LaTeX 2.09 comments (not necessarily accurate any more):

`\footnote{NOTE}` : User command to insert a footnote.

`\footnote[NUM]{NOTE}`: User command to insert a footnote numbered NUM, where NUM is a number – 1, 2, etc. For example, if footnotes are numbered *, **, etc. within pages, then `\footnote[2]{...}` produces footnote '**'. This command does not step the footnote counter.

`\footnotemark[NUM]` : Command to produce just the footnote mark in the text, but no footnote. With no argument, it steps the footnote counter before generating the mark.

`\footnotetext[NUM]{TEXT}` : Command to produce the footnote but no mark. `\footnote` is equivalent to `\footnotemark \footnotetext` .

As in PLAIN, footnotes use `\insert\footins,` and the following parameters:

`\footnotesize` : Size-changing command for footnotes.

`\footnotesep` : The height of a strut placed at the beginning of every footnote.

`\skip\footins` : Space between main text and footnotes. The rule separating footnotes from text occurs in this space. This space lies above the strut of height `\footnotesep` which is at the beginning of the first footnote.

`\footnoterule` : Macro to draw the rule separating footnotes from text. It is executed right after a `\vspace` of `\skip\footins`. It should take zero vertical space–i.e., it should to a negative skip to compensate for any positive space it occupies. (See PLAIN.TEX.)

`\interfootnotelinepenalty` : Interline penalty for footnotes.

`\thefootnote` : In usual LaTeX style, produces the footnote number. If footnotes are to be numbered within pages, then the document style file must include an `\@addtoreset` command to cause the footnote counter to be reset when the page counter is stepped. This is not a good idea, though, because the counter will not always be reset in time to ensure that the first footnote on a page is footnote number one.

`\@thefnmark` : Holds the current footnote's mark–e.g., `\dag` or '1' or 'a'.

`\@mpfnnumber` : A macro that generates the numbers for `\footnote` and `\footnotemark` commands. It == `\thefootnote` outside a minipage environment, but can be changed inside to generate numbers for `\footnote`'s.

`\@makefnmark` : A macro to generate the footnote marker from `\@thefnmark` The default definition was `\hbox{$^\@thefnmark$}`.

This is now replaced by `\textsuperscript{\@thefnmark}`

`\@makefntext{NOTE}` :

Must produce the actual footnote, using `\@thefnmark` as the mark of the footnote and NOTE as the text. It is called when effectively inside a `\parbox`, with `\hsize` = `\columnwidth`.

For example, it might be as simple as

`$^{\@thefnmark}$` NOTE

In a minipage environment, \footnote and \footnotetext are redefined so that

(a) they use the counter mpfootnote

(b) the footnotes they produce go at the bottom of the minipage.

The switch is accomplished by letting \@mpfn == footnote or mpfootnote and \thempfn == \thefootnote or \thempfootnote, and by redefining \@footnotetext to be \@mpfootnotetext in the minipage.

```
\footnote{NOTE}  ==
 BEGIN
   \stepcounter{\@mpfn}
   begingroup
      \protect == \noexpand
      \@thefnmark :=G eval (\thempfn)
   endgroup
   \@footnotemark
   \@footnotetext{NOTE}
 END

\footnote[NUM]{NOTE} ==
 BEGIN
   begingroup
      \protect == \noexpand
      counter \@mpfn :=L NUM
      \@thefnmark :=G eval (\thempfn)
   endgroup
   \@footnotemark
   \@footnotetext{NOTE}
 END

\footnotemark      ==
 BEGIN \stepcounter{footnote}
        begingroup
           \protect == \noexpand
           \@thefnmark :=G eval(\thefootnote)
        endgroup
        \@footnotemark
 END

\footnotemark[NUM] ==
  BEGIN
      begingroup
        footnote counter :=L NUM
        \protect == \noexpand
       \@thefnmark :=G eval(\thefootnote)
      endgroup
      \@footnotemark
  END
```

```
\@footnotemark ==
  BEGIN
   \leavevmode
   IF hmode THEN \@x@sf := \the\spacefactor FI
   \@makefnmark          % put number in main text
   IF hmode THEN \spacefactor := \@x@sf FI
  END

\footnotetext      ==
   BEGIN begingroup \protect == \noexpand
                    \@thefnmark :=G eval (\thempfn)
         endgroup
         \@footnotetext
   END

\footnotetext[NUM] ==
   BEGIN begingroup  counter \@mpfn :=L NUM
                     \protect == \noexpand
                     \@thefnmark :=G eval (\thempfn)
         endgroup
         \@footnotetext
   END
```

End of historical LaTeX 2.09 comments.

`\footins` LaTeX does use the same insert for footnotes as PLAIN.

```
\newinsert\footins
```

LaTeX leaves these initializations for the `\footins` insert.

```
\skip\footins=\bigskipamount % space added when footnote is present
\count\footins=1000 % footnote magnification factor (1 to 1)
\dimen\footins=8in % maximum footnotes per page
```

(*End of definition for* `\footins`.)

`\footnoterule` LaTeX keeps PLAIN TeX's `\footnoterule` as the default.

```
\def\footnoterule{\kern-3\p@
  \hrule \@width 2in \kern 2.6\p@} % the \hrule is .4pt high
```

(*End of definition for* `\footnoterule`.)

`\thefootnote`

```
\@definecounter{footnote}
\def\thefootnote{\@arabic\c@footnote}
```

(*End of definition for* `\thefootnote`.)

`\thempfootnote` The default display for the footnote counter in minipages is to use italic letters. We use `\itshape` not `\textit` as the latter would add an italic correction.

```
\@definecounter{mpfootnote}
\def\thempfootnote{{\itshape\@alph\c@mpfootnote}}
```

(*End of definition for* `\thempfootnote`.)

`\@makefnmark` Default definition.

```
%\def\@makefnmark{\hbox{$^{\@thefnmark}\m@th$}}
\def\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}
```

(*End of definition for* `\@makefnmark`.)

`\textsuperscript` This command provides superscript characters in the current text font. It's implementation might change!!!

```
\DeclareRobustCommand*\textsuperscript[1]{%
  \@textsuperscript{\selectfont#1}}
```

(*End of definition for* `\textsuperscript`.)

`\@textsuperscript` This command should not be used directly, but may be used to define other commands `\textsuperscript`, `\@makefnmark`. `#1` should always start with a font selection command, to activate the font size switch.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@textsuperscript}{superscript baseline}%
\def\@textsuperscript#1{%
  {\m@th\ensuremath{^{\mbox{\fontsize\sf@size\sf@size#1}}}}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@textsuperscript}{superscript baseline}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@textsuperscript#1{%
⟨latexrelease⟩  {\m@th\ensuremath{^{\mbox{\fontsize\sf@size\z@#1}}}}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@textsuperscript`.)

`\textsubscript`

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\textsubscript}{\textsubscript}%
⟨*2ekernel | latexrelease⟩
\DeclareRobustCommand*\textsubscript[1]{%
  \@textsubscript{\selectfont#1}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\textsubscript}{\textsubscript}%
⟨latexrelease⟩\let\textsubscript\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\textsubscript`.)

\@textsubscript

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@textsubscript}{subscript baseline}%
\def\@textsubscript#1{%
  {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\sf@size#1}}}}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\@textsubscript}{subscript baseline}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@textsubscript#1{%
⟨latexrelease⟩  {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\z@#1}}}}}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@textsubscript}{subscript baseline}%
⟨latexrelease⟩\let\@textsubscript\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \@textsubscript.)

\footnotesep

```
\newdimen\footnotesep
```

(*End of definition for* \footnotesep.)

\footnote

```
\def\footnote{\@ifnextchar[\@xfootnote{\stepcounter\@mpfn
      \protected@xdef\@thefnmark{\thempfn}%
      \@footnotemark\@footnotetext}}
```

(*End of definition for* \footnote.)

\@xfootnote

```
\def\@xfootnote[#1]{%
   \begingroup
     \csname c@\@mpfn\endcsname #1\relax
     \unrestored@protected@xdef\@thefnmark{\thempfn}%
   \endgroup
   \@footnotemark\@footnotetext}
```

(*End of definition for* \@xfootnote.)

\@footnotetext

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                  {\@footnotetext}{footnotetext tagging}%
\long\def\@footnotetext#1{\insert\footins{%
    \reset@font\footnotesize
    \interlinepenalty\interfootnotelinepenalty
    \splittopskip\footnotesep
```

```
    \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
    \hsize\columnwidth \@parboxrestore
    \def\@currentcounter{footnote}%
    \protected@edef\@currentlabel{%
       \csname p@footnote\endcsname\@thefnmark
    }%
    \color@begingroup
      \@makefntext{%
        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
    \par
    \color@endgroup}}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease

⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                 {\@footnotetext}{footnotetext tagging}%
⟨latexrelease⟩\long\def\@footnotetext#1{\insert\footins{%
⟨latexrelease⟩    \reset@font\footnotesize
⟨latexrelease⟩    \interlinepenalty\interfootnotelinepenalty
⟨latexrelease⟩    \splittopskip\footnotesep
⟨latexrelease⟩    \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
⟨latexrelease⟩    \hsize\columnwidth \@parboxrestore
⟨latexrelease⟩    \protected@edef\@currentlabel{%
⟨latexrelease⟩       \csname p@footnote\endcsname\@thefnmark
⟨latexrelease⟩    }%
⟨latexrelease⟩    \color@begingroup
⟨latexrelease⟩      \@makefntext{%
⟨latexrelease⟩        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
⟨latexrelease⟩    \par
⟨latexrelease⟩    \color@endgroup}}%
⟨latexrelease⟩\EndIncludeInRelease

⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@footnotetext}{footnotetext tagging}%
⟨latexrelease⟩
⟨latexrelease⟩\long\def\@footnotetext#1{\insert\footins{%
⟨latexrelease⟩    \reset@font\footnotesize
⟨latexrelease⟩    \interlinepenalty\interfootnotelinepenalty
⟨latexrelease⟩    \splittopskip\footnotesep
⟨latexrelease⟩    \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
⟨latexrelease⟩    \hsize\columnwidth \@parboxrestore
⟨latexrelease⟩    \protected@edef\@currentlabel{%
⟨latexrelease⟩       \csname p@footnote\endcsname\@thefnmark
⟨latexrelease⟩    }%
⟨latexrelease⟩    \color@begingroup
⟨latexrelease⟩      \@makefntext{%
⟨latexrelease⟩        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
⟨latexrelease⟩    \color@endgroup}}%
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@footnotetext`.)

`\footnotemark`

```
\def\footnotemark{%
   \@ifnextchar[\@xfootnotemark
     {\stepcounter{footnote}%
      \protected@xdef\@thefnmark{\thefootnote}%
      \@footnotemark}}
```

(*End of definition for* `\footnotemark`.)

`\@xfootnotemark`

```
\def\@xfootnotemark[#1]{%
   \begingroup
      \c@footnote #1\relax
      \unrestored@protected@xdef\@thefnmark{\thefootnote}%
   \endgroup
   \@footnotemark}
```

(*End of definition for* `\@xfootnotemark`.)

`\@footnotemark`

```
\def\@footnotemark{%
  \leavevmode
  \ifhmode\edef\@x@sf{\the\spacefactor}\nobreak\fi
  \@makefnmark
  \ifhmode\spacefactor\@x@sf\fi
  \relax}
```

(*End of definition for* `\@footnotemark`.)

`\footnotetext`

```
\def\footnotetext{%
     \@ifnextchar [\@xfootnotenext
       {\protected@xdef\@thefnmark{\thempfn}%
    \@footnotetext}}
```

(*End of definition for* `\footnotetext`.)

`\@xfootnotenext`

```
\def\@xfootnotenext[#1]{%
  \begingroup
     \csname c@\@mpfn\endcsname #1\relax
     \unrestored@protected@xdef\@thefnmark{\thempfn}%
  \endgroup
  \@footnotetext}
```

(*End of definition for* `\@xfootnotenext`.)

`\thempfn`
`\@mpfn`

```
\def\@mpfn{footnote}
\def\thempfn{\thefootnote}
```

(*End of definition for* `\thempfn` *and* `\@mpfn`.)

`\footref` This command generates a footnote mark. The value is produced by referencing a `\label` placed into a `\footnote` elsewhere (can be one in the main galley or in a minipage).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                  {\footref}{Add footref}%
\def\footref#1{%
  \begingroup
    \unrestored@protected@xdef\@thefnmark{\ref{#1}}%
  \endgroup
  \@footnotemark
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

We don't remove it when rolling back so that packages offered it in the past do not need to alter their behavior in a rollback situation.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\footref}{Add footref}%
⟨latexrelease⟩
⟨latexrelease⟩  % \let\footref\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\footref`.)

```
⟨/2ekernel⟩
```

File 46
ltidxglo.dtx

1 Index and Glossary Generation

Index and Glossary commands.

`\makeindex` A preamble command to turn on indexing.

`\makeglossary` A preamble command to turn on making glossary entries.

`\index` Make an index entry for `#1`.

`\glossary` Make a glossary entry for `#1`.

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
\makeindex ==
  BEGIN
            \index ==  BEGIN \@bsphack
                             \begingroup
                                \protect{X} == \string X\space
                                 %% added 3 Feb 87 for \index commands
                                 %% in \footnotes
                                 re-\catcode special characters
                                 to 'other'
                                 \@wrindex
  END

\@wrindex{ITEM} ==
  BEGIN
      write of {\indexentry{ITEM}{page number}}
    \endgroup
    \@esphack
  END

INITIALIZATION:

\index == BEGIN \@bsphack
                \begingroup
                   re-\catcode special characters (in case '%' there)
                   \@index
          END

\@index{ITEM} == BEGIN \endgroup \@esphack END
```

Changes made 14 Apr 89 to write `\glossaryentry`'s instead of `\indexentry`'s on the .glo file.

End of historical LaTeX 2.09 comments.

```
⟨*2ekernel⟩
\message{index,}
```

\makeindex

```
\def\makeindex{%
  \newwrite\@indexfile
  \immediate\openout\@indexfile=\jobname.idx
  \def\index{\@bsphack\begingroup
             \@sanitize
             \@wrindex}\typeout
    {Writing index file \jobname.idx}%
```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```
  \let\makeindex\@empty
}
\@onlypreamble\makeindex
```

(*End of definition for* `\makeindex`.)

\@wrindex

```
\def\@wrindex#1{%
   \protected@write\@indexfile{}%
      {\string\indexentry{#1}{\thepage}}%
 \endgroup
 \@esphack}
```

(*End of definition for* `\@wrindex`.)

\index

```
\def\index{\@bsphack\begingroup \@sanitize\@index}
```

(*End of definition for* `\index`.)

\@index

```
\def\@index#1{\endgroup\@esphack}
```

(*End of definition for* `\@index`.)

\makeglossary

```
\def\makeglossary{%
  \newwrite\@glossaryfile
  \immediate\openout\@glossaryfile=\jobname.glo
  \def\glossary{\@bsphack\begingroup
                \@sanitize
                \@wrglossary}\typeout
    {Writing glossary file \jobname.glo }%
```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```
  \let\makeglossary\@empty
}
\@onlypreamble\makeglossary
```

(*End of definition for* `\makeglossary`.)

\@wrglossary

```
\def\@wrglossary#1{%
   \protected@write\@glossaryfile{}%
      {\string\glossaryentry{#1}{\thepage}}%
 \endgroup
 \@esphack}
```

(*End of definition for* `\@wrglossary`.)

\glossary

```
\def\glossary{\@bsphack\begingroup\@sanitize\@index}
```

(*End of definition for* `\glossary`.)

⟨/2ekernel⟩

File 47
ltbibl.dtx

1 Bibliography Generation

A bibliography is created by the `thebibliography` environment, which generates a title such as "References", and a list of entries. The BIBTEX program will create a file containing such an environment, which will be read in by the `\bibliography` command. With BIBTEX, the following commands will be used.

`\bibliography` `\bibliography{`⟨*file1,file2, ...,filen*⟩`}` : specifies the bibdata files. Writes a `\bibdata` entry on the `.aux` file and tries to read in `mainfile.bbl`.

`\bibliographystyle` `\bibliographystyle{`⟨*style*⟩`}` : Writes a `\bibstyle` entry on the `.aux` file.

`thebibliography` (*env.*) The `thebibliography` environment is a list environment. To save the use of an extra counter, it should use `enumiv` as the item counter. Instead of using `\item`, items in the bibliography are produced by the following commands:

`\bibitem{`⟨*name*⟩`}` : Produces a numbered entry cited as ⟨*name*⟩.

`\bibitem[`⟨*label*⟩`]{`⟨*name*⟩`}` : Produces an entry labeled by ⟨*Label*⟩ and cited by ⟨*name*⟩.

The former is used for bibliographies with citations like [1], [2], etc.; the latter is used for citations like [Knuth82].

The document class must define the thebibliography environment. This environment has a single argument, which is the widest bibliography label– e.g., if the [Knuth67] is the widest entry, then this argument will be Knuth67. The `\thebibliography` command must begin a list environment, which the `\endthebibliography` command ends.

`\cite` Entries are cited by the command `\cite{`⟨*name*⟩`}`.

`\nocite` `\nocite{`⟨*citations*⟩`}` puts information on the `.aux` file that causes BIBTEX to include the `{`⟨*citations*⟩`}` list in the bibliography, but puts nothing in the text.

`\nocite{*}` is special: it tells BIBTEX to put the whole of a collection of references into the bibliography.

```
⟨*2ekernel⟩
\message{bibliography,}
```

Historical LATEX 2.09 comments (not necessarily accurate any more):

PARAMETERS

```
\@cite    : A macro such that \@cite{LABEL1,LABEL2}{NOTE}
            produces the output for a \cite[NOTE]{FOO1,FOO2} command,
            where entry FOOi is defined by \bibitem[LABELi]{FOOi}.
            The switch @tempswa is true if the optional NOTE argument
            is present.
            The default definition is :
              \@cite{LABELS}{NOTE} ==
                 BEGIN [LABELS
                        IF @tempswa = T THEN , NOTE FI
                        ]
                 END

\@biblabel : A macro to produce the label in the bibliography
             entry.  For \bibitem[LABEL]{NAME}, the label is
```

generated by \@biblabel{LABEL}. It has the default definition \@biblabel{LABEL} -> [LABEL].

CONVENTION

\b@FOO : The name or number of the reference created by \cite{FOO} E.g., if \cite{FOO} -> [17] , then \b@FOO -> 17.

End of historical LaTeX 2.09 comments.

\bibitem

```
\def\bibitem{\@ifnextchar[\@lbibitem\@bibitem}
```

(*End of definition for* \bibitem.)

\@lbibitem

```
\def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
      {\let\protect\noexpand
       \immediate
       \write\@auxout{\string\bibcite{#2}{#1}}}\fi\ignorespaces}
```

(*End of definition for* \@lbibitem.)

\@bibitem

```
\def\@bibitem#1{\item\if@filesw \immediate\write\@auxout
       {\string\bibcite{#1}{\the\value{\@listctr}}}\fi\ignorespaces}
```

(*End of definition for* \@bibitem.)

\bibcite

```
\def\bibcite{\@newl@bel b}
```

(*End of definition for* \bibcite.)

\citation

```
\let\citation\@gobble
```

(*End of definition for* \citation.)

\cite

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                  {\cite}{check for blank}%
\DeclareRobustCommand\cite{%
  \@ifnextchar [{\@tempswatrue\@citex@checkblank}{\@tempswafalse\@citex@checkblank[]}}
```

Due to the way \@for as used in \@citex behaves an empty argument to \cite did not produce any warning for a missing citation. So we now inject a command before calling \@citex that does the checking for us. It is not done in \@citex directly, because that command is altered by a number of packages/classes and this way it is more likely that the check survives.

```
\def\@citex@checkblank[#1]#2{%
   \IfBlankTF {#2}%
     {\@citex[#1]{\space}}%
     {\@citex[#1]{#2}}%
}
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\cite}{check for blank}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\cite{%
⟨latexrelease⟩  \@ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex[]}}
⟨latexrelease⟩\let\@citex@checkblank\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\cite`.)

`\@citex` `\penalty\@m` added to definition of `\@citex` to allow a line break after the ',' in citations like [Jones80,Smith77] (Added 23 Oct 86)

space added after the ',' (21 Nov 87)

```
\def\@citex[#1]#2{\leavevmode
  \let\@citea\@empty
  \@cite{\@for\@citeb:=#2\do
    {\@citea\def\@citea{,\penalty\@m\ }%
     \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
     \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
```

Using `\hbox` instead of `\mbox` is fine because of the `\leavevmode` above. In fact the use of a box around the citation contents is more than questionable in my view (FMi), but within 2e I have to keep that for compatibility reasons as it would probably change too many existing documents. Its main reason is to avoid hyphenation of labels such as [FOOB89] into [FOO- B89] so in certain styles it makes sense; but, for example, in author year citations it becomes more than questionable.

So Chris added yet another hook here, as suggested by, at least, Donald Arseneau. Note that this one is inside the first argument of the `\@cite` hook. This decouples the top-level typesetting of the citation from the details of the other business conducted here. All this really needs a complete rethink to get the right modularity.

```
      \@ifundefined{b@\@citeb}{\hbox{\reset@font\bfseries ?}%
        \G@refundefinedtrue
        \@latex@warning
          {Citation '\@citeb' on page \thepage \space undefined}}%
        {\@cite@ofmt{\csname b@\@citeb\endcsname}}}}{#1}}
```

(*End of definition for* `\@citex`.)

`\bibdata`
`\bibstyle`

```
\let\bibdata=\@gobble
\let\bibstyle=\@gobble
```

(*End of definition for* `\bibdata` *and* `\bibstyle`.)

`\bibliography`

```
\def\bibliography#1{%
  \if@filesw
    \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty}}%
  \fi
  \@input@{\jobname.bbl}}
```

(*End of definition for* `\bibliography`.)

`\bibliographystyle`

```
\def\bibliographystyle#1{%
  \ifx\@begindocumenthook\@undefined\else
    \expandafter\AtBeginDocument
  \fi
    {\if@filesw
       \immediate\write\@auxout{\string\bibstyle{#1}}%
     \fi}}
```

(*End of definition for* `\bibliographystyle`.)

`\nocite` (Added 14 Jun 85)

This puts information on the `.aux` file that causes BibTeX to include the citation list in the bibliography, but puts nothing in the text.

RmS 93/08/06: Made loop for `\nocite` like that for `\@citex`, to get rid of leading spaces.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                  {\nocite}{Allow nocite in preamble}%
\def\nocite#1{\@bsphack
```

With the implementation designed already in LaTeX 2.09 the `\nocite` command will not work before `\begin{document}` since it tries to write to the `.aux` file which is not open before that point. As a result the "reference" will appear on the terminal and nothing else will happen.

[This would be easy to fix, but then a document using the fix will silently fail on an older release of LaTeX, missing all citations done with `\nocite`. Thus we do only generate an error message and leave the fix for a LaTeX 2_ε successor.]

Given that we are now a quarter century into using LaTeX 2_ε there is no good reason any more do limit ourself to 2.09 conciderations. So we now simply delay the `\nocide` if it is issued in the preamble.

```
  \ifx\@onlypreamble\document
```

Since we are after `\begin{document}` we can do the citations:

```
    \@for\@citeb:=#1\do{%
      \edef\@citeb{\expandafter\@firstofone\@citeb}%
      \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
      \@ifundefined{b@\@citeb}{\G@refundefinedtrue
          \@latex@warning{Citation ‘\@citeb’ undefined}}{}}%
  \else
```

But before `\begin{document}` we raised an error message in the past but as of 2021/05 not any longer.

```
%    \@latex@error{Cannot be used in preamble}\@eha
```

Instead we delay the declaration to the start of the document. We have to use a late hook for this, so that it comes after the `.aux` file is open for writing and after `\@preamblecmds` was executed to change the above test. Therefore `\AtBeginDocument` would still be too early.

```
    \AddToHook{begindocument/end}[kernel]{\nocite{#1}}%
  \fi
```

```
  \@esphack}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\nocite}{Allow nocite in preamble}%
⟨latexrelease⟩
⟨latexrelease⟩\def\nocite#1{\@bsphack
⟨latexrelease⟩  \ifx\@onlypreamble\document
⟨latexrelease⟩    \@for\@citeb:=#1\do{%
⟨latexrelease⟩      \edef\@citeb{\expandafter\@firstofone\@citeb}%
⟨latexrelease⟩      \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
⟨latexrelease⟩      \@ifundefined{b@\@citeb}{\G@refundefinedtrue
⟨latexrelease⟩          \@latex@warning{Citation ‘\@citeb’ undefined}}{}}%
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@latex@error{Cannot be used in preamble}\@eha
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \@esphack}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

Since `\nocite{*}` should not produce a warning about undefined citation keys (see PR 557), we need to set the control sequence ‘`\b@*`’ to something other than `\relax`. As a result `\cite{*}` will not warn either (but that never worked with BibTeX in the first place).

```
\expandafter\let\csname b@*\endcsname\@empty
```

(*End of definition for* `\nocite`.)

1.1 Default definitions

This hook determines the ‘relative formatting’ of the two logical parts of a citation with comment.

`\@cite`

```
\def\@cite#1#2{[{#1\if@tempswa , #2\fi}]}
```

(*End of definition for* `\@cite`.)

`\@cite@ofmt` This is, in general, a command that appears to have one argument whose value is, in the kernel, a single cs whose name is the expansion of `b@\@citeb`; the expansion of this cs will typically be some hmode material that produces the detailed typeset form of just the citations themselves.

```
\let\@cite@ofmt\hbox
```

(*End of definition for* `\@cite@ofmt`.)

`\@biblabel`

```
\def\@biblabel#1{[#1]}
⟨/2ekernel⟩
```

(*End of definition for* `\@biblabel`.)

File 48
ltmarks.dtx

Abstract

Marks are used to communicate information about the content of a page to the output routine. For example, in order to construct running headers, the output routine needs information about which section names are present on a page, and this information is passed to it through the mark system. However, marks may also be used for other purposes. This module provides a generalized mechanism for marks of independent classes.

1 Introduction

The TEX engines offer a low-level mark mechanism to communicate information about the content of the current page to the asynchronous operating output routine. It works by placing `\mark` commands into the source document. When the material for the current page is assembled in box 255, TEX scans for such marks and sets the commands `\topmark`, `\firstmark` and `\botmark`. The `\firstmark` receives the content of the first `\mark` seen in box 255 and `\botmark` the content of the last mark seen. The `\topmark` holds the content of the last mark seen on the previous page or more exactly the value of `\botmark` from the previous page. If there are no marks on the current page then all three are made equal to the `\botmark` from the previous page.

This mechanism works well for simple formats (such as plain TEX) whose output routines are only called to generate pages. It fails, however, in LATEX (and other more complex formats), because here the output routine is sometimes called without producing a page, e.g., when encountering a float and placing it into one of the float regions. In that case the output routine is called, determines where to place the float, alters the goal for assembling text material (if the float was added to the top or bottom region) and then it resumes collecting textual material.

As a result the `\botmark` gets updated and so `\topmark` no longer reflects the situation at the top of the next page when that page is finally boxed.

Another problem for LATEX was that it wanted to use several "independent" marks and in the early implementations of TEX there was only a single `\mark` command available. For that reason LATEX implemented its own mark mechanism where the marks always contained two parts with their own interfaces: `\markboth` and `\markright` to set marks and `\leftmark` and `\rightmark` to retrieve them.

However, this extended mechanism (while supporting scenarios such as chapter/section marks) was far from general. The mark situation at the top of a page (i.e., `\topmark`) remained unusable and the two marks offered were not really independent of each other because `\markboth` (as the name indicates) was always setting both.

The new mechanism overcomes both issues:

- It provides arbitrarily many, fully independent named marks, that can be allocated and, from that point onwards, used.
- It offers access for each such marks to retrieve its top, first, and bottom values separately.
- Furthermore, the mechanism is augmented to give access to marks in different "regions" which may not be just full pages.

2 Design-level and code-level interfaces

The interfaces are mainly meant for package developers, but they are usable (with appropriate care) also in the document preamble, for example, when setting up special running headers with fancyhdr, etc. They are therefore available both as CamelCase commands as well as commands for use in the L3 programming layer. Both are described together below.

`\NewMarkClass` `\NewMarkClass {⟨class⟩}`
`\mark_new_class:n` `\mark_new_class:n {⟨class⟩}`

Declares a new ⟨*class*⟩ of marks to be tracked by LaTeX. Each ⟨*class*⟩ must be declared before it is used.

Mark classes can only be declared before `\begin{document}`.

`\InsertMark` `\InsertMark {⟨class⟩} {⟨text⟩}`
`\mark_insert:nn` `\mark_insert:nn {⟨class⟩} {⟨text⟩}`

Adds a mark to the current galley for the ⟨*class*⟩, containing the ⟨*text*⟩.

It has no effect in places in which you can't place floats, e.g., a mark inside a box or inside a footnote never shows up anywhere.

If used in vertical mode it obeys LaTeX's internal `\@nobreak` switch, i.e., it does not introduce a breakpoint if used after a heading. If used in horizontal mode it doesn't handle spacing (like, for example, `\index` or `\label` does, so it should be attached to material that is typeset.

`insertmark` `\AddToHook {insertmark} {⟨code⟩}`

When marks are inserted, the mark content may need some special treatment, e.g., by default `\label`, `\index`, and `\glossary` do not expand at this time (but only later if and when the mark content is actually used. In order to allow packages to augment or alter this setup there is a public hook `insertmark` that is executed at this point. It runs in a group so local modification to commands are only applied to the ⟨*text*⟩ argument of `\InsertMark` or `\mark_insert:nn`.

\TopMark	⋆ \TopMark [⟨*region*⟩] {⟨*class*⟩}
\FirstMark	⋆ \FirstMark [⟨*region*⟩] {⟨*class*⟩}
\LastMark	⋆ \LastMark [⟨*region*⟩] {⟨*class*⟩}
\mark_use_top:nn	⋆ \mark_use_top:nn {⟨*region*⟩} {⟨*class*⟩}
\mark_use_first:nn	⋆ \mark_use_first:nn {⟨*region*⟩} {⟨*class*⟩}
\mark_use_last:nn	⋆ \mark_use_last:nn {⟨*region*⟩} {⟨*class*⟩}

These functions expand to the appropriate mark ⟨*text*⟩ for the given ⟨*class*⟩ in the specified ⟨*region*⟩. The default ⟨*region*⟩ in the design-level commands is `page`. Note that with the L3 layer commands there are no optional arguments, i.e., both arguments have to be provided.

TEXhackers note: The result is returned within the `\unexpanded` primitive (`\exp_not:n`), which means that the ⟨*text*⟩ does not expand further when appearing in an `x`-type or `e`-type argument expansion.

The "first" and "last" marks are those seen first and last in the current region/page, respectively. The "top" mark is the last mark of the ⟨*class*⟩ seen in an earlier region, i.e., the ⟨*text*⟩ what would be "current" at the very top of the region.

Important! The commands are only meaningful inside the output routine, in other places their result is (while not random) unpredictable due to the way LATEX cuts text material into pages.

Currently, ⟨*region*⟩ is one of `page`, `previous-page`, `column`, and `previous-column`. If a page has just been finished then the region `page` refers to the current page and `previous-page`, as the name indicates, to the page that has been finished previously. This means you are able to access mark information for the current page as well as for the page before if you are inside the output routine, without the need to explicitly save that information beforehand.

In single column documents the `column` is the same as the `page` region, but in two-column documents, `column` refers to the current column that just got finished and previous-column to the one previously finished. Code for running headers are (in standard LATEX) only evaluated when both columns are assembled, which is another way of saying that in that case `previous-column` refers to the left column and `column` to the right column. However, to make this a bit nicer to access, there are also alias regions named `first-column` and `last-column`[40] to access these regions.[41]

Note that you can only look backwards at already processed regions, e.g., in a `twoside` document finishing a recto (odd, right-hand) page you can access the data from the facing verso (left-hand) page, but if you are finishing a left-hand page you can't integrate data from the upcoming right-hand page. If such a scenario needs to be realized then it is necessary to save the left-hand page temporarily instead of finalizing it, process material for the right-hand page and once both are ready, attach running headers and footers and shipout out both in one go.[42]

[40] This is called "last" not "second" in anticipation of extending the mechanism to multiple columns, where first and last would still make sense.

[41] At the moment there aren't any `previous-...-column` regions to access the columns from the previous page. If necessary, the mechanism could be easily augmented to cover them too, though.

[42] As of now that scenario is not yet officially supported.

\IfMarksEqualTF ⋆	\IfMarksEqualTF [⟨*region*⟩] {⟨*class*⟩} {⟨*pos*$_1$⟩} {⟨*pos*$_2$⟩} {⟨*true*⟩} {⟨*false*⟩}
\IfMarksEqualT ⋆	\mark_if_eq:nnnnTF {⟨*region*⟩} {⟨*class*⟩} {⟨*pos*$_1$⟩} {⟨*pos*$_2$⟩} {⟨*true*⟩} {⟨*false*⟩}
\IfMarksEqualF ⋆	\mark_if_eq:nnnnnnTF {⟨*region*$_1$⟩} {⟨*class*$_1$⟩} {⟨*pos*$_1$⟩}
\mark_if_eq:nnnnTF ⋆	{⟨*region*$_2$⟩} {⟨*class*$_2$⟩} {⟨*pos*$_2$⟩} {⟨*true*⟩} {⟨*false*⟩}
\mark_if_eq:nnnnnnTF ⋆	

These conditionals allow you to compare the content of two marks and act based on the result. The commands work in an expansion context, if necessary.

It is quite common when programming with marks to need to interrogate conditions such as whether marks have appeared on a previous page, or if there are multiple marks present on the current page, and so on. The tests above allow for the construction of a variety of typical test scenarios, with three examples presented below.

The first two conditionals cover only the common scenarios. Both marks are picked up from the same ⟨*region*⟩ (by default `page`) and they have to be of the same ⟨*class*⟩.[43] The ⟨*pos*$_i$⟩ argument can be either `top`, `first`, or `last`.

Important to note is that the comparison is not with respect to the textual content of the marks but whether or not they originated from the same `\InsertMark` command (or the L3 layer version `\mark_insert:nn`).

If you wish to compare marks across different regions or across different classes, you have to do it using the generic test only available in the L3 programming layer or do it manually, i.e., get the marks and then compare the values yourself.[44]

2.1 Use cases for conditionals

However, the basic version is enough for the following typical use cases:

Test for at most one mark of class `myclass` on current page: If the first and last mark in a region are the same then either there was no mark at all, or there was at most one. To test this on the current page:

```
\NewMarkClass{myclass}
\IfMarksEqualTF{myclass}{first}{last}
                { <zero or one mark> }{ <two or more marks> }
```

Test for no mark of class `myclass` in the previous page: If the top mark is the same as the first mark, there is no mark in the region at all. If we wanted to do this test for the previous page:

```
\IfMarksEqualTF[previous-page]{myclass}{top}{first}
                { <no marks> }{ <at least one mark> }
```

Comparing `top` and `last` would give you the same result.

Test for zero, one, or more than one: Combining the two tests from above you can test for zero, one or more than one mark.

```
\IfMarksEqualTF{myclass}{top}{first}
                { <no marks> }
                {\IfMarksEqualTF{myclass}{first}{last}
                  { <exactly one mark> }{ <more than one mark> }}
```

[43] If an undeclared mark class is used the tests return *true* (not an error).

[44] If two undeclared mark classes are compared the result is always *true*; if a declared and an undeclared mark class is used it is always *false*.

If you need one of such tests more often (or if you want a separate command for it for readability), then consider defining:

```
\providecommand\IfNoMarkTF[2][page]{\IfMarksEqualTF[#1]{#2}{first}{last}}
```

2.2 Debugging mark code

\DebugMarksOn
\DebugMarksOff
\mark_debug_on:
\mark_debug_off:

`\DebugMarksOn ... \DebugMarksOff`

Commands to turn the debugging of mark code on or off. The debugging output is rather coarse and not really intended for normal use at this point in time.

3 Application examples

If you want to figure out if a break was taken at a specific point, e.g., whether a heading appears at the top of the page, you can do something like this:

```
\newcounter{breakcounter}
\NewMarkClass{break}
\newcommand\markedbreak[1]{\stepcounter{breakcounter}%
                           \InsertMark{break}{\arabic{breakcounter}%
                           \penalty #1\relax
                           \InsertMark{break}{-\arabic{breakcounter}}
```

To test if the break was taken you can test if `\TopMark{break}` is positive (taken) or negative (not taken) or zero (there was never any marked break so far). The absolute value can be used to keep track of which break it was (with some further coding).

to be extended with additional application examples

4 Legacy LaTeX 2_ε interface

Here we describe the interfaces that LaTeX 2_ε offered since the early nineties and some minor extensions.

4.1 Legacy design-level and document-level interfaces

\markboth
\markright

`\markboth {⟨left⟩} {⟨right⟩}`
`\markright {⟨right⟩}`

LaTeX 2_ε uses two marks which aren't fully independent. A "left" mark generated by the first argument of `\markboth` and a "right" mark generated by the second argument of `\markboth` or by the only argument of `\markright`. The command `\markboth` and `\markright` are in turn called from heading commands such as `\chaptermark` or `\sectionmark` and their behavior is controlled by the document class.

For example, in the `article` class with `twoside` in force the `\sectionmark` will issue `\markboth` with an empty second argument and `\subsectionmark` will issue `\markright`. As a result the left mark will contain chapter titles and the right mark subsection titles.

Note, however, that in one-sided documents the standard behavior is that only `\markright` is used, i.e., there will only be right-marks but no left marks!

\leftmark ⋆ \leftmark
\rightmark ⋆ \rightmark

These functions return the appropriate mark value from the current page and work as before, that is \leftmark will get the last (!) left mark from the page and \rightmark the first (!) right mark.

In other words they work reasonably well if you want to show the section title that is current when you are about to turn the page and also show the first subsection title on the current page (or the last from the previous page if there wasn't one). Other combinations can't be shown using this interface.

The commands are fully expandable, because this is how they have been always defined in LaTeX. However, this is of course only true if the content of the mark they return is itself expandable and does not contain any fragile material. Given that this can't be guaranteed for arbitrary content, a programmer using them in this way should use \protected@edef and *not* \edef to avoid bad surprises as far as this is possible, or use the new interfaces (\TopMark, \FirstMark, and \LastMark) which return the ⟨*text*⟩ in \exp_not:n to prevent uncontrolled expansion.

4.2 Legacy interface extensions

The new implementation adds three mark classes: 2e-left, 2e-right and 2e-right-nonempty and patches \markboth and \markright slightly so that they also update these new mark classes, so that the new classes work with existing document classes.

As a result you can use \LastMark{2e-left} and \FirstMark{2e-right} instead of \leftmark and \rightmark. But more importantly, you can use any of the other retrieval commands to get a different status value from those marks, e.g., \LastMark{2e-right} would return the last subsection on the page (instead of the first as returned by \rightmark).

The difference between 2e-right and 2e-right-nonempty is that the latter will only be updated if the material for the mark is not empty. Thus \markboth{title}{} as issued by, say, \sectionmark, sets a 2e-left mark with title and a 2e-right mark with the empty string but does not add a 2e-right-nonempty mark.

Thus, if you have a section at the start of a page and you would ask for \FirstMark{2e-right} you would get an empty string even if there are subsections on that page. But 2e-right-nonempty would then give you the first or last subsection on that page. Of course, nothing is simple. If there are no subsections it would tell you the last subsection from an earlier page. We therefore need comparison tools, e.g., if top and first are identical you know that the value is bogus, i.e., a suitable implementation would be

```
\IfMarksEqualTF{2e-right-nonempty}{top}{first}
               { <appropriate action if there was no real mark> }
               {\FirstMark{2e-right-nonempty}}
```

5 Notes on the mechanism

In contrast to vanilla TeX, ε-TeX extends the mark system to allow multiple independent marks. However, it does not solve the \topmark problem which means that LaTeX still needs to manage marks almost independently of TeX. The reason for this is that the more complex output routine used by LaTeX to handle floats (and related structures)

means that `\topmark(s)` remain unreliable. Each time the output routine is fired up, TEX moves `\botmark` to `\topmark`, and while ε-TEX extends this to multiple registers the fundamental concept remains the same. That means that the state of marks needs to be tracked by LATEX itself. An early implementation of this package used TEX's `\botmark` only to ensure the correct interaction with the output routine (this was before the ε-TEX mechanism was even available). However, other than in a prototype implementation for LATEX3, this package was never made public.

The new implementation now uses ε-TEX's marks as they have some advantages, because with them we can leave the mark text within the galley and only extract the marks during the output routine when we are finally shipping out a page or storing away a column for use in the next page. That means we do not have to maintain a global data structure that we have to keep in sync with informational marks in the galley but can rely on everything being in one place and thus manipulations (e.g. reordering of material) will take the marks with them without a need for updating a fragile linkage.

To allow for completely independent marks we use the following procedure:

- For every type of marks we allocate a mark class so that in the output routine TEX can calculate for each class the current top, first, and bottom mark independently. For this we use `\newmarks`, i.e., one marks register per class.
- As already mentioned firing up an output routine without shipping out a page means that TEX's top marks get wrong so it is impossible to rely on TEX's approach directly. What we do instead is to keep track of the real marks (for the last page or more generally last region) in some global variables.
- These variables are updated in the output routine at defined places, i.e., when we do real output processing but not if we use special output routines to do internal housekeeping.
- The trick we use to get correctly updated variables is the following: the material that contains new marks (for example the page to be shipped out) is stored in a box. We then use TEX primitive box splitting functions by splitting off the largest amount possible (which should be the whole box if nothing goes really wrong). While that seems a rather pointless thing to do, it has one important side effect: TEX sets up first and bottom marks for each mark class from the material it has split off. This way we get the first and last marks (if there have been any) from the material in the box.
- The top marks are simply the last marks from the previous page or region. And if there hasn't been a first or bottom mark in the box then the new top mark also becomes new first and last mark for that class.
- That mark data is then stored in global token lists for use during the output routine and legacy commands such as `\leftmark` or new commands such as `\TopMark` simply access the data stored in these token lists.

That's about it in a nutshell. Of course, there are some details to be taken care of—those are discussed in the implementation sections.

6 Internal output routine functions

The functions in this section are tied to the output routine and used in the interface to LaTeX 2ε and perhaps at some later time within a new output routine for LaTeX. They are not meant for general use and are therefore made internal. Internal means that `@@` automatically gets replaced in the code (and in the documentation) so we have to give it a suitable value.

```
⟨@@=mark⟩
```

`\__mark_update_singlecol_structures:` `\__mark_update_singlecol_structures:`

LaTeX 2ε integration function in case we are doing single column layouts. It assumes that the page content is already stored in `\@outputbox` and processes the marks inside that box. It is called as part of `\@opcol`.

`\__mark_update_dblcol_structures:` `\__mark_update_singlecol_structures:`

LaTeX 2ε integration function mark used when we are doing double column documents. It assumes that the page content is already stored in `\@outputbox` and processes the marks inside that box. It then does different post-processing depending on the start of the switch `\if@firstcolumn`. If we are in the second column it also has to update page marks, otherwise it only updates column marks. It too is called as part of `\@opcol`.

`\__mark_update_structure_from_material:nn` `\__mark_update_structure_from_material:nn {⟨region⟩} {⟨material with marks⟩}`

Helper function that inspects the marks inside the second argument and assigns new mark values based on that to the ⟨*region*⟩ given in the first argument. For this it first copies the mark structure from ⟨*region*⟩ to `previous-`⟨*region*⟩ and then takes all last mark values currently in the region and makes them the new top mark values. Finally it assigns new first and last values for all mark classes based on what was found in the second argument.

As a consequence, the allowed values for ⟨*region*⟩ are `page` and `column` because only they have `previous-...` counterparts.

Another important aspect to keep in mind is that marks are recognized only if they appear on the top level, e.g., if we want to process material stored in boxes we need to put it unboxed (using `\unvcopy` etc.) into the second argument.

`\__mark_update_structure_alias:nn` `\__mark_update_structure_alias:nn {⟨alias⟩} {⟨source⟩}`

Helper function that copies all mark values in the ⟨*source*⟩ region to ⟨*alias*⟩, i.e., make the structures identical. Used to update the `previous-...` structures inside `\__mark_update_structure_from_material:nn` and `first-column` and `last-column` structures inside `\__mark_update_singlecol_structures:` or `\__mark_update_dblcol_structures:`.

`\__mark_update_structure_to_err:n` `\__mark_update_structure_to_err:n {⟨region⟩}`

Helper function that sets all mark values in the ⟨*region*⟩ to an error message. This is currently used for `last-column` at times where using marks from it would be questionable/wrong, i.e., when we have just processed the first column in a two-column document.

__mark_get_marks_for_reinsertion:nNN

__mark_get_marks_for_reinsertion:nNN {⟨*source*⟩}
⟨*token-list-var for collecting first marks*⟩
⟨*token-list-var for collecting last marks*⟩

Helper function for extracting marks that would otherwise get lost, for example when they are hidden inside a box. This helper does not update mark structures and can therefore be used outside the output routine as well.

It collect all the top-level marks from inside the ⟨*source*⟩ and adds suitable `\mark_insert:nn` in the two token lists. These token lists can then be executed at the right place to reinsert the marks, e.g., directly after the box. This is, for example, going to be used[45] by multicol when a short balanced `multicols` is returned to the galley for typesetting.

If the ⟨*source*⟩ consists of a single vertical box (plus possibly followed by some glue but nothing else) then the box is unpacked and the top-level marks are collected from its content. However, if it is not a vertical box or there are are other data then nothing is unpacked and you have to do the unpacking yourself to get at the marks inside.

It is quite likely that one only needs a single token list for returning the `\mark_insert:nn` statements. If that is the case this command may change to take only two arguments.

7 The Implementation

```
⟨*2ekernel | latexrelease⟩
\ExplSyntaxOn
⟨latexrelease⟩\NewModuleRelease{2022/06/01}{ltmarks}
⟨latexrelease⟩                    {Marks~handling}
```

7.1 Allocating new mark classes

\g__mark_classes_seq

A list holding all the mark classes that have been declared.

```
\seq_new:N \g__mark_classes_seq
```

\mark_new_class:n
__mark_new_class:nn

A mark class is created by initializing a number of data structures. First, we get a register number to refer to the mark class. The new mark class is then added to the `\g__mark_classes_seq` sequence to be able to easily loop over all classes. Finally a number of top-level global token lists are declared that hold various versions of the mark for access.

```
\cs_new_protected:Npn \mark_new_class:n #1
{
  \seq_if_in:NnTF \g__mark_classes_seq {#1}
      {
        \msg_error:nnn { mark } { class-already-defined }
          {#1}
      }
      { \__mark_new_class:nn {#1} }
}
```

This is only available in the preamble.

```
\@onlypreamble \mark_new_class:n
```

[45] Probably not before 2025, though.

The internal command carries out the necessary allocations.

```
\cs_new_protected:Npn \__mark_new_class:nn #1
{
⟨*trace⟩
  \__mark_debug:n { \iow_term:x { Marks:~new~mark:~#1~\msg_line_context: } }
⟨/trace⟩
```

Use the LaTeX 2_ε interface for now as the L3 programming layer doesn't have one for marks yet.

```
  \exp_args:Nc \newmarks {c__mark_class_ #1 _mark}
```

Remember the new class in the sequence.

```
  \seq_gput_right:Nn \g__mark_classes_seq {#1}
```

We need three token lists for each region, one for top, first, and last.

```
  \tl_new:c   { g__mark_page_top_   #1 _tl }
  \tl_new:c   { g__mark_page_first_ #1 _tl }
  \tl_new:c   { g__mark_page_last_  #1 _tl }
```

For the `page` region we also keep track of the `previous-page`.

```
  \tl_new:c   { g__mark_previous-page_top_   #1 _tl }
  \tl_new:c   { g__mark_previous-page_first_ #1 _tl }
  \tl_new:c   { g__mark_previous-page_last_  #1 _tl }
```

Same game for `column` and `previous-column`

```
  \tl_new:c   { g__mark_column_top_   #1 _tl }
  \tl_new:c   { g__mark_column_first_ #1 _tl }
  \tl_new:c   { g__mark_column_last_  #1 _tl }
  \tl_new:c   { g__mark_previous-column_top_   #1 _tl }
  \tl_new:c   { g__mark_previous-column_first_ #1 _tl }
  \tl_new:c   { g__mark_previous-column_last_  #1 _tl }
```

But for columns we also allocate token lists for the alias regions `first-column` and `last-column`.

```
  \tl_new:c   { g__mark_first-column_top_   #1 _tl }
  \tl_new:c   { g__mark_first-column_first_ #1 _tl }
  \tl_new:c   { g__mark_first-column_last_  #1 _tl }
  \tl_new:c   { g__mark_last-column_top_    #1 _tl }
  \tl_new:c   { g__mark_last-column_first_  #1 _tl }
  \tl_new:c   { g__mark_last-column_last_   #1 _tl }
```

All marks will have an identification at the beginning of the form `\__mark_id:n{`⟨*number*⟩`}` and therefore the initial empty values should have that too, so that data extraction is going to be uniform.

```
  \tl_set:cn   { g__mark_page_top_   #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_page_first_ #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_page_last_  #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_previous-page_top_   #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_previous-page_first_ #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_previous-page_last_  #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_column_top_   #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_column_first_ #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_column_last_  #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_previous-column_top_   #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_previous-column_first_ #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_previous-column_last_  #1 _tl }{ \__mark_id:n{0} }
```

```
  \tl_set:cn   { g__mark_first-column_top_   #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_first-column_first_ #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_first-column_last_  #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_last-column_top_    #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_last-column_first_  #1 _tl }{ \__mark_id:n{0} }
  \tl_set:cn   { g__mark_last-column_last_   #1 _tl }{ \__mark_id:n{0} }
}
```

(*End of definition for* `\mark_new_class:n` *and* `\__mark_new_class:nn`*. This function is documented on page* *943*.)

7.2 Updating mark structures

`\l__mark_box`
`\l__mark_ii_box`
`\g__mark_tmp_tl`
`\g__mark_new_top_tl`

For some operations we need two temporary private boxes and two private global token lists.

```
\box_new:N \l__mark_box
\box_new:N \l__mark_ii_box
\tl_new:N  \g__mark_tmp_tl
\tl_new:N  \g__mark_new_top_tl
```

(*End of definition for* `\l__mark_box` *and others.*)

`\__mark_extract_and_handle_marks:nn`

This is the main macro to extract and handle marks inside some vertical material. It is used by `\__mark_update_structure_from_material:nn` (for updating the mark structure for a region based on the marks found) and by `\__mark_get_marks_for_reinsertion:nNN` (for extracting marks from some material and prepare for reinserting them later (e.g., out of a box that is placed as a box into the main galley).

```
\cs_new_protected:Npn \__mark_extract_and_handle_marks:nn #1#2 {
```

This macro expects code to handle extracted marks in its first argument and vertical material (not boxed or just consisting of a single vertical box) as its second. It extracts top-level mark information from `#2`, stores them as split marks and then calls `#1` to make use of this information.

If it finds a forced break in the material it removes it and then restarts the attempt without it.

We start with a group to keep most changes local.

```
  \group_begin:
```

Getting the first and last marks out of the material in `#2` is done by putting the material in a box and then doing a split operation to the maximum size possible (which hopefully gets us all of the content).[46] Because this action is used only to get the mark values, we don't want any underfull box warnings so we (locally) turn those off.

```
    \dim_set_eq:NN \tex_splitmaxdepth:D \c_max_dim
    \int_set_eq:NN \tex_vbadness:D      \c_max_int
    \dim_set_eq:NN \tex_vfuzz:D         \c_max_dim
```

There is a further complication: if the material contains infinite shrinking glue then a `\vsplit` operation will balk with a low-level error. Now pages or columns, which are our main concern here, can't have such infinite shrinkage if they are cut straight from the galley, however the use of `\enlargethispage` actually does add some at the very bottom

[46] With normal column material cut from the main galley we should always get all material in one go, but in certain situations, for example, in a multicols environment that contains some `\columnbreak`s a single split operation will not be enough. Thus, this is something we need to handle.

(and also wraps the whole page into a box by itself, so if we leave it this way then a) we get this error and b) we don't see any marks because they are hidden one level down).

Another possible issue are packages or user code that place stray \vboxes directly into the main galley (an example is marginnote that attaches its marginals in this way). If such boxes end up as the last item on the page we should not unpack them.

All these issues need to be handled, which is done in __mark_prepare_and_extract:nn.

```
    \__mark_prepare_and_extract:nn {#1} {#2}
```

Once all mark classes have been processed, the data structures are updated and we can close the group, which undoes our local changes and retains only the global ones.

```
  \group_end:
}
```

(*End of definition for* __mark_extract_and_handle_marks:nn.)

__mark_prepare_and_extract:nn

This macro does the dirty work. It is not directly integrated in __mark_extract_and_handle_marks:nn because we may have to call it recursively if we find forced breaks.

```
\cs_new_protected:Npn \__mark_prepare_and_extract:nn #1#2 {
```

To handle the \enlargethispage case we do an \unskip to get rid of any glue that is present at the very end of the material and also check if we have then a \vbox as the last item and if so unpack that too, but only under certain conditions, see below. All this is temporary done in a group, just for getting the marks out, so it doesn't affect the final page production.

```
 \vbox_set:Nn \l__mark_box
   {
     #2
     \tex_unskip:D
     \box_set_to_last:N \l__mark_box
```

After having removed the last box from the current list (if there was one) we check whether the vertical list is now empty. If not, then the last box is definitely not the one from \enlargethispage and so we can, and should, leave it alone. Otherwise we check if this last box is a \vbox.

```
     \int_compare:nNnT \tex_lastnodetype:D < 0
       {
         \box_if_vertical:NT \l__mark_box
```

If it is, we unpack the box.

```
           { \vbox_unpack:N \l__mark_box }
       }
```

If it wasn't a vbox, it was either an hbox or there was no box. Given that we are only interested in the marks we don't need put it back in that case.

```
   }
```

We are now ready to \vsplit the box to get at the marks. If the box contains some infinite negative glue the TEX will produce an error complaining about it but it will correctly find the the split marks. Given that we can't prevent that error, we hide it from the user and ensure that TEX doesn't stop. The error message still shows in the log, but even that is mitigated as best as possible—see the definition of __mark_vbox_set_split_to_maxdimen:NN for the tricks employed.

```
   \__mark_vbox_set_split_to_maxdimen:NN \l__mark_ii_box \l__mark_box
```

After splitting we check if there is anything left in `\l__mark_box`. If not then the above split has set some split marks that we can then use to finish the extraction:

```
    \box_if_empty:NTF \l__mark_box
      { #1 }
```

If we have a remainder after the split then this means that there was some forced break in the material. We get rid of that by combining the content of the two boxes and restart.

```
      {
⟨*trace⟩
        \__mark_debug:n { \iow_term:x
          { Marks:~ mark~ extraction~needs~ recursion~
            \msg_line_context: } }
⟨/trace⟩
        \__mark_prepare_and_extract:nn {#1}
           { \vbox_unpack:N \l__mark_ii_box
             \vbox_unpack:N \l__mark_box    }
      }
}
```

(*End of definition for* `\__mark_prepare_and_extract:nn`.)

`\__mark_vbox_set_split_to_maxdimen:NN` Split a box to get at its marks without pausing even if TEX is producing an error message because of infinite negative glue in the box. If there is such an error we ensure that it only shows up in the log but not on the terminal.

The nice low-level hack by DPC records in the `.log` that a glue shrinkage error is harmless.

We disguise `\c_max_dim` in an odd looking csname, which then shows up as part of the display of an error message if that error happens. This csname forms part of the error display so what you get is something like

```
! Infinite glue shrinkage found in box being split.
<argument> Infinite shrink error above ignored !
l. ...  }
```

which hopefully makes it clear that the error is harmless and and should be ignored by the reader of the `.log`.

```
\cs_set_eq:cN {Infinite~shrink~error~above~ignored~!}\c_max_dim
```

The whole definition of `\__mark_vbox_set_split_to_maxdimen:NN` below is fully expanded, so we have to use a lot of `\exp_not:N` commands to prevent expansion where necessary.

```
\cs_new_protected:Npx \__mark_vbox_set_split_to_maxdimen:NN #1#2 {
```

We start by saving the current interaction and escape char settings.

```
  \tl_set:Ne \exp_not:N \l__mark_saved_parameters_tl
     {
       \tex_interactionmode:D
          \exp_not:N \int_use:N \tex_interactionmode:D \scan_stop:
       \tex_escapechar:D
          \exp_not:N \int_use:N \tex_escapechar:D \scan_stop:
     }
```

Then we change them so that no escape char is printed in the error message (accounts for the missing backslash in front of `Infinite shrink ...`) and we set the interaction to `\nonstopmode` so that the the error (if any) just goes into the `.log` file and TEX doesn't stop at that point.

```
\tex_escapechar:D     -1 \scan_stop:
\tex_interactionmode:D 0 \scan_stop:
```

Then we do the splitting of the box to `\c_max_dim` to get at the marks. This may generate the error we are worried about, i.e., if the box contains infinite negative glue. However, TEX makes this glue finite and continues, which means we get our split marks which is really all we care about.

```
\tex_setbox:D #1 \tex_vsplit:D #2 to
```

The `\use:n` may seem pointless, and it is to some extent, but we need it to get our disguised `\c_max_dim` displayed properly as part of the error message if there is one. Without it, the display would show only part of what we want it to show (try it).

```
      \exp_not:N \use:n {
        \use:c{Infinite~shrink~error~above~ignored~!}
      }
```

Finally, we change the escape char and the interaction mode back to what it was before:

```
  \exp_not:N \l__mark_saved_parameters_tl
}
```

(*End of definition for* `\__mark_vbox_set_split_to_maxdimen:NN`.)

`\l__mark_saved_parameters_tl` The temporary variable used for resetting escape char and interaction mode.

```
\tl_new:N \l__mark_saved_parameters_tl
```

(*End of definition for* `\l__mark_saved_parameters_tl`.)

`\__mark_update_structure_from_material:nn` This function updates the mark structures of a region. The first argument is the region to update and second argument receives the material that holds the marks. Out of this material we extract the first and last marks for all classes (if there are any) to do the assignments.

```
\cs_new_protected:Npn \__mark_update_structure_from_material:nn #1#2 {
  \__mark_extract_and_handle_marks:nn
```

Once the marks can be extracted we update the structure from the split marks (code in `\__mark_update_structure_from_splitmarks:n`).

```
    { \__mark_update_structure_from_splitmarks:n {#1} }
    { #2 }
}
```

(*End of definition for* `\__mark_update_structure_from_material:nn`.)

`\__mark_update_structure_from_splitmarks:n` This macro is called after we have done a `\tex_vsplit:D` operation and the mark data is in the split marks.

```
\cs_new_protected:Npn \__mark_update_structure_from_splitmarks:n #1 {
```

The first thing we do is to copy the current region structure to `previous-...`; this leaves the current structure untouched so we can update it class by class (which is necessary).

```
  \__mark_update_structure_alias:nn { previous-#1 } {#1}
```

After this action we can get first and last marks of the various classes through `\tex_splitfirstmarks:D` and `\tex_splitbotmarks:D`. So now we loop over all classes stored in `\g__mark_classes_seq`.

```
  \seq_map_inline:Nn \g__mark_classes_seq
            {
```

First action: get the last mark from the previous region, i.e., `previous-#1`. But because it is also still inside `#1`, at the moment we use that to construct the name because this is a tiny bit faster. Given that we need this value in various assignments we store it away which avoids unnecessary further csname generations.

```
              \tl_gset_eq:Nc \g__mark_new_top_tl { g__mark_#1_last_##1_tl }
```

This will first of all become the new top mark for the current class.

```
              \tl_gset_eq:cN { g__mark_#1_top_##1_tl } \g__mark_new_top_tl
```

Next action is to get ourselves the new last mark from the material supplied.

```
              \tl_gset:No \g__mark_tmp_tl
                { \tex_splitbotmarks:D \use:c { c__mark_class_##1_mark } }
```

If this mark doesn't exist then obviously neither does the first mark, so both become the last mark from the previous region. We have to be a little careful here: something like `\mark_insert:nn{foo}{}` adds an "empty" mark that should not be confused with no mark at all. But no mark in our material will result in `\g__mark_tmp_tl` being fully empty. This is why we have to make sure that "empty" from `\mark_insert:nn` only appears to be empty when typeset but fails the next test (see below how this is done).

```
              \tl_if_empty:NTF \g__mark_tmp_tl
                {
                  \tl_gset_eq:cN { g__mark_#1_last_ ##1_tl }
                    \g__mark_new_top_tl
                  \tl_gset_eq:cN { g__mark_#1_first_##1_tl }
                    \g__mark_new_top_tl
                }
```

If it wasn't empty, i.e., if it had a real value then we use this value for our new last mark instead.

```
                {
                  \tl_gset_eq:cN { g__mark_#1_last_##1_tl } \g__mark_tmp_tl
```

Because we had a last mark we also have a first mark (which might be the same, but might be not), so we pick that up and assign it to the appropriate token list. This explains why we first checked for the last mark because that makes the processing faster in case there is none.

```
                  \tl_gset:co { g__mark_#1_first_##1_tl }
                    {
                      \tex_splitfirstmarks:D
                        \use:c { c__mark_class_##1_mark }
                    }
                }
            }
}
```

(*End of definition for* `\__mark_update_structure_from_splitmarks:n`.)

__mark_get_marks_for_reinsertion:nNN

This function extracts the marks from the material in the first argument but it does not update any the mark structures. Instead, it collects the marks in the token lists given as the second and third argument, in such a way that they can be reinserted by just executing the token lists.[47]

```
\cs_new_protected:Npn \__mark_get_marks_for_reinsertion:nNN #1#2#3 {
```

First we clear the temporary token lists as we haven't seen any marks yet.

```
  \tl_clear:N \g__mark_first_marks_tl
  \tl_clear:N \g__mark_last_marks_tl
```

Then we extract all top-level marks, thereby filling the token lists with suitable `\mark_insert:nn` calls.

```
  \__mark_extract_and_handle_marks:nn
```

The first argument holds the code for fill the token lists and the second is the material we extract from.

```
      \__mark_get_from_splitmarks:
      { #1 }
```

Finally, we copy the updated (or not updated) temporary token lists to the two that have been supplied when the function was called. By convention "get" operations return their values in local variables and `\__mark_extract_and_handle_marks:nn` runs in a group, which is why we have to use global temporary variables for collecting.

```
  \tl_set_eq:NN #2 \g__mark_first_marks_tl
  \tl_set_eq:NN #3 \g__mark_last_marks_tl
}
```

(*End of definition for* `\__mark_get_marks_for_reinsertion:nNN`.)

__mark_get_from_splitmarks:

This function is called after we have done a `\vsplit` to update the split marks. It loops through all mark classes to find out if there are marks for this class and if so updates the global tls used for collecting.

```
\cs_new_protected:Npn  \__mark_get_from_splitmarks:  {
  \seq_map_inline:Nn \g__mark_classes_seq
            {
```

First we to get the last mark for the current class from the material supplied.

```
              \tl_gset:No \g__mark_tmp_tl
                { \tex_splitbotmarks:D \use:c { c__mark_class_##1_mark } }
```

If this mark doesn't exist then obviously first mark doesn't either, so we do nothing (other than issuing some debugging info).

We have to be a little careful here: something like `\mark_insert:nn{foo}{}` adds an "empty" mark that we should not confuse with the case where there is no mark at all.

When there is no mark at all we get a truly empty `\g__mark_tmp_tl` as a result. This is why we have to make sure that an "empty" mark generated with `\mark_insert:nn` only appears to be empty when it is typeset, but fails the next test (see below how this is done).

```
              \tl_if_empty:NTF \g__mark_tmp_tl
                {
⟨*trace⟩
```

[47] It is probably enough to collect everything in a single token list as long as we put the first marks first and the last marks last). But for extra flexibility, I currently use 2 token lists. This might change when it is really clear that this is never needed.

```
                    \__mark_debug:n { \iow_term:x { Marks:~no~ marks~
                        for~ class~ '##1'~\msg_line_context: } }
⟨/trace⟩
                  }
```

If it wasn't empty, i.e., if it had a real value then we use this value for our new last mark instead. This means we put an appropriate `\mark_insert:nn` statement into `\g__mark_last_marks_tl`.

```
                  {
⟨*trace⟩
                    \__mark_debug:n { \iow_term:x { Marks:~ extract~ last~
                         mark~ for~ class~ '##1'~ =~  \g__mark_tmp_tl } }
⟨/trace⟩
                    \tl_gput_right:Ne \g__mark_last_marks_tl
                       { \mark_insert:nn {##1} { \g__mark_tmp_tl } }
```

Because we had a last mark we also have a first mark (which might be the same, but might not be), so we pick that up and add it to the `\g__mark_first_marks_tl` token list. This explains why we first checked for the last mark because that makes the processing faster in case there is none.

```
⟨*trace⟩
                    \__mark_debug:n { \iow_term:x { Marks:~ extract~ first~
                        mark~ for~ class~ '##1'~ =~
                        \tex_splitfirstmarks:D
                          \use:c { c__mark_class_##1_mark }
                    } }
⟨/trace⟩
                    \tl_gput_right:Ne \g__mark_first_marks_tl
                       { \mark_insert:nn {##1}
                         {
                           \tex_splitfirstmarks:D
                           \use:c { c__mark_class_##1_mark }
                         }
                       }
                  }
              }
}
```

(*End of definition for* `\__mark_get_from_splitmarks:`.)

`\g__mark_first_marks_tl`
`\g__mark_last_marks_tl`

These are two global temporary variables used in the code above.

```
\tl_new:N  \g__mark_first_marks_tl
\tl_new:N  \g__mark_last_marks_tl
```

(*End of definition for* `\g__mark_first_marks_tl` *and* `\g__mark_last_marks_tl`.)

`\__mark_update_structure_alias:nn`

This function copies the structure for one region to another (name), e.g., from `page` to `previous-page` above, or later from `column` to `first-column`, etc.

```
\cs_new_protected:Npn \__mark_update_structure_alias:nn #1#2 {
```

This requires a simple loop through all mark classes copying the token list from one name to the next.

```
  \seq_map_inline:Nn \g__mark_classes_seq
    {
      \tl_gset_eq:cc { g__mark_ #1 _top_   ##1 _tl }
```

```
                    { g__mark_ #2 _top_   ##1 _tl }
      \tl_gset_eq:cc { g__mark_ #1 _first_ ##1 _tl }
                    { g__mark_ #2 _first_ ##1 _tl }
      \tl_gset_eq:cc { g__mark_ #1 _last_  ##1 _tl }
                    { g__mark_ #2 _last_  ##1 _tl }
    }
}
```

(*End of definition for* `\__mark_update_structure_alias:nn`.)

`\__mark_update_structure_to_err:n` `\__mark_error:n`

A slight variation is to install a fixed error message as the value.

```
\cs_new_protected:Npn \__mark_update_structure_to_err:n #1 {
  \seq_map_inline:Nn \g__mark_classes_seq
      {
        \tl_gset:cn { g__mark_ #1 _top_   ##1 _tl } { \__mark_error:n {#1} }
        \tl_gset:cn { g__mark_ #1 _first_ ##1 _tl } { \__mark_error:n {#1} }
        \tl_gset:cn { g__mark_ #1 _last_  ##1 _tl } { \__mark_error:n {#1} }
      }
}
```

Given that this is used in only one place, we could hardwire the argument which would be a bit more compact, but who knows, perhaps we end up with another reason to use this error command elsewhere, so for now we keep the argument.

```
\cs_new_protected:Npn \__mark_error:n #1 {
  \msg_error:nnn { mark } { invalid-use } {#1}
}
```

(*End of definition for* `\__mark_update_structure_to_err:n` *and* `\__mark_error:n`.)

7.3 Placing and retrieving marks

`\mark_insert:nn`

This function puts a mark for some ⟨*class*⟩ at the current point.

```
\cs_new_protected:Npn \mark_insert:nn #1#2
{
  \seq_if_in:NnTF \g__mark_classes_seq {#1}
      {
```

We need to pass the evaluated argument into the mark but protected commands should not expand including those protected using the `\protect` approach of LaTeX 2_ε. We also disable `\label` and the like.[48]

At this point the code eventually should get a public (and a kernel) hook instead of a set of hardwired settings.

```
        \group_begin:
```

Within the group we alter some comments, e.g, `\label` or `\index`, to do the right at this point. This is done in the kernel hook `\@kernel@before@insertmark` which is followed by the public hook `insertmark` that can be used by packages to augment or alter that setup as necessary.

```
          \@kernel@before@insertmark
          \hook_use:n { insertmark }
          \unrestored@protected@xdef \g__mark_tmp_tl
               {
```

[48] Straight copy from `latex.ltx` but is this even correct? At least a label in a running header makes little sense if it get set several times! Maybe that needs looking at in the 2e kernel.

To ensure that marks are unique we insert a hidden sequence marker at the beginning of the content of the mark containing the sequence number of the mark.

```
                  \__mark_id:n{ \int_use:N\g__mark_int }
                  #2
                }
⟨*trace⟩
          \__mark_debug:n{ \iow_term:x { Marks:~ set~#1~<-~
              '\tl_to_str:V \g__mark_tmp_tl' ~ \msg_line_context: } }
⟨/trace⟩
          \tex_marks:D \use:c { c__mark_class_ #1 _mark }
            {
```

Here is the trick to avoid truly empty marks: if the result from the above processing is empty we add something which eventually becomes empty, but not immediately; otherwise we just put `\g__mark_tmp_tl` in.

```
% this is no longer needed with 1.0f
%               \tl_if_empty:NTF \g__mark_tmp_tl
%                 { \exp_not:n { \prg_do_nothing: } }
%                 { \exp_not:o { \g__mark_tmp_tl } }
               \exp_not:o { \g__mark_tmp_tl }
             }
         \group_end:
```

A mark introduces a possible break point and in certain situations that should not happen in vertical mode in LaTeX. This needs some cleanup ....

```
         \if@nobreak\ifvmode\nobreak\fi\fi
       }
```

If the mark class was not known, raise an error.

```
       {
         \msg_error:nnx { mark } { unknown-class }
           { \tl_to_str:n {#1} }
       }
}
```

(*End of definition for* `\mark_insert:nn`. *This function is documented on page* 943.)

`\__mark_id:n` A hidden marker is placed into every mark added by `\mark_insert:nn`. It will will not show up in the output but its argument (a counter value that is incremented) makes all marks unique so the test for "equal" is not fooled by two different marks having the same mark text.

```
\cs_new_protected:Npn \__mark_id:n #1 { }
```

(*End of definition for* `\__mark_id:n`.)

`\@kernel@before@insertmark` `insertmark` By default `\label`, `\index`, and `\glossary` do nothing when the mark is inserted.

```
\int_new:N \g__mark_int
\cs_new:Npn \@kernel@before@insertmark {
          \cs_set_eq:NN \label    \scan_stop:
          \cs_set_eq:NN \index    \scan_stop:
          \cs_set_eq:NN \glossary \scan_stop:
```

We count each mark and use that to place a hidden marker in front of the mark text. To ensure that there is no overflow (very unlikely but you never know) we restart every 100000 marks. Thus, if somebody puts more than that number of marks on a single page you could construct a scenario in which that approach fails.

```
            \int_compare:nNnTF \g__mark_int < {99999}
                { \int_gincr:N \g__mark_int }
                { \int_gzero:N \g__mark_int }

}
```

The public hook to augment the setup.

```
\hook_new:n {insertmark}
```

(*End of definition for* `\@kernel@before@insertmark` *and* `insertmark`.)

`\mark_use_top:nn`
`\mark_use_first:nn`
`\mark_use_last:nn`

To retrieve the first, last or top region mark, we grab the appropriate value stored in the corresponding token list variable and pass its contents back. These functions should be used only in output routines and only after `\__mark_update_structure_from_material:nn` has acted, otherwise their value will be wrong.

If used with an unknown class or region they generate an error (fairly low-level because we are in an expandable context).

Each mark starts with an id and while the id does not print it is nevertheless better to remove it when returning the mark, so that downstream manipulation of the data doesn't have to deal with it.

```
\cs_new:Npn \mark_use_first:nn #1#2 { \__mark_use:v { g__mark_#1_first_#2_tl } }
\cs_new:Npn \mark_use_last:nn #1#2  { \__mark_use:v { g__mark_#1_last_#2_tl }  }
\cs_new:Npn \mark_use_top:nn #1#2   { \__mark_use:v { g__mark_#1_top_#2_tl }   }
```

This is what the `\use_none:nn` accomplishes.

```
\cs_new:Npn \__mark_use:n #1 { \exp_not:o { \use_none:nn #1 } }
\cs_generate_variant:Nn \__mark_use:n { v }
```

(*End of definition for* `\mark_use_top:nn`, `\mark_use_first:nn`, *and* `\mark_use_last:nn`. *These functions are documented on page* 944.)

7.4 Comparing mark values

`\mark_if_eq:nnnnTF`
`\mark_if_eq:nnnnnnTF`

Test if in a given region (`#1`) for a given class (`#2`) the marks in position `#3` and `#4` (top, first, or last) are identical

```
\prg_new_conditional:Npnn \mark_if_eq:nnnn #1#2#3#4 { T , F , TF }
{
  \tl_if_eq:ccTF { g__mark_ #1 _#3_ #2 _tl }
                 { g__mark_ #1 _#4_ #2 _tl }
                 \prg_return_true:
                 \prg_return_false:
}
```

The fully general test (with two triplets of the form ⟨*region*⟩, ⟨*class*⟩, and ⟨*position*⟩) is this:

```
\prg_new_conditional:Npnn \mark_if_eq:nnnnnn #1#2#3#4#5#6 { T , F , TF }
{
  \tl_if_eq:ccTF { g__mark_ #1 _#3_ #2 _tl }
                 { g__mark_ #4 _#6_ #5 _tl }
                 \prg_return_true:
```

```
                    \prg_return_false:
}
```

(*End of definition for* `\mark_if_eq:nnnnTF` *and* `\mark_if_eq:nnnnnnTF`*. These functions are documented on page 945.*)

7.5 Messages

Mark errors are LaTeX kernel errors:

```
\prop_gput:Nnn \g_msg_module_type_prop { mark } { LaTeX }
\msg_new:nnnn { mark } { class-already-defined }
  { Mark~class~'#1'~already~defined }
  {
    \c__msg_coding_error_text_tl
    LaTeX~was~asked~to~define~a~new~mark~class~called~'#1':~
    this~mark~class~already~exists.
    \c__msg_return_text_tl
  }
\msg_new:nnnn { mark } { unknown-class }
  { Unknown~mark~class~'#1'. }
  {
    \c__msg_coding_error_text_tl
    LaTeX~was~asked~to~manipulate~a~mark~of~class~'#1',~
    but~this~class~of~marks~does~not~exist.
  }

\msg_new:nnnn { mark } { invalid-use }
  { Mark~region~'#1'~not ~usable }
  {
    \c__msg_coding_error_text_tl
    The~region~'#1'~can~only~be~used~after~
    all~columns~have~been~assembled.
    \c__msg_return_text_tl
  }
```

7.6 Debugging the mark structures

Code and commands in this section are not final, it needs more experimentation to see what kind of tracing information is going to be useful in practice. For now the tracing is mainly meant to be used for code testing and not so much for application testing.

It is quite likely that the commands and the behavior of the tracing might change in the future once we gained some experience with it.

`\g__mark_debug_bool` Holds the current debugging state.

```
\bool_new:N \g__mark_debug_bool
```

(*End of definition for* `\g__mark_debug_bool`.)

`\mark_debug_on:`
`\mark_debug_off:`
`\__mark_debug:n`
`\__mark_debug_gset:`

Turns debugging on and off by redefining `\__mark_debug:n`.

```
\cs_new_eq:NN \__mark_debug:n \use_none:n
\cs_new_protected:Npn \mark_debug_on:
  {
```

```
    \bool_gset_true:N \g__mark_debug_bool
    \__mark_debug_gset:
  }
\cs_new_protected:Npn \mark_debug_off:
  {
    \bool_gset_false:N \g__mark_debug_bool
    \__mark_debug_gset:
  }
\cs_new_protected:Npn \__mark_debug_gset:
  {
    \cs_gset_protected:Npx \__mark_debug:n ##1
      { \bool_if:NT \g__mark_debug_bool {##1} }
  }
```

(*End of definition for* `\mark_debug_on:` *and others. These functions are documented on page* 946.)

`\DebugMarksOn` `\DebugMarksOff`

CamelCase commands for debugging.

```
\cs_new_eq:NN \DebugMarksOn   \mark_debug_on:
\cs_new_eq:NN \DebugMarksOff  \mark_debug_off:
```

(*End of definition for* `\DebugMarksOn` *and* `\DebugMarksOff`*. These functions are documented on page* 946.)

`\__mark_class_status:nn`

Shows the mark values across all regions for one mark class (#2). The first argument gives some ⟨*info*⟩ to help identifying where the command was called.

```
⟨*trace⟩
\cs_new_protected:Npn \__mark_class_status:nn #1#2
   {
            \typeout{ Marks:~#2~ #1:}
            \typeout{\@spaces page~ (current):
              | \exp_not:v { g__mark_page_top_ #2 _tl }
              | \exp_not:v { g__mark_page_first_ #2 _tl }
              | \exp_not:v { g__mark_page_last_ #2 _tl }     |}
            \typeout{\@spaces page~ (previous):
              | \exp_not:v { g__mark_previous-page_top_ #2 _tl }
              | \exp_not:v { g__mark_previous-page_first_ #2 _tl }
              | \exp_not:v { g__mark_previous-page_last_ #2 _tl }     |}
            \typeout{\@spaces column~ (previous):
              | \exp_not:v { g__mark_previous-column_top_ #2 _tl }
              | \exp_not:v { g__mark_previous-column_first_ #2 _tl }
              | \exp_not:v { g__mark_previous-column_last_ #2 _tl }     |}
            \typeout{\@spaces column~ (current):
              | \exp_not:v { g__mark_column_top_ #2 _tl }
              | \exp_not:v { g__mark_column_first_ #2 _tl }
              | \exp_not:v { g__mark_column_last_ #2 _tl }     |}
            \typeout{\@spaces column~ (first):
              | \exp_not:v { g__mark_first-column_top_ #2 _tl }
              | \exp_not:v { g__mark_first-column_first_ #2 _tl }
              | \exp_not:v { g__mark_first-column_last_ #2 _tl }     |}
            \typeout{\@spaces column~ (second):
              | \exp_not:v { g__mark_last-column_top_ #2 _tl }
              | \exp_not:v { g__mark_last-column_first_ #2 _tl }
              | \exp_not:v { g__mark_last-column_last_ #2 _tl }     |}
}
```

(*End of definition for* `\__mark_class_status:nn`.)

`\__mark_status:n` Show a snapshot of all mark class values across all regions.

```
\cs_new_protected:Npn \__mark_status:n #1
  {
    \seq_map_inline:Nn \g__mark_classes_seq
      { \__mark_class_status:nn {#1} {##1} }
  }
⟨/trace⟩
```

(*End of definition for* `\__mark_status:n`.)

`\ShowMarksAt` Debugging helper that displays a snapshot of all known mark structures. The argument is a text string that is displayed to help identifying when the snapshot was made.

This may not stay like this (or at all), which is why it isn't yet documented as an official command.

```
\cs_new_protected:Npn \ShowMarksAt #1 {
⟨*trace⟩
     \__mark_debug:n { \__mark_status:n {#1} }
⟨/trace⟩
}
```

(*End of definition for* `\ShowMarksAt`.)

7.7 Designer-level interfaces

`\NewMarkClass` `\InsertMark` These two are identical to the L3 programming layer commands.

```
\cs_new_eq:NN  \NewMarkClass \mark_new_class:n
\@onlypreamble \NewMarkClass
```

```
\cs_new_eq:NN \InsertMark  \mark_insert:nn
```

(*End of definition for* `\NewMarkClass` *and* `\InsertMark`. *These functions are documented on page* 943.)

`\TopMark` `\FirstMark` `\LastMark` The following commands take an optional argument that defaults to page. There is no checking that the region is actually valid. If not there is simply an empty return.

```
\NewExpandableDocumentCommand \FirstMark { O{page} m }
                    { \mark_use_first:nn {#1}{#2} }
```

```
\NewExpandableDocumentCommand \LastMark { O{page} m }
                    { \mark_use_last:nn {#1}{#2} }
```

```
\NewExpandableDocumentCommand \TopMark { O{page} m }
                    { \mark_use_top:nn {#1}{#2} }
```

(*End of definition for* `\TopMark`, `\FirstMark`, *and* `\LastMark`. *These functions are documented on page* 944.)

`\IfMarksEqualTF` `\IfMarksEqualT` `\IfMarksEqualF` We only provide CamelCase commands for the case with one region (optional) and one class. One could think of also providing a version for the general case with several optional arguments, but use cases for this are most likely rare, so not done yet.

```
\NewExpandableDocumentCommand \IfMarksEqualTF {O{page}mmm} {
  \mark_if_eq:nnnnTF {#1}{#2}{#3}{#4}
}
\NewExpandableDocumentCommand \IfMarksEqualT {O{page}mmm} {
  \mark_if_eq:nnnnT {#1}{#2}{#3}{#4}
```

```
}
\NewExpandableDocumentCommand \IfMarksEqualF {O{page}mmm} {
  \mark_if_eq:nnnnF {#1}{#2}{#3}{#4}
}
```

(*End of definition for* `\IfMarksEqualTF`, `\IfMarksEqualT`, *and* `\IfMarksEqualF`. *These functions are documented on page* 945.)

8 LaTeX 2ε integration

8.1 Core LaTeX 2ε integration

`\__mark_update_singlecol_structures:` This command updates the mark structures if we are producing a single column document.

```
\cs_new_protected:Npn \__mark_update_singlecol_structures: {
```

First we update the `page` region (which also updates the `previous-page`.

The `\@outputbox` is normally in `\vbox` in LaTeX but we can't take that for granted (an amsmath test document changed it to an `\hbox` just to trip me up) so we are a little careful with unpack now.

```
  \box_if_vertical:NTF \@outputbox
      {
        \__mark_update_structure_from_material:nn {page}
           { \vbox_unpack:N  \@outputbox }
      }
      {
        \__mark_update_structure_from_material:nn {page}
           { \hbox_unpack:N  \@outputbox }
      }
```

The we provide the necessary updates for the aliases.

```
  \__mark_update_structure_alias:nn {previous-column}{previous-page}
  \__mark_update_structure_alias:nn {column}{page}
  \__mark_update_structure_alias:nn {first-column}{page}
  \__mark_update_structure_alias:nn {last-column}{page}
⟨*trace⟩
% move this into status itself?
    \__mark_debug:n
        {
          \__mark_status:n
               { in~ OR~ (
                 \legacy_if:nTF {@twoside}
                        { twoside-
                          \int_if_odd:nTF \c@page
                              { odd }{ even }
                        }
                        { oneside }
                        )
               }
        }
⟨/trace⟩
}
```

(*End of definition for* `\__mark_update_singlecol_structures:`.)

__mark_update_dblcol_structures: This commands handles the updates if we are doing two-column pages.

```
\cs_new_protected:Npn \__mark_update_dblcol_structures: {
```

First we update the `column` and `previous-column` regions using the material assembled in `\@outputbox`.

```
  \box_if_vertical:NTF \@outputbox
      {
        \__mark_update_structure_from_material:nn {column}
           { \vbox_unpack:N  \@outputbox }
      }
      {
        \__mark_update_structure_from_material:nn {column}
           { \hbox_unpack:N  \@outputbox }
      }
```

How we have to update the alias regions depends on whether or not `\@opcol` was called to process the first column or to produce the completed page

```
  \legacy_if:nTF {@firstcolumn}
    {
```

If we are processing the first column then `column` is our `first-column` and there is no `last-column` yet, so we make those an error.

```
      \__mark_update_structure_alias:nn {first-column}{column}
      \__mark_update_structure_to_err:n {last-column}
    }
    {
```

If we produce the completed page then the `first-column` is the same as the new `previous-column`. However, the structure should already be correct if you think about it (because is was set to `column` last time which is now the `previous-column`), thus there is no need to make an update.

```
%     \__mark_update_structure_alias:nn {first-column}{previous-column}
```

However, we now have a proper `last-column` so we assign that.

```
      \__mark_update_structure_alias:nn {last-column}{column}
```

What now remains doing is to update the `page` and `previous-page` regions. For this we have to copy the settings in `page` into `previous-page` and then update `page` such that the top and first marks are taken from the `first-column` region and the last marks are taken from the `last-column` region. All this has to be done for all mark classes so we loop over our sequence.

Note that one loop is needed if we arrange the copy statements in a suitable way.

```
      \seq_map_inline:Nn \g__mark_classes_seq
        {
```

The `previous-page` updates need to come before the updates for `page` region because otherwise the values to copy are already overwritten. necessary values.

```
          \tl_gset_eq:cc { g__mark_previous-page_top_   ##1 _tl }
                         { g__mark_page_top_            ##1 _tl }
          \tl_gset_eq:cc { g__mark_previous-page_first_ ##1 _tl }
                         { g__mark_page_first_          ##1 _tl }
          \tl_gset_eq:cc { g__mark_previous-page_last_  ##1 _tl }
                         { g__mark_page_last_           ##1 _tl }
```

To update the `top` we only have to copy what is in `first-column`:

```
          \tl_gset_eq:cc { g__mark_page_top_           ##1 _tl }
                         { g__mark_first-column_top_   ##1 _tl }

```

Updating the `first` mark for the `page` region is more complicated. We first have to find out of there is any mark in the first column (this can be done by comparing the `top` and the `first` mark of of that region).

```
          \tl_if_eq:ccTF { g__mark_first-column_top_   ##1 _tl }
                         { g__mark_first-column_first_ ##1 _tl }
             {
```

If there is no mark in the first column we copy the first mark of the last column. If that doesn't contain a mark we still get the right result because the first mark is then equal to the top mark.

```
               \tl_gset_eq:cc { g__mark_page_first_        ##1 _tl }
                              { g__mark_last-column_first_ ##1 _tl }
             }
             {
```

On the other hand, if there is a mark in the first column we copy over the `first` mark from that column.

```
               \tl_gset_eq:cc { g__mark_page_first_         ##1 _tl }
                              { g__mark_first-column_first_ ##1 _tl }
             }
```

The logic for the `last` page mark is again simple, we can just copy the value in the `last` mark of the last column. If that column doesn't contain any marks, then the value in `last` will be automatically the same as the `last` from the first column.

```
          \tl_gset_eq:cc { g__mark_page_last_        ##1 _tl }
                         { g__mark_last-column_last_ ##1 _tl }
        }
    }
⟨*trace⟩
     \__mark_debug:n
         {
           \__mark_status:n
              { in~ OR~ (
                \legacy_if:nTF {@twoside}
                       { twoside-
                         \int_if_odd:nTF \c@page
                             { odd }{ even }
                       }
                       { oneside }
                       \space
                       \legacy_if:nTF {@firstcolumn}
                              { first~ }{ second~ }
                              column )
              }
         }
⟨/trace⟩
}
```

(*End of definition for* `\__mark_update_dblcol_structures:`.)

```
⟨@@=⟩
```

\@expl@@@mark@update@singlecol@structures@@

```
\cs_new_eq:NN  \@expl@@@mark@update@singlecol@structures@@
               \__mark_update_singlecol_structures:
```

(*End of definition for* `\@expl@@@mark@update@singlecol@structures@@`.)

\@expl@@@mark@update@dblcol@structures@@

```
\cs_new_eq:NN  \@expl@@@mark@update@dblcol@structures@@
               \__mark_update_dblcol_structures:
```

(*End of definition for* `\@expl@@@mark@update@dblcol@structures@@`.)

8.2 Other LaTeX 2_ε output routines

This section will cover multicol and other packages altering or providing their own output routine. Not done yet.

8.3 Rollback information

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{ltmarks}%
⟨latexrelease⟩                  {Undo~Marks~handling}
⟨latexrelease⟩
```

We keep the interface commands around even if we roll back in case they are used in packages that don't roll back. Not likely to do a lot of good, but then there is not much we can do, but this at least they won't give unknown csname errors.

```
⟨latexrelease⟩\DeclareRobustCommand \NewMarkClass[1]{}
⟨latexrelease⟩\DeclareRobustCommand \InsertMark[2]{}
⟨latexrelease⟩\RenewExpandableDocumentCommand \FirstMark { O{} m } { }
⟨latexrelease⟩\RenewExpandableDocumentCommand \LastMark { O{} m } { }
⟨latexrelease⟩\RenewExpandableDocumentCommand \TopMark { O{} m } { }
⟨latexrelease⟩\RenewExpandableDocumentCommand \IfMarksEqualTF { O{} mmm }{ }
⟨latexrelease⟩
```

Same here, this avoided extra roll back code in the OR.

```
⟨latexrelease⟩\let \@expl@@@mark@update@singlecol@structures@@ \relax
⟨latexrelease⟩\let \@expl@@@mark@update@dblcol@structures@@ \relax
⟨latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩\EndModuleRelease
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
```

Reset module prefix:

```
⟨@@=⟩
```

File 49
ltpage.dtx

1 Page styles and related commands

1.1 Page Style Commands

`\pagestyle{`⟨*style*⟩`}` : sets the page style of the current and succeeding pages to *style*

`\thispagestyle{`⟨*style*⟩`}` : sets the page style of the current page only to *style*.

To define a page style *style*, you must define `\ps@`*style* to set the page style parameters.

1.2 How a page style makes running heads and feet

The `\ps@`... command defines the macros `\@oddhead`, `\@oddfoot`, `\@evenhead`, and `\@evenfoot` to define the running heads and feet. (See output routine.) To make headings determined by the sectioning commands, the page style defines the commands `\chaptermark`, `\sectionmark`, etc., where `\chaptermark{`⟨*text*⟩`}` is called by `\chapter` to set a mark. The `\...mark` commands and the `\...head` macros are defined with the help of the following macros.

(All the `\...mark` commands should be initialized to no-ops.)

1.3 marking conventions

LaTeX extends TeX's `\mark` facility by producing two kinds of marks a 'left' and a 'right' mark, using the following commands:

`\markboth{`⟨*left*⟩`}{`⟨*right*⟩`}` : Adds both marks.

`\markright{`⟨*right*⟩`}` : Adds a 'right' mark.

`\leftmark` : Used in the output routine, gets the current 'left' mark. Works like TeX's `\botmark`.

`\rightmark` : Used in the output routine, gets the current 'right' mark. Works like TeX's `\firstmark`. The marking commands work reasonably well for right marks 'numbered within' left marks—e.g., the left mark is changed by a `\chapter` command and the right mark is changed by a `\section` command. However, it does produce somewhat anomalous results if 2 `\markboth`'s occur on the same page.

Commands like `\tableofcontents` that should set the marks in some page styles use a `\@mkboth` command, which is `\let` by the pagestyle command (`\ps@...`) to `\markboth` for setting the heading or to `\@gobbletwo` to do nothing.

```
⟨*2ekernel⟩
```

`\pagestyle` User command to set the page style for this and following pages.

```
\def\pagestyle#1{%
  \@ifundefined{ps@#1}%
    \undefinedpagestyle
    {\@nameuse{ps@#1}}}
```

(*End of definition for* `\pagestyle`.)

\thispagestyle User command to set the page style for this page only.

```
\def\thispagestyle#1{%
  \@ifundefined{ps@#1}%
    \undefinedpagestyle
    {\global\@specialpagetrue\gdef\@specialstyle{#1}}}
```

(*End of definition for* \thispagestyle.)

\ps@empty The empty page style: No head or foot line.

```
\def\ps@empty{%
  \let\@mkboth\@gobbletwo\let\@oddhead\@empty\let\@oddfoot\@empty
  \let\@evenhead\@empty\let\@evenfoot\@empty}
```

(*End of definition for* \ps@empty.)

\ps@plain The plain page style: No head, centred page number in foot.

```
\def\ps@plain{\let\@mkboth\@gobbletwo
      \let\@oddhead\@empty\def\@oddfoot{\reset@font\hfil\thepage
      \hfil}\let\@evenhead\@empty\let\@evenfoot\@oddfoot}
```

(*End of definition for* \ps@plain.)

\@leftmark
\@rightmark We implement \@leftmark and \@rightmark in terms of already defined commands to save token space. We can't get rid of them since they are sometimes used in applications.

```
\let\@leftmark\@firstoftwo
\let\@rightmark\@secondoftwo
```

(*End of definition for* \@leftmark *and* \@rightmark.)

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                    {\markboth}{New mark support}%
```

\markboth
\markright User commands for setting LaTeX marks.

Test for \@nobreak added 15 Apr 86 in \markboth and \markright letting \label and \index to \relax added 22 Feb 86 so these commands can appear in sectioning command arguments RmS 91/06/21 Same for \glossary

```
\ExplSyntaxOn
\DeclareRobustCommand*\markboth[2]{%
  \begingroup
    \let\label\relax \let\index\relax \let\glossary\relax
    \unrestored@protected@xdef\@themark {{#1}{#2}}%
    \@temptokena \expandafter{\@themark}%
```

In addition to generating the legacy mark we output the individual ones as well at the very same point. The legacy mark is kept unchanged in order to work with packages that expect that mark in exactly the way it is right now.

We might want to think about how to improve this in one-side documents, see comments below.

We have not changed all of the code to L3 prog layer convention, in case packages attempt to do some patching and expect the 2e names being around. Eventually this should and will change.

```
    \mark_insert:nn{2e-left}{#1}
    \mark_insert:nn{2e-right}{#2}
```

```
    \tl_if_empty:nF{#2}{ \mark_insert:nn{2e-right-nonempty}{#2} }
    \mark{\the\@temptokena}%
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi}
```

```
\DeclareRobustCommand*\markright[1]{%
  \begingroup
    \let\label\relax \let\index\relax \let\glossary\relax
```

Protection is handled inside `\@markright`.

```
    \expandafter\@markright\@themark {#1}%
    \@temptokena \expandafter{\@themark}%
```

Same game with `\markright` more or less ...

```
    \mark_insert:nn{2e-right}{#1}
    \tl_if_empty:nF{#1}{ \mark_insert:nn{2e-right-nonempty}{#1} }
```

The legacy LaTeX mechanism always sets left and right mark, i.e., if a sub-mark (i.e., right mark) is set the corresponding main mark also is getting a mark with the same value it had previously. However, for the individual mark classes this means we are losing information so for them that is not done.

```
%   \mark_insert:nn{2e-left}{\exp_after:wN \use_i:nn \@themark }
    \mark{\the\@temptokena}%
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi}
\ExplSyntaxOff
```

(*End of definition for* `\markboth` *and* `\markright`*. These functions are documented on page 946.*)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩                  {\markboth}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand*\markboth[2]{%
⟨latexrelease⟩  \begingroup
⟨latexrelease⟩    \let\label\relax \let\index\relax \let\glossary\relax
⟨latexrelease⟩    \unrestored@protected@xdef\@themark {{#1}{#2}}%
⟨latexrelease⟩    \@temptokena \expandafter{\@themark}%
⟨latexrelease⟩    \mark{\the\@temptokena}%
⟨latexrelease⟩  \endgroup
⟨latexrelease⟩  \if@nobreak\ifvmode\nobreak\fi\fi}
⟨latexrelease⟩\DeclareRobustCommand*\markright[1]{%
⟨latexrelease⟩  \begingroup
⟨latexrelease⟩    \let\label\relax \let\index\relax \let\glossary\relax
⟨latexrelease⟩    \expandafter\@markright\@themark {#1}%
⟨latexrelease⟩    \@temptokena \expandafter{\@themark}%
⟨latexrelease⟩    \mark{\the\@temptokena}%
⟨latexrelease⟩  \endgroup
⟨latexrelease⟩  \if@nobreak\ifvmode\nobreak\fi\fi}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\markboth}{Make commands robust}%
⟨latexrelease⟩
⟨latexrelease⟩\kernel@make@fragile\markboth
```

```
⟨latexrelease⟩\kernel@make@fragile\markright
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\@markright
\leftmark
\rightmark

```
\def\@markright#1#2#3{\@temptokena {#1}%
  \unrestored@protected@xdef\@themark{{\the\@temptokena}{#3}}}
\def\leftmark{\expandafter\@leftmark\botmark\@empty\@empty}
\def\rightmark{\expandafter\@rightmark\firstmark\@empty\@empty}
```

(*End of definition for* \@markright*,* \leftmark*, and* \rightmark*.*)

\@themark

Initialise LaTeX's marks without setting a TeX mark ⟨*whatsit*⟩.

```
\def\@themark{{}{}}
```

(*End of definition for* \@themark*.*)

\mark

Test versions of LaTeX 2_ε initialised TeX's \mark system at this point, but this was removed before the first release.

```
AtBeginDocument{\mark{{}{}}}
```

(*End of definition for* \mark*.*)

\raggedbottom

\raggedbottom typesets pages with no vertical stretch, so they have their natural height instead of all being exactly the same height. (Uses a space of .0001fil to avoid interfering with the 1fil space of \newpage.)

```
\DeclareRobustCommand\raggedbottom{%
  \def\@textbottom{\vskip \z@ \@plus.0001fil}\let\@texttop\relax}
```

(*End of definition for* \raggedbottom*.*)

\flushbottom

\flushbottom: Inverse of \raggedbottom — makes all pages the same height.

```
\DeclareRobustCommand\flushbottom{%
  \let\@textbottom\relax \let\@texttop\relax}
```

(*End of definition for* \flushbottom*.*)

\sloppy

\sloppy will never (well, hardly ever) produce overfull boxes, but may produce underfull ones. (14 June 85)

```
\DeclareRobustCommand\sloppy{%
  \tolerance 9999%
  \emergencystretch 3em%
  \hfuzz .5\p@
  \vfuzz\hfuzz}
```

(*End of definition for* \sloppy*.*)

sloppypar (*env.*)

A sloppypar environment is equivalent to {\par \sloppy ... \par}.

```
\def\sloppypar{\par\sloppy}
\def\endsloppypar{\par}
```

`\fussy` Resets TEX's parameters to their normal finicky values.

```
\DeclareRobustCommand\fussy{%
  \emergencystretch\z@
  \tolerance 200%
  \hfuzz .1\p@
  \vfuzz\hfuzz}
```

(*End of definition for* `\fussy`.)

`\overfullrule` LATEX default is no overfull box rule. Changed by document class option.

```
\overfullrule 0pt
```

(*End of definition for* `\overfullrule`.)

⟨/2ekernel⟩

File 50
ltclass.dtx

1 Introduction

This file implements the following declarations, which replace `\documentstyle` in LaTeX 2_ε documents.

Note that old documents containing `\documentstyle` will be run using a compatibility option—thus keeping everyone happy, we hope!

The overall idea is that there are two types of 'style files': 'class files' which define elements and provide a default formatting for them; and 'packages' which provide extra functionality. One difference between LaTeX 2_ε and LaTeX2.09 is that LaTeX 2_ε packages may have options. Note that options to classes packages may be implemented such that they input files, but these file names are not necessarily directly related to the option name.

2 User interface

`\documentclass[`⟨*main-option-list*⟩`]{`⟨*class*⟩`}[`⟨*version*⟩`]`

There must be exactly one such declaration, and it must come first. The ⟨*main-option-list*⟩ is a list of options which can modify the formatting of elements which are defined in the ⟨*class*⟩ file as well as in all following `\usepackage` declarations (see below). The ⟨*version*⟩ is a version number, beginning with a date in the format `YYYY/MM/DD`. If an older version of the class is found, a warning is issued.

`\documentstyle[`⟨*main-option-list*⟩`]{`⟨*class*⟩`}[`⟨*version*⟩`]`

The `\documentstyle` declaration is kept in order to maintain upward compatibility with LaTeX2.09 documents. It is similar to `\documentclass`, but it causes all options in ⟨*main-option-list*⟩ that the ⟨*class*⟩ does not use to be passed to `\RequirePackage` after the options have been processed. This maintains compatibility with the 2.09 behaviour. Also a flag is set to indicate that the document is to be processed in LaTeX2.09 compatibility mode. As far as most packages are concerned, this only affects the warnings and errors LaTeX generates. This flag does affect the definition of font commands, and `\sloppy`.

`\usepackage[`⟨*package-option-list*⟩`]{`⟨*package-list*⟩`}[`⟨*version*⟩`]`

There can be any number of these declarations. All packages in ⟨*package-list*⟩ are called with the same options.

Each ⟨*package*⟩ file defines new elements (or modifies those defined in the ⟨*class*⟩), and thus extends the range of documents which can be processed. The ⟨*package-option-list*⟩ is a list of options which can modify the formatting of elements defined in the ⟨*package*⟩ file. The ⟨*version*⟩ is a version number, beginning with a date in the format `YYYY/MM/DD`. If an older version of the package is found, a warning is issued.

Each package is loaded only once. If the same package is requested more than once, nothing happens, unless the package has been requested with options that were not given the first time it was loaded, in which case an error is produced.

As well as processing the options given in the ⟨*package-option-list*⟩, each package processes the ⟨*main-option-list*⟩. This means that options that affect all of the packages can be given globally, rather than repeated for every package.

Note that class files have the extension `.cls`, packages have the extension `.sty`.

`filecontents` (*env.*) The environment `filecontents` is intended for passing the contents of packages, options, or other files along with a document in a single file. It has one argument, which is the name of the file to create. If that file already exists (maybe only in the current directory if the OS supports a notion of a 'current directory' or 'default directory') then nothing happens (except for an information message) and the body of the environment is bypassed. Otherwise, the body of the environment is written verbatim to the file name given as the first argument, together with some comments about how it was produced.

The environment can also be called with an optional argument which is used to alter some of its behavior: option `force` or `overwrite` will allow for overwriting existing files, option `nosearch` will only check the current directory when looking if the file exists. This can be useful if you want to generate a local (modified) copy of some file that is already in the search tree of TEX. Finally, you can use `noheader` to prevent it from writing the standard blurb at the top of the file (this is actually the same as using the star form of the environment).

The environment is now allowed anywhere in the document, but to ensure that all packages or options necessary are available when the document is run, it is normally best to place it at the top of your file (before `\documentclass`). A possible use case for using it inside the document body is if you want to reuse some text several times in the document you could then write it and later use `\input` to retrieve it where needed.

The begin and end tags should each be on a line by itself.

2.1 Option processing

When the options are processed, they are divided into two types: *local* and *global*:

- For a class, the options in the `\documentclass` command are local.
- For a package, the options in the `\usepackage` command are local, and the options in the `\documentclass` command are global.

The options for `\documentclass` and `\usepackage` are processed in the following way:

1. The local and global options that have been declared (using `\DeclareOption` as described below) are processed first.

   In the case of `\ProcessOptions`, they are processed in the order that they were declared in the class or package.

   In the case of `\ProcessOptions*`, they are processed in the order that they appear in the option-lists. First the global options, and then the local ones.

2. Any remaining local options are dealt with using the default option (declared using the `\DeclareOption*` declaration described below). For document classes, this usually does nothing, but records the option on a list of unused options. For packages, this usually produces an error.

Finally, when `\begin{document}` is reached, if there are any global options which have not been used by either the class or any package, the system will produce a warning.

3 Class and Package interface

3.1 Class name and version

\ProvidesClass A class can identify itself with the \ProvidesClass{⟨*name*⟩}[⟨*version*⟩] command. The ⟨*version*⟩ should begin with a date in the format YYYY/MM/DD.

3.2 Package name and version

\ProvidesPackage A package can identify itself with the \ProvidesPackage{⟨*name*⟩}[⟨*version*⟩] command. The ⟨*version*⟩ should begin with a date in the format YYYY/MM/DD.

3.3 Requiring other packages

\RequirePackage Packages or classes can load other packages using
\RequirePackage[⟨*options*⟩]{⟨*name*⟩}[⟨*version*⟩].
If the package has already been loaded, then nothing happens unless the requested options are not a subset of the options with which it was loaded, in which case an error is called.

\LoadClass Similar to \RequirePackage, but for classes, may not be used in package files.

\PassOptionsToPackage Packages can pass options to other packages using:
\PassOptionsToPackage{⟨*options*⟩}{⟨*package*⟩}.

\PassOptionsToClass This adds the ⟨*options*⟩ to the options list of any future \RequirePackage or \usepackage command. For example:

```
\PassOptionsToPackage{foo,bar}{fred}
```

is the same as:

```
\RequirePackage[foo,bar,baz]{fred}
```

\LoadClassWithOptions \LoadClassWithOptions{⟨*name*⟩}[⟨*version*⟩]:
This is similar to \LoadClass, but it always calls class ⟨*name*⟩ with exactly the same option list that is being used by the current class, rather than an option explicitly supplied or passed on by \PassOptionsToClass. \RequirePackageWithOptions is the analogous command for packages.

\RequirePackageWithOptions

This is mainly intended to allow one class to simply build on another, for example:

```
\LoadClassWithOptions{article}
```

This should be contrasted with the slightly different construction

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

As used here, the effects are more or less the same, but the version using \LoadClassWithOptions is slightly quicker (and less to type). If, however, the class declares options of its own then the two constructions are different; compare, for example:

```
\DeclareOption{landscape}{...}
\ProcessOptions
\LoadClassWithOptions{article}
```

with:

```
\DeclareOption{landscape}{...}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

In the first case, the article class will be called with option `landscape` precisely when the current class is called with this option; but in the second example it will not as in that case article is only passed options by the default option handler, which is not used for `landscape` as that option is explicitly declared.

`\IfPackageLoadedTF` `\IfClassLoadedTF` `\@ifpackageloaded` `\@ifclassloaded`

To find out if a package has already been loaded, use

```
\IfPackageLoadedTF{⟨package⟩}{⟨true⟩}{⟨false⟩}
```

or the old name `\@ifpackageloaded`.

`\IfPackageAtLeastTF` `\IfClassAtLeastTF` `\IfFileAtLeastTF` `\@ifpackagelater` `\@ifclasslater`

To find out if a package has already been loaded with a version equal to or more recent than ⟨*date*⟩, use

```
\IfPackageAtLeastTF{⟨package⟩}{⟨date⟩}{⟨true⟩}{⟨false⟩}
```

or the old name `\@ifpackagelater`.

`\IfFormatAtLeastTF`

To test the format date use

```
\IfFormatAtLeastTF{⟨date⟩}{⟨true⟩}{⟨false⟩}
```

`\IfPackageLoadedWithOptionsTF` `\IfClassLoadedWithOptionsTF` `\@ifpackagewith` `\@ifclasswith`

To find out if a package has already been loaded with at least the options ⟨*options*⟩, use

```
\IfPackageLoadedWithOptionsTF{⟨package⟩}{⟨options⟩}{⟨true⟩}{⟨false⟩}
```

or the old name `\@ifpackagewith`.

There exists one package that can't be tested with the above commands: the `fontenc` package pretends that it was never loaded to allow for repeated reloading with different options (see `ltoutenc.dtx` for details).

3.4 Declaring new options

Options for classes and packages are built using the same macros.

`\DeclareOption` `\DeclareOption*`

To define a builtin option, use `\DeclareOption{⟨name⟩}{⟨code⟩}`.

To define the default action to perform for local options which have not been declared, use `\DeclareOption*{⟨code⟩}`.

Note: there should be no use of `\RequirePackage`, `\DeclareOption`, `\DeclareOption*` or `\ProcessOptions` inside `\DeclareOption` or `\DeclareOption*`.

Possible uses for `\DeclareOption*` include:

`\DeclareOption*{}`
Do nothing. Silently accept unknown options. (This suppresses the usual warnings.)

`\DeclareOption*{\@unkownoptionerror}`
Complain about unknown local options. (The initial setting for package files.)

`\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{⟨pkg-name⟩}}`
Handle the current option by passing it on to the package ⟨*pkg-name*⟩, which will presumably be loaded via `\RequirePackage` later in the file. This is useful for building

'extension' packages, that perhaps handle a couple of new options, but then pass everything else on to an existing package.

```
\DeclareOption*{\InputIfFileExists{xx-\CurrentOption.yyy}%
                {}%
                {\OptionNotUsed}}
```

Handle the option foo by loading the file `xx-foo.yyy` if it exists, otherwise do nothing, but declare that the option was not used. Actually the `\OptionNotUsed` declaration is only needed if this is being used in class files, but does no harm in package files.

3.5 Safe Input Macros

`\InputIfFileExists` `\InputIfFileExists{⟨file⟩}{⟨then⟩}{⟨else⟩}`

Inputs ⟨*file*⟩ if it exists. Immediately before the input, ⟨*then*⟩ is executed. Otherwise ⟨*else*⟩ is executed.

`\IfFileExists` As above, but does not input the file.

One thing you might like to put in the ⟨*else*⟩ clause is

`\@missingfileerror` This starts an interactive request for a filename, supplying default extensions. Just hitting return causes the whole input to be skipped and entering `x` quits the current run,

`\input` This has been redefined from the LaTeX2.09 definition, in terms of the new commands `\InputIfFileExists` and `\@missingfileerror`.

`\listfiles` Giving this declaration in the preamble causes a list of all files input via the 'safe input' commands to be listed at the end. Any strings specified in the optional argument to `\ProvidesPackage` are listed alongside the file name. So files in standard (and other non-standard) distributions can put informative strings in this argument.

4 Implementation

```
⟨*2ekernel⟩
```

`\if@compatibility` The flag for compatibility mode.

```
\newif\if@compatibility
```

(*End of definition for* `\if@compatibility`.)

`\@documentclasshook` This legacy hook is called after the first `\documentclass` command. It is *not* integrated with the new 2020 hook management system! By default this checks to see if `\@normalsize` is undefined, and if so, sets it to `\normalsize`.

```
\def\@documentclasshook{%
   \ifx\@normalsize\@undefined
      \let\@normalsize\normalsize
   \fi
}
```

(*End of definition for* `\@documentclasshook`.)

`\@declaredoptions` This list is automatically built by `\DeclareOption`. It is the list of options (separated by commas) declared in the class or package file and it defines the order in which the corresponding `\ds@`⟨*option*⟩ commands are executed. All local ⟨*option*⟩s which are not declared will be processed in the order defined by the optional argument of `\documentclass` or `\usepackage`.

```
\let\@declaredoptions\@empty
```

(*End of definition for* `\@declaredoptions`.)

`\@classoptionslist` List of options of the main class.

```
\let\@classoptionslist\relax
%\@onlypreamble\@classoptionslist
```

(*End of definition for* `\@classoptionslist`.)

`\@raw@classoptionslist` List of options of the main class (unprocessed).

```
\let\@raw@classoptionslist\relax
```

(*End of definition for* `\@raw@classoptionslist`.)

`\@unusedoptionlist` List of options of the main class that haven't been declared or loaded as class option files.

```
\let\@unusedoptionlist\@empty
%\@onlypreamble\@unusedoptionlist
```

(*End of definition for* `\@unusedoptionlist`.)

`\CurrentOption` Name of current package or option.

```
\let\CurrentOption\@empty
```

(*End of definition for* `\CurrentOption`.)

`\@currpath` Path to the current file if explicitly given.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\@currpath}%
⟨latexrelease⟩   {Add \@currpath}%
\let\@currpath\@empty
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@currpath}%
⟨latexrelease⟩   {Add \@currpath}%
⟨latexrelease⟩\let\@currpath\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
```

(*End of definition for* `\@currpath`.)

`\@currname` Name of current package or option.

```
\let\@currname\@empty
```

(*End of definition for* `\@currname`.)

`\@currext` The current file extension.

```
\global\let\@currext=\@empty
```

(*End of definition for* `\@currext`.)

`\@clsextension` `\@pkgextension` The two possible values of `\@currext`.

```
\def\@clsextension{cls}
\def\@pkgextension{sty}
```

(*End of definition for* `\@clsextension` *and* `\@pkgextension`.)

`\@pushfilename` `\@popfilename` `\@currnamestack`

Commands to push and pop the file name and extension.

#1 current name.
#2 current extension.
#3 current catcode of `@`.
#4 Rest of the stack.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\@pushfilename}%
⟨latexrelease⟩  {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
\def\@pushfilename{%
```

The push and pop macros are injected in `\@pushfilename` and `\@popfilename` so that they correctly keep track of the hook labels.

This needs cleanup with the expl3 interfaces also playing here, e.g., `\@expl@push@filename@@` needs cleanup and (and should probably not have this name either).

```
  \@expl@push@filename@@
  \xdef\@currnamestack{%
    {\@currname}%
    {\@currext}%
    {\the\catcode`\@}%
    \@currnamestack}%
```

Temporarily add a stack for `\@currpath` here. This should be integrated in the main file stack eventually, but other packages rely on `\@currnamestack` having three elements per file, so that isn't a trivial change. The prefix `\@kernel@...` hopefully discourages people from using it.

```
  \xdef\@kernel@currpathstack{%
    {\@currpath}%
    \@kernel@currpathstack}%
  \@expl@push@filename@aux@@}
⟨latexrelease⟩\EndIncludeInRelease
```

The following version of `\@pushfilename` didn't formally exist in this file, but in the 2020/02/02 release, expl3 was preloaded and it patched `\@pushfilename` (and `\@popfilename`) by adding some hooks in there. But rolling back to 2020/02/02, expl3 doesn't patch these macros again, so rolling back has to take those hooks into account. Same goes for `\@popfilename`.

```
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}{\@pushfilename}%
⟨latexrelease⟩  {Add \@expl@push@filename@@}%
⟨latexrelease⟩\def\@pushfilename{%
⟨latexrelease⟩  \@expl@push@filename@@
⟨latexrelease⟩  \xdef\@currnamestack{%
⟨latexrelease⟩    {\@currname}%
⟨latexrelease⟩    {\@currext}%
⟨latexrelease⟩    {\the\catcode`\@}%
⟨latexrelease⟩    \@currnamestack}%
⟨latexrelease⟩    \@expl@push@filename@aux@@}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
```

When we roll back from a release that has expl3 preloaded, the definitions of \@pushfilename and \@popfilename can't be completely rolled back otherwise expl3-based packages won't have the automatic \ExplSyntaxOff at the end. Here and below for \@popfilename, we don't roll back all the way through if coming from LaTeX > 2020 − 02 − 02.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@pushfilename}%
⟨latexrelease⟩  {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
⟨latexrelease⟩\ifnum\sourceLaTeXdate<20200202\relax
⟨latexrelease⟩  \GenericInfo{}{Defining 00-00-00\string\@pushfilename.}
⟨latexrelease⟩\def\@pushfilename{%
⟨latexrelease⟩  \xdef\@currnamestack{%
⟨latexrelease⟩    {\@currname}%
⟨latexrelease⟩    {\@currext}%
⟨latexrelease⟩    {\the\catcode`\@}%
⟨latexrelease⟩    \@currnamestack}}
⟨latexrelease⟩\else
⟨latexrelease⟩  \GenericInfo{}{Defining 2020-02-02\string\@pushfilename.}
⟨latexrelease⟩\def\@pushfilename{%
⟨latexrelease⟩  \@expl@push@filename@@
⟨latexrelease⟩  \xdef\@currnamestack{%
⟨latexrelease⟩    {\@currname}%
⟨latexrelease⟩    {\@currext}%
⟨latexrelease⟩    {\the\catcode`\@}%
⟨latexrelease⟩    \@currnamestack}%
⟨latexrelease⟩    \@expl@push@filename@aux@@}
⟨latexrelease⟩\fi
⟨latexrelease⟩\EndIncludeInRelease
\@onlypreamble\@pushfilename
```

```
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\@popfilename}%
⟨latexrelease⟩  {Add \@expl@pop@filename@@}%
\def\@popfilename{\@expl@@@hook@curr@name@pop@@
  \expandafter\@p@pfilename\@currnamestack\@nil
```

Same for popping:

```
  \expandafter\@p@pfilepath\@kernel@currpathstack\@nil
  \@expl@pop@filename@@}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/02/02}{\@popfilename}%
⟨latexrelease⟩  {Add \@expl@push@filename@@}%
⟨latexrelease⟩\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil
⟨latexrelease⟩  \@expl@pop@filename@@}
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@popfilename}%
⟨latexrelease⟩  {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
⟨latexrelease⟩\ifnum\sourceLaTeXdate<20200202\relax
⟨latexrelease⟩  \GenericInfo{}{Defining 00-00-00\string\@popfilename.}
⟨latexrelease⟩\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil}
⟨latexrelease⟩\else
⟨latexrelease⟩  \GenericInfo{}{Defining 2020-02-02\string\@popfilename.}
```

```
⟨latexrelease⟩\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil
⟨latexrelease⟩   \@expl@pop@filename@@}
⟨latexrelease⟩\fi
⟨latexrelease⟩\EndIncludeInRelease
\@onlypreamble\@popfilename
```

```
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
```

```
\def\@p@pfilename#1#2#3#4\@nil{%
  \gdef\@currname{#1}%
  \gdef\@currext{#2}%
  \catcode`\@#3\relax
  \gdef\@currnamestack{#4}}
\@onlypreamble\@p@pfilename
```

```
\gdef\@currnamestack{}
\@onlypreamble\@currnamestack
```

(*End of definition for* `\@pushfilename`*,* `\@popfilename`*, and* `\@currnamestack`*.*)

`\@kernel@currpathstack` Path to the current file if explicitly given. The auxiliary is needed here to insert a `\@empty` to prevent the loss of braces.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\@kernel@currpathstack}%
⟨latexrelease⟩   {Add \@kernel@currpathstack}%
```

If rolling backwards to this release, `\@kernel@currpathstack` will be defined, so the `\gdef` line should not be executed, thus the `\@gobblethree` will take it out, so the satck isn't touched.

```
⟨latexrelease⟩\@ifundefined{@kernel@currpathstack}{}{\@gobblethree}
\gdef\@kernel@currpathstack{}%
```

If rolling forward to this release, then the `\gdef` line above will define the path stack to be empty (which it can't be, inside a file), so the code below will traverse the `\@currnamestack`, and add as many empty items to `\@kernel@currpathstack` as there are items in `\@currnamestack`, so both are back in sync. Most of the time latexrelease is loaded on top-level, so only one item is needed, but platexrelease loads it internally, so the more complicated loop is needed.

```
⟨latexrelease⟩\ifx\@kernel@currpathstack\@empty
⟨latexrelease⟩   \def\reserved@a#1#2#3{%
⟨latexrelease⟩     \ifx\relax#3\else
⟨latexrelease⟩       \g@addto@macro\@kernel@currpathstack{{}}%
⟨latexrelease⟩       \expandafter\reserved@a
⟨latexrelease⟩     \fi}%
⟨latexrelease⟩   \expandafter\reserved@a\@currnamestack{}{}{\relax}%
⟨latexrelease⟩\fi
\def\@p@pfilepath#1{%
  \gdef\@currpath{#1}\@p@pfilepath@aux\@empty}
\def\@p@pfilepath@aux#1\@nil{%
  \xdef\@kernel@currpathstack{#1}}
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@kernel@currpathstack}%
```

```
⟨latexrelease⟩  {Add \@kernel@currpathstack}%
⟨latexrelease⟩\let\@kernel@currpathstack\@undefined
⟨latexrelease⟩\let\@p@pfilepath\@undefined
⟨latexrelease⟩\let\@p@pfilepath@aux\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
```

(*End of definition for* `\@kernel@currpathstack`.)

`\@ptionlist` Returns the option list of the file.

```
\def\@ptionlist#1{%
  \@ifundefined{opt@#1}\@empty{\csname opt@#1\endcsname}}
%\@onlypreamble\@ptionlist
```

(*End of definition for* `\@ptionlist`.)

`\@ifpackageloaded` `\@ifclassloaded`

`\@ifpackageloaded{`⟨*name*⟩`}` Checks to see whether a file has been loaded.

```
\def\@ifpackageloaded{\@ifl@aded\@pkgextension}
\def\@ifclassloaded{\@ifl@aded\@clsextension}
```

```
\def\@ifl@aded#1#2{%
  \expandafter\ifx\csname ver@#2.#1\endcsname\relax
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi}
```

(*End of definition for* `\@ifpackageloaded` *and* `\@ifclassloaded`.)

`\@ifpackagelater` `\@ifclasslater`

`\@ifpackagelater{`⟨*name*⟩`}{YYYY/MM/DD}{`⟨*true code*⟩`}{`⟨*false code*⟩`}` Checks that the package loaded is more recent or equal to the given date. A better name for it would therefore been `\@ifpackagelaterorequal` but it is in use for more than 30 years, so ...

```
\def\@ifpackagelater{\@ifl@ter\@pkgextension}
\def\@ifclasslater{\@ifl@ter\@clsextension}
```

(*End of definition for* `\@ifpackagelater` *and* `\@ifclasslater`.)

`\IfPackageAtLeastTF` `\IfClassAtLeastTF` `\IfFileAtLeastTF` `\IfFormatAtLeastTF`

`\IfFormatAtLeastTF{YYYY/MM/DD}{`⟨*true code*⟩`}{`⟨*false code*⟩`}` Test if the format is later or equal to the given date.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\IfFormatAtLeastTF}{Test format date}%
\def\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
\let\IfPackageAtLeastTF\@ifpackagelater
\let\IfClassAtLeastTF\@ifclasslater
\def\IfFileAtLeastTF#1{\expandafter\@ifl@t@r\csname ver@#1\endcsname}
```

For rollback pretend it was available since the beginning of dawn.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\IfFormatAtLeastTF}{Test format date}%
⟨latexrelease⟩\def\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
```

```
⟨latexrelease⟩\let\IfPackageAtLeastTF\@ifpackagelater
⟨latexrelease⟩\let\IfClassAtLeastTF\@ifclasslater
⟨latexrelease⟩\def\IfFileAtLeastTF#1{\expandafter\@ifl@t@r\csname ver@#1\endcsname}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\IfPackageAtLeastTF` *and others.*)

`\@ifl@ter`

```
\def\@ifl@ter#1#2{%
  \expandafter\@ifl@t@r
    \csname ver@#2.#1\endcsname}
⟨/2ekernel⟩
```

This internal macro is also used in `\NeedsTeXFormat`.

```
⟨latexrelease⟩\IncludeInRelease{2018/04/01}%
⟨latexrelease⟩                 {\@ifl@t@r}{Guard against bad input}%
⟨*2ekernel | latexrelease⟩
\def\@ifl@t@r#1#2{%
  \ifnum\expandafter\@parse@version@#1//00\@nil<%
        \expandafter\@parse@version@#2//00\@nil
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi}
\def\@parse@version@#1{\@parse@version0#1}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@ifl@t@r}{Guard against bad input}%
⟨latexrelease⟩\def\@ifl@t@r#1#2{%
⟨latexrelease⟩  \ifnum\expandafter\@parse@version#1//00\@nil<%
⟨latexrelease⟩        \expandafter\@parse@version#2//00\@nil
⟨latexrelease⟩    \expandafter\@secondoftwo
⟨latexrelease⟩  \else
⟨latexrelease⟩    \expandafter\@firstoftwo
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\let\@parse@version@\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@ifl@ter`.)

```
⟨/2ekernel⟩
⟨*2ekernel | latexreleasefirst⟩
\def\@parse@version#1/#2/#3#4#5\@nil{%
\@parse@version@dash#1-#2-#3#4\@nil
}
```

The `\if` test here ensures that an argument with no `/` or `-` produces 0 (actually 00).

```
\def\@parse@version@dash#1-#2-#3#4#5\@nil{%
  \if\relax#2\relax\else#1\fi#2#3#4 }
⟨/2ekernel | latexreleasefirst⟩
⟨*2ekernel⟩
```

\@ifpackagewith
\@ifclasswith

\@ifpackagewith{⟨*name*⟩}{⟨*option-list*⟩} Checks that ⟨*option-list*⟩ is a subset of the options **with** which ⟨*name*⟩ was loaded.

```
\def\@ifpackagewith{\@if@ptions\@pkgextension}
\def\@ifclasswith{\@if@ptions\@clsextension}
```

```
\def\@if@ptions#1#2{%
  \@expandtwoargs\@if@pti@ns{\@ptionlist{#2.#1}}}
```

Probably shouldn't use \CurrentOption here... (changed to \reserved@b.)

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease⟩                  {\@if@pti@ns}{Spaces in option clash check}%
⟨*2ekernel | latexrelease⟩
\def\@if@pti@ns#1#2{%
 \let\reserved@a\@firstoftwo
```

```
 \edef\reserved@b{\zap@space#2 \@empty}%
 \@for\reserved@b:=\reserved@b\do{%
   \ifx\reserved@b\@empty
   \else
     \expandafter\in@\expandafter{\expandafter,\reserved@b,}{,#1,}%
     \ifin@
     \else
       \let\reserved@a\@secondoftwo
     \fi
   \fi
 }%
 \reserved@a}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@if@pti@ns}{Spaces in option clash check}%
⟨latexrelease⟩\def\@if@pti@ns#1#2{%
⟨latexrelease⟩ \let\reserved@a\@firstoftwo
⟨latexrelease⟩ \@for\reserved@b:=#2\do{%
⟨latexrelease⟩  \ifx\reserved@b\@empty
⟨latexrelease⟩   \else
⟨latexrelease⟩   \expandafter\in@\expandafter
⟨latexrelease⟩                   {\expandafter,\reserved@b,}{,#1,}%
⟨latexrelease⟩    \ifin@
⟨latexrelease⟩    \else
⟨latexrelease⟩     \let\reserved@a\@secondoftwo
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩ }%
⟨latexrelease⟩ \reserved@a}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \@ifpackagewith *and* \@ifclasswith.)

\IfPackageLoadedTF
\IfPackageLoadedWithOptionsTF
\IfClassLoadedTF
\IfClassLoadedWithOptionsTF

More public names for the commands already available for a long time.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/06/01}%
```

```
⟨latexrelease⟩                 {\IfPackageLoadedTF}{Test package loading}%
\let \IfPackageLoadedTF            \@ifpackageloaded
\let \IfClassLoadedTF              \@ifclassloaded
\let \IfPackageLoadedWithOptionsTF \@ifpackagewith
\let \IfClassLoadedWithOptionsTF   \@ifclasswith
```

For rollback/rollforward pretend everything was available since the beginning of dawn.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\IfPackageLoadedTF}{Test package loading}%
⟨latexrelease⟩
⟨latexrelease⟩\let \IfPackageLoadedTF            \@ifpackageloaded
⟨latexrelease⟩\let \IfClassLoadedTF              \@ifclassloaded
⟨latexrelease⟩\let \IfPackageLoadedWithOptionsTF \@ifpackagewith
⟨latexrelease⟩\let \IfClassLoadedWithOptionsTF   \@ifclasswith
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\IfPackageLoadedTF` *and others.*)

`\IfPackageLoadedT`
`\IfPackageLoadedF`
`\IfPackageAtLeastT`
`\IfPackageAtLeastF`
`\IfClassAtLeastT`
`\IfClassAtLeastF`
`\IfFileAtLeastT`
`\IfFileAtLeastF`
`\IfFormatAtLeastT`
`\IfFormatAtLeastF`
`\IfPackageLoadedWithOptionsT`
`\IfPackageLoadedWithOptionsF`
`\IfClassLoadedT`
`\IfClassLoadedF`
`\IfClassLoadedWithOptionsF`
`\IfClassLoadedWithOptionsTF`

A few more conditionals for convenience

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/06/01}%
⟨latexrelease⟩                 {\IfPackageLoadedT}{More conditionals}%
\def\IfPackageLoadedT #1#2{\IfPackageLoadedTF{#1}{#2}{}}
\def\IfPackageLoadedF   #1{\IfPackageLoadedTF{#1}{}}
\def\IfClassLoadedT   #1#2{\IfClassLoadedTF{#1}{#2}{}}
\def\IfClassLoadedF     #1{\IfClassLoadedTF{#1}{}}
\def\IfPackageAtLeastT#1#2#3{\IfPackageAtLeastTF{#1}{#2}{#3}{}}
\def\IfPackageAtLeastF  #1#2{\IfPackageAtLeastTF{#1}{#2}{}}
\def\IfClassAtLeastT  #1#2#3{\IfClassAtLeastTF{#1}{#2}{#3}{}}
\def\IfClassAtLeastF    #1#2{\IfClassAtLeastTF{#1}{#2}{}}
\def\IfFileAtLeastT   #1#2#3{\IfFileAtLeastTF{#1}{#2}{#3}{}}
\def\IfFileAtLeastF     #1#2{\IfFileAtLeastTF{#1}{#2}{}}
\def\IfFormatAtLeastT   #1#2{\IfFormatAtLeastTF{#1}{#2}{}}
\def\IfFormatAtLeastF     #1{\IfFormatAtLeastTF{#1}{}}
\def\IfPackageLoadedWithOptionsT #1#2#3{\IfPackageLoadedWithOptionsTF{#1}{#2}{#3}{}}
\def\IfPackageLoadedWithOptionsF   #1#2{\IfPackageLoadedWithOptionsTF{#1}{#2}{}}
\def\IfClassLoadedWithOptionsT #1#2#3{\IfClassLoadedWithOptionsTF{#1}{#2}{#3}{}}
\def\IfClassLoadedWithOptionsF   #1#2{\IfClassLoadedWithOptionsTF{#1}{#2}{}}
```

These three commands haven't been there at all in the past.

```
\def\IfFileLoadedTF#1{%
  \expandafter\ifx\csname ver@#1\endcsname\relax
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi}
\def\IfFileLoadedT#1#2{\IfFileLoadedTF{#1}{#2}{}}
\def\IfFileLoadedF  #1{\IfFileLoadedTF{#1}{}}
```


`\IfFileLoadedTF`
`\IfFileLoadedT`
`\IfFileLoadedF`

For rollback/rollforward pretend everything was available since the beginning of dawn.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\IfPackageLoadedT}{More conditionals}%
⟨latexrelease⟩
⟨latexrelease⟩\def\IfPackageLoadedT #1#2{\IfPackageLoadedTF{#1}{#2}{}}
⟨latexrelease⟩\def\IfPackageLoadedF   #1{\IfPackageLoadedTF{#1}{}}
⟨latexrelease⟩\def\IfClassLoadedT   #1#2{\IfClassLoadedTF{#1}{#2}{}}
⟨latexrelease⟩\def\IfClassLoadedF     #1{\IfClassLoadedTF{#1}{}}
⟨latexrelease⟩\def\IfPackageAtLeastT#1#2#3{\IfPackageAtLeastTF{#1}{#2}{#3}{}}
⟨latexrelease⟩\def\IfPackageAtLeastF  #1#2{\IfPackageAtLeastTF{#1}{#2}{}}
⟨latexrelease⟩\def\IfClassAtLeastT  #1#2#3{\IfClassAtLeastTF{#1}{#2}{#3}{}}
⟨latexrelease⟩\def\IfClassAtLeastF    #1#2{\IfClassAtLeastTF{#1}{#2}{}}
⟨latexrelease⟩\def\IfFileAtLeastT   #1#2#3{\IfFileAtLeastTF{#1}{#2}{#3}{}}
⟨latexrelease⟩\def\IfFileAtLeastF     #1#2{\IfFileAtLeastTF{#1}{#2}{}}
⟨latexrelease⟩\def\IfFormatAtLeastT   #1#2{\IfFormatAtLeastTF{#1}{#2}{}}
⟨latexrelease⟩\def\IfFormatAtLeastF     #1{\IfFormatAtLeastTF{#1}{}}
⟨latexrelease⟩\def\IfPackageLoadedWithOptionsT #1#2#3{\IfPackageLoadedWithOptionsTF{#1}{#2}{#3}{
⟨latexrelease⟩\def\IfPackageLoadedWithOptionsF   #1#2{\IfPackageLoadedWithOptionsTF{#1}{#2}{}}
⟨latexrelease⟩\def\IfClassLoadedWithOptionsT #1#2#3{\IfClassLoadedWithOptionsTF{#1}{#2}{#3}{}}
⟨latexrelease⟩\def\IfClassLoadedWithOptionsF   #1#2{\IfClassLoadedWithOptionsTF{#1}{#2}{}}
⟨latexrelease⟩
⟨latexrelease⟩\def\IfFileLoadedTF#1{%
⟨latexrelease⟩  \expandafter\ifx\csname ver@#1\endcsname\relax
⟨latexrelease⟩    \expandafter\@secondoftwo
⟨latexrelease⟩  \else
⟨latexrelease⟩    \expandafter\@firstoftwo
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\def\IfFileLoadedT#1#2{\IfFileLoadedTF{#1}{#2}{}}
⟨latexrelease⟩\def\IfFileLoadedF  #1{\IfFileLoadedTF{#1}{}}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\IfPackageLoadedT` *and others.*)

`\ProvidesPackage` Checks that the current filename is correct, and defines `\ver@filename`.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩  {\ProvidesPackage}{Check name with \strcmp}%
⟨*2ekernel | latexrelease⟩
\def\ProvidesPackage#1{%
  \xdef\@gtempa{#1}%
```

Here `\@currpath` is explicitly added to the file name to report when a package or class is loaded using an explicit path. Loading using a path in the argument is supported but not encouraged.

```
  \@expandtwoargs\@expl@str@if@eq@@nnTF
      {\@gtempa}{\@currpath\@currname}{}{%
    \@latex@warning@no@line{You have requested
      \@cls@pkg\space`\@currpath\@currname',\MessageBreak
       but the \@cls@pkg\space provides `#1'}%
    }%
```

```
  \@ifnextchar[\@pr@videpackage{\@pr@videpackage[]}%]
\@onlypreamble\ProvidesPackage
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
%
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\ProvidesPackage}{Undo: check name with \strcmp}%
⟨latexrelease⟩\def\ProvidesPackage#1{%
⟨latexrelease⟩  \xdef\@gtempa{#1}%
⟨latexrelease⟩  \ifx\@gtempa\@currname\else
⟨latexrelease⟩    \@latex@warning@no@line{You have requested
⟨latexrelease⟩      \@cls@pkg\space`\@currname',\MessageBreak
⟨latexrelease⟩       but the \@cls@pkg\space provides `#1'}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \@ifnextchar[\@pr@videpackage{\@pr@videpackage[]}%]
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \ProvidesPackage.)

\@pr@videpackage This is the helper command for \ProvidesPackage. It tries to be cautious when handling the identification string in case it contains UTF-8 characters.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                 {\@pr@videpackage}{Allow for package substitution}%
\def\@pr@videpackage[#1]{%
  \expandafter\protected@xdef                %     <-- protected...
     \csname ver@\@currname.\@currext\endcsname{#1}% Loaded package
  \expandafter\let
    \csname ver@\@currpkg@reqd\expandafter\endcsname % Requested package
    \csname ver@\@currname.\@currext\endcsname
  \ifx\@currext\@clsextension
    \typeout{Document Class: \@gtempa\space#1}%
  \else
    \protected@wlog{Package: \@gtempa\space#1}%   <--- protected
  \fi}
```

(*End of definition for* \@pr@videpackage.)

\protected@wlog This is like plain TEX's \wlog but gracefully handles protected commands.

```
\long\def\protected@wlog#1{\begingroup
  \set@display@protect
  \immediate \write \m@ne {#1}\endgroup }
```

(*End of definition for* \protected@wlog.)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
⟨latexrelease⟩                 {\@pr@videpackage}{Protection for package info}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@pr@videpackage[#1]{%
⟨latexrelease⟩  \expandafter\protected@xdef                %     <-- protected...
⟨latexrelease⟩     \csname ver@\@currname.\@currext\endcsname{#1}%
```

```
⟨latexrelease⟩\ifx\@currext\@clsextension
⟨latexrelease⟩    \typeout{Document Class: \@gtempa\space#1}%
⟨latexrelease⟩  \else
⟨latexrelease⟩    \protected@wlog{Package: \@gtempa\space#1}%   <--- protected
⟨latexrelease⟩  \fi}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                {\@pr@videpackage}{Protection for package info}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@pr@videpackage[#1]{%
⟨latexrelease⟩  \expandafter\xdef\csname ver@\@currname.\@currext\endcsname{#1}%
⟨latexrelease⟩  \ifx\@currext\@clsextension
⟨latexrelease⟩    \typeout{Document Class: \@gtempa\space#1}%
⟨latexrelease⟩  \else
⟨latexrelease⟩    \wlog{Package: \@gtempa\space#1}%
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\let\protected@wlog\@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
\@onlypreamble\@pr@videpackage
```

`\ProvidesClass` Like `\ProvidesPackage`, but for classes. This needs a dummy latexrelease block to copy the definition of `\ProvidesPackage` as it changes across releases.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\ProvidesClass}{Track \ProvidesPackage}%
⟨*2ekernel | latexrelease⟩
\let\ProvidesClass\ProvidesPackage
\@onlypreamble\ProvidesClass
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\ProvidesClass`.)

`\ProvidesFile` Like `\ProvidesPackage`, but for arbitrary files. Do not apply `\@onlypreamble` to these, as we may want to label files input during the document.

`\@providesfile`

```
\def\ProvidesFile#1{%
  \begingroup
    \catcode`\ 10 %
    \ifnum \endlinechar<256 %
      \ifnum \endlinechar>\m@ne
        \catcode\endlinechar 10 %
      \fi
    \fi
    \@makeother\/%
    \@makeother\&%
    \kernel@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}}
```

During initex a special version of `\@providesfile` is used. The real definition is installed right at the end, in `ltfinal.dtx`.

```
def\@providesfile#1[#2]{%
   \wlog{File: #1 #2}%
   \expandafter\xdef\csname ver@#1\endcsname{#2}%
 \endgroup}
```

(*End of definition for* `\ProvidesFile` *and* `\@providesfile`.)

`\PassOptionsToPackage`
`\PassOptionsToClass`

If the package has been loaded, we check that it was first loaded with the options. Otherwise we add the option list to that of the package.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                  {\@pass@ptions}{Raw option lists}%
⟨*2ekernel | latexrelease⟩
\def\@pass@ptions#1#2#3{%
  \@expl@@@filehook@set@curr@file@@nNN
    {\@expl@@@filehook@resolve@file@subst@@w #3.#1\@nil}%
      \reserved@a\reserved@b
  \@expl@@@filehook@clear@replacement@flag@@

  \expandafter\protected@xdef\csname opt@\reserved@a\endcsname{%
    \@ifundefined{opt@\reserved@a}\@empty
      {\csname opt@\reserved@a\endcsname,}%
    \zap@space#2 \@empty}%

  \expandafter\let
    \csname opt@#3.#1\expandafter\endcsname
    \csname opt@\reserved@a\endcsname
```

Extend raw option list

```
  \@ifundefined{@raw@opt@#3.#1}%
    {\expandafter\gdef\csname @raw@opt@#3.#1\expandafter\endcsname
              \expandafter{#2}}%
    {\expandafter\g@addto@macro\csname @raw@opt@#3.#1\expandafter\endcsname
              \expandafter{\expandafter,#2}}%
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease

⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\@pass@ptions}
⟨latexrelease⟩  {Add file replacement in \@pass@ptions}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@pass@ptions#1#2#3{%
⟨latexrelease⟩  \@expl@@@filehook@set@curr@file@@nNN
⟨latexrelease⟩    {\@expl@@@filehook@resolve@file@subst@@w #3.#1\@nil}%
⟨latexrelease⟩      \reserved@a\reserved@b
⟨latexrelease⟩  \@expl@@@filehook@clear@replacement@flag@@
⟨latexrelease⟩  \expandafter\xdef\csname opt@\reserved@a\endcsname{%
⟨latexrelease⟩    \@ifundefined{opt@\reserved@a}\@empty
⟨latexrelease⟩      {\csname opt@\reserved@a\endcsname,}%
⟨latexrelease⟩    \zap@space#2 \@empty}%
⟨latexrelease⟩  \expandafter\let
⟨latexrelease⟩    \csname opt@#3.#1\expandafter\endcsname
⟨latexrelease⟩    \csname opt@\reserved@a\endcsname}
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@pass@ptions}
⟨latexrelease⟩  {\@pass@ptions}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@pass@ptions#1#2#3{%
⟨latexrelease⟩  \expandafter\xdef\csname opt@#3.#1\endcsname{%
⟨latexrelease⟩    \@ifundefined{opt@#3.#1}\@empty
⟨latexrelease⟩      {\csname opt@#3.#1\endcsname,}%
⟨latexrelease⟩    \zap@space#2 \@empty}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\@pass@ptions
```

```
\def\PassOptionsToPackage{\@pass@ptions\@pkgextension}
\def\PassOptionsToClass{\@pass@ptions\@clsextension}
\@onlypreamble\PassOptionsToPackage
\@onlypreamble\PassOptionsToClass
```

(*End of definition for* `\PassOptionsToPackage` *and* `\PassOptionsToClass`.)

`\DeclareOption`
`\DeclareOption*`

Adds an option as a `\ds@` command, or the default `\default@ds` command.

```
\def\DeclareOption{%
  \let\@fileswith@pti@ns\@badrequireerror
  \@ifstar\@defdefault@ds\@declareoption}
\long\def\@declareoption#1#2{%
   \xdef\@declaredoptions{\@declaredoptions,#1}%
   \toks@{#2}%
   \expandafter\edef\csname ds@#1\endcsname{\the\toks@}}
\long\def\@defdefault@ds#1{%
  \toks@{#1}%
  \edef\default@ds{\the\toks@}}
\@onlypreamble\DeclareOption
\@onlypreamble\@declareoption
\@onlypreamble\@defdefault@ds
```

(*End of definition for* `\DeclareOption` *and* `\DeclareOption*`.)

`\OptionNotUsed`
`\@remove@eq@value`

If we are in a class file, add `\CurrentOption` to the list of unused options. Otherwise, in a package file do nothing.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                  {\OptionNotUsed}{filter unused option list}%
⟨*2ekernel | latexrelease⟩
\ExplSyntaxOn
\def\@remove@eq@value#1=#2\@nil{\tl_trim_spaces:n{#1}}
\ExplSyntaxOff
```

```
\def\OptionNotUsed{%
  \ifx\@currext\@clsextension
    \xdef\@unusedoptionlist{%
      \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
      \expandafter\@remove@eq@value\CurrentOption=\@nil}%
  \fi}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
```

```
⟨latexrelease⟩                  {\OptionNotUsed}{filter unused option list}%
⟨latexrelease⟩\let\@remove@eq@value\@undefined
```

```
⟨latexrelease⟩\def\OptionNotUsed{%
⟨latexrelease⟩   \ifx\@currext\@clsextension
⟨latexrelease⟩     \xdef\@unusedoptionlist{%
⟨latexrelease⟩       \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
⟨latexrelease⟩       \CurrentOption}%
⟨latexrelease⟩   \fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\OptionNotUsed
```

(*End of definition for* `\OptionNotUsed` *and* `\@remove@eq@value`.)

`\default@ds` The default option code. Set by `\@onefilewithoptions` to either `\OptionNotUsed` for classes, or `\@unknownoptionerror` for packages. This may be reset in either case with `\DeclareOption*`.

```
% \let\default@ds\OptionNotUsed
```

(*End of definition for* `\default@ds`.)

`\ProcessOptions` `\ProcessOptions*` `\ProcessOptions` calls `\ds@option` for each known package option, then calls `\default@ds` for each option on the local options list. Finally resets all the declared options to `\relax`. The empty option does nothing, this has to be reset on the off chance it's set to `\relax` if an empty element gets into the `\@declaredoptions` list.

The star form is similar but executes options given in the order specified in the document, not the order they are declared in the file. In the case of packages, global options are executed before local ones.

```
\def\ProcessOptions{%
  \let\ds@\@empty
  \protected@edef\@curroptions{\@ptionlist{\@currname.\@currext}}%
  \@ifstar\@xprocess@ptions\@process@ptions}
\@onlypreamble\ProcessOptions
```

```
\def\@process@ptions{%
  \@for\CurrentOption:=\@declaredoptions\do{%
    \ifx\CurrentOption\@empty\else
      \@expandtwoargs\in@{,\CurrentOption,}{%
         ,\ifx\@currext\@clsextension\else\@classoptionslist,\fi
         \@curroptions,}%
      \ifin@
        \@use@ption
        \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
      \fi
    \fi}%
  \@process@pti@ns}
\@onlypreamble\@process@ptions
```

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                  {\@xprocess@ptions}{safer @xprocess@ptions}%
⟨*2ekernel | latexrelease⟩
\def\@xprocess@ptions{%
```

```
  \ifx\@currext\@clsextension\else
   \ifx\@classoptionslist\relax\else
    \@for\CurrentOption:=\@classoptionslist\do{%
      \ifx\CurrentOption\@empty\else
        \@ifundefined{ds@\detokenize\expandafter{\CurrentOption}}{}{%
          \@use@ption
          \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
        }%
      \fi}%
    \fi
  \fi
  \@process@pti@ns}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@xprocess@ptions}{safer @xprocess@ptions}%
⟨latexrelease⟩\let\@remove@eq@value\@undefined
⟨latexrelease⟩\def\@xprocess@ptions{%
⟨latexrelease⟩  \ifx\@currext\@clsextension\else
⟨latexrelease⟩    \@for\CurrentOption:=\@classoptionslist\do{%
⟨latexrelease⟩      \ifx\CurrentOption\@empty\else
⟨latexrelease⟩        \@expandtwoargs\in@{,\CurrentOption,}{,\@declaredoptions,}%
⟨latexrelease⟩        \ifin@
⟨latexrelease⟩          \@use@ption
⟨latexrelease⟩          \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
⟨latexrelease⟩        \fi
⟨latexrelease⟩      \fi}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \@process@pti@ns}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
\@onlypreamble\@xprocess@ptions
```

The common part of \ProcessOptions and \ProcessOptions*.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\@process@pti@ns}{Unused options issue}%
\def\@process@pti@ns{%
  \@for\CurrentOption:=\@curroptions\do{%
    \@ifundefined{ds@\detokenize\expandafter{\CurrentOption}}%
      {\@use@ption
       \default@ds}%
```

There should not be any non-empty definition of \CurrentOption at this point, as all the declared options were executed earlier. This is for compatibility with 2.09 styles which use \def\ds@... directly, and so have options which do not appear in \@declaredoptions.

```
      \@use@ption}%
```

Clear all the definitions for option code. First set all the declared options to \relax, then reset the 'default' and 'empty' options. and the lst of declared options.

```
  \@for\CurrentOption:=\@declaredoptions\do{%
    \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%
```

```
  \let\CurrentOption\@empty
  \let\@fileswith@pti@ns\@@fileswith@pti@ns
  \AtEndOfPackage{\expandafter\let
                     \csname unprocessedoptions-\@currname.\@currext\endcsname
                     \relax}}
\@onlypreamble\@process@pti@ns
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@process@pti@ns}{Unused options issue}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@process@pti@ns{%
⟨latexrelease⟩  \@for\CurrentOption:=\@curroptions\do{%
⟨latexrelease⟩    \@ifundefined{ds@\CurrentOption}%
⟨latexrelease⟩      {\@use@ption
⟨latexrelease⟩       \default@ds}%
⟨latexrelease⟩      \@use@ption}%
⟨latexrelease⟩  \@for\CurrentOption:=\@declaredoptions\do{%
⟨latexrelease⟩    \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%
⟨latexrelease⟩  \let\CurrentOption\@empty
⟨latexrelease⟩  \let\@fileswith@pti@ns\@@fileswith@pti@ns
⟨latexrelease⟩  \AtEndOfPackage{\let\@unprocessedoptions\relax}}
⟨latexrelease⟩\EndIncludeInRelease
⟨∗2ekernel⟩
```

(*End of definition for* `\ProcessOptions` *and* `\ProcessOptions*`.)

`\@options` `\@options` is a synonym for `\ProcessOptions*` for upward compatibility with LaTeX2.09 style files.

```
\def\@options{\ProcessOptions*}
\@onlypreamble\@options
```

(*End of definition for* `\@options`.)

`\@use@ption` Execute the code for the current option.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                 {\@use@ption}{filter unused option list}%
⟨∗2ekernel | latexrelease⟩
\def\@use@ption{%
  \@expandtwoargs\@removeelement
     {\expandafter\@remove@eq@value\CurrentOption=\@nil}%
  \@unusedoptionlist\@unusedoptionlist
  \csname ds@\detokenize\expandafter{\CurrentOption}\endcsname}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\@use@ption}{filter unused option list}%
⟨latexrelease⟩\def\@use@ption{%
⟨latexrelease⟩  \@expandtwoargs\@removeelement\CurrentOption
⟨latexrelease⟩  \@unusedoptionlist\@unusedoptionlist
⟨latexrelease⟩  \csname ds@\CurrentOption\endcsname}
⟨latexrelease⟩\EndIncludeInRelease
⟨∗2ekernel⟩
```

```
\@onlypreamble\@use@ption
```

(*End of definition for* `\@use@ption`.)

`\ExecuteOptions` `\ExecuteOptions{⟨option-list⟩}` executes the code declared for each option.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease⟩                  {\ExecuteOptions}{Spaces in \ExecuteOptions}%
⟨*2ekernel | latexrelease⟩
\def\ExecuteOptions#1{%
```

Use `\@fortmp` here as it is anyway cleared during `\@for` loop so does not change any existing names.

```
  \edef\@fortmp{\zap@space#1 \@empty}%
  \def\reserved@a##1\@nil{%
    \@for\CurrentOption:=\@fortmp\do
             {\csname ds@\CurrentOption\endcsname}%
    \edef\CurrentOption{##1}}%
  \expandafter\reserved@a\CurrentOption\@nil}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\ExecuteOptions}{Spaces in \ExecuteOptions}%
⟨latexrelease⟩\def\ExecuteOptions#1{%
⟨latexrelease⟩  \def\reserved@a##1\@nil{%
⟨latexrelease⟩    \@for\CurrentOption:=#1\do
⟨latexrelease⟩             {\csname ds@\CurrentOption\endcsname}%
⟨latexrelease⟩    \edef\CurrentOption{##1}}%
⟨latexrelease⟩  \expandafter\reserved@a\CurrentOption\@nil}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\ExecuteOptions
```

(*End of definition for* `\ExecuteOptions`.)

The top-level commands, which just set some parameters then call the internal command, `\@fileswithoptions`.

`\documentclass` The main new-style class declaration.

```
\def\documentclass{%
  \let\documentclass\@twoclasseserror
  \if@compatibility\else\let\usepackage\RequirePackage\fi
  \@fileswithoptions\@clsextension}
\@onlypreamble\documentclass
```

(*End of definition for* `\documentclass`.)

`\documentstyle` 2.09 style class ‘style’ declaration.

```
\def\documentstyle{%
  \makeatletter\input{latex209.def}\makeatother
  \documentclass}
\@onlypreamble\documentstyle
```

(*End of definition for* `\documentstyle`.)

\RequirePackage Load package if not already loaded.

```
\def\RequirePackage{%
  \@fileswithoptions\@pkgextension}
\@onlypreamble\RequirePackage
```

(*End of definition for* \RequirePackage.)

\LoadClass Load class.

```
\def\LoadClass{%
  \ifx\@currext\@pkgextension
     \@latex@error
      {\noexpand\LoadClass in package file}%
      {You may only use \noexpand\LoadClass in a class file.}%
  \fi
  \@fileswithoptions\@clsextension}
\@onlypreamble\LoadClass
```

(*End of definition for* \LoadClass.)

\@loadwithoptions Pass the current option list on to a class or package. #1 is \@*cls-or-pkg*extension, #2 is \RequirePackage or \LoadClass, #3 is the class or package to be loaded.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                 {\@loadwithoptions}{Raw option lists load with options}%
⟨*2ekernel | latexrelease⟩
\def\@loadwithoptions#1#2#3{%
  \expandafter\let\csname opt@#3.#1\expandafter\endcsname
       \csname opt@\@currname.\@currext\endcsname
  \expandafter\let\csname @raw@opt@#3.#1\expandafter\endcsname
       \csname @raw@opt@\@currname.\@currext\endcsname
   #2{#3}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}
⟨latexrelease⟩                 {\@loadwithoptions}{Raw option lists load with options}%
⟨latexrelease⟩\def\@loadwithoptions#1#2#3{%
⟨latexrelease⟩  \expandafter\let\csname opt@#3.#1\expandafter\endcsname
⟨latexrelease⟩       \csname opt@\@currname.\@currext\endcsname
⟨latexrelease⟩   #2{#3}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\@loadwithoptions
```

(*End of definition for* \@loadwithoptions.)

\LoadClassWithOptions Load class '#1' with the current option list.

```
\def\LoadClassWithOptions{%
  \@loadwithoptions\@clsextension\LoadClass}
\@onlypreamble\LoadClassWithOptions
```

(*End of definition for* \LoadClassWithOptions.)

\RequirePackageWithOptions Load package '#1' with the current option list.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\RequirePackageWithOptions}{Unused options issue}%
\def\RequirePackageWithOptions{%
```

The resetting of the unprocessed options is now done on a par package basis.

```
  \AtEndOfPackage{\expandafter\let
                    \csname unprocessedoptions-\@currname.\@currext\endcsname
                    \relax}%
  \@loadwithoptions\@pkgextension\RequirePackage}
\@onlypreamble\RequirePackageWithOptions
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\RequirePackageWithOptions}{Unused options issue}%
⟨latexrelease⟩
⟨latexrelease⟩\def\RequirePackageWithOptions{%
⟨latexrelease⟩  \AtEndOfPackage{\let\@unprocessedoptions\relax}%
⟨latexrelease⟩  \@loadwithoptions\@pkgextension\RequirePackage}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \RequirePackageWithOptions.)

\usepackage To begin with, \usepackage produces an error. This is reset by \documentclass.

```
\def\usepackage#1#{%
  \@latex@error
    {\noexpand \usepackage before \string\documentclass}%
    {\noexpand \usepackage may only appear in the document
      preamble, i.e.,\MessageBreak
      between \noexpand\documentclass and
      \string\begin{document}.}%
  \@gobble}
\@onlypreamble\usepackage
```

(*End of definition for* \usepackage.)

\NeedsTeXFormat Check that the document is running on the correct system.

```
\def\NeedsTeXFormat#1{%
  \def\reserved@a{#1}%
  \ifx\reserved@a\fmtname
    \expandafter\@needsformat
  \else
     \@latex@error{This file needs format '\reserved@a'%
       \MessageBreak but this is '\fmtname'}{%
       The current input file will not be processed
       further,\MessageBreak
       because it was written for some other flavor of
       TeX.\MessageBreak\@ehd}%
```

If the file is not meant to be processed by LaTeX 2_ε we stop inputting it, but we do not end the run. We just end inputting the current file.

```
     \endinput \fi}
\@onlypreamble\NeedsTeXFormat
```

```
\def\@needsformat{%
  \@ifnextchar[%]
    \@needsf@rmat
    {}}
\@onlypreamble\@needsformat
```

```
\def\@needsf@rmat[#1]{%
    \@ifl@t@r\fmtversion{#1}{}%
    {\@latex@warning@no@line
        {You have requested release ‘#1’ of LaTeX,\MessageBreak
         but only release ‘\fmtversion’ is available}}}
\@onlypreamble\@needsf@rmat
```

(*End of definition for* `\NeedsTeXFormat`.)

`\zap@space` `\zap@space foo`⟨*space*⟩`\@empty` removes all spaces from `foo` that are not protected by `{ }` groups.

```
\def\zap@space#1 #2{%
  #1%
  \ifx#2\@empty\else\expandafter\zap@space\fi
  #2}
```

(*End of definition for* `\zap@space`.)

`\@fileswithoptions` The common part of `\documentclass` and `\usepackage`.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2024/06/01}%
⟨latexrelease⟩                  {\@fileswithoptions}{Check Group}%
⟨*2ekernel | latexrelease⟩
\def\@fileswithoptions#1{%
    \ifnum\currentgrouplevel>\z@
     \@latex@error
      {Loading a class or package in a group}%
      {Classes and packages should only be loaded at the top level}%
  \fi
  \@ifnextchar[%]
    {\@fileswith@ptions#1}%
    {\@fileswith@ptions#1[]}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@fileswithoptions}{Check Group}%
⟨latexrelease⟩\def\@fileswithoptions#1{%
⟨latexrelease⟩   \@ifnextchar[%]
⟨latexrelease⟩     {\@fileswith@ptions#1}%
⟨latexrelease⟩     {\@fileswith@ptions#1[]}}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\@onlypreamble\@fileswithoptions
```

```
\def\@fileswith@ptions#1[#2]#3{%
  \@ifnextchar[%]
  {\@fileswith@pti@ns#1[{#2}]#3}%
  {\@fileswith@pti@ns#1[{#2}]#3[]}}
\@onlypreamble\@fileswith@ptions
```

Then we do some work.

First of all, we define the global variables. Then we look to see if the file has already been loaded. If it has, we check that it was first loaded with at least the current options. If it has not, we add the current options to the package options, set the default version to be `0000/00/00`, and load the file if we can find it. Then we check the version number.

Finally, we restore the old file name, reset the default option, and we set the catcode of `@`.

For classes, we can immediately process the file. For other types, `#2` could be a comma separated list, so loop through, processing each one separately.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩        {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
⟨*2ekernel | latexrelease⟩
\def\@fileswith@pti@ns#1[#2]#3[#4]{%
  \ifx#1\@clsextension
    \ifx\@classoptionslist\relax
      \protected@xdef\@classoptionslist{\zap@space#2 \@empty}%
```

Save raw class list.

```
      \gdef\@raw@classoptionslist{#2}%
      \def\reserved@a{%
        \@onefilewithoptions#3[{#2}][{#4}]#1%
        \@documentclasshook}%
    \else
      \def\reserved@a{%
        \@onefilewithoptions#3[{#2}][{#4}]#1}%
    \fi
  \else
```

build up a list of calls to `\@onefilewithoptions` (one for each package) without thrashing the parameter stack.

```
    \def\reserved@b##1,{%
```

If `#1` is `\@nnil` we have reached the end of the list (older version used `\@nil` here but `\@nil` is undefined so `\ifx` equal to all undefined commands)

```
      \ifx\@nnil##1\relax\else
```

If `\ifx\@nnil##1\@nnil` is true then `#1` is (presumably) empty (Older code used `\relax` which is slightly easier to get into `#1` by mistake, which would spoil this test.)

```
        \ifx\@nnil##1\@nnil\else
         \noexpand\@onefilewithoptions##1[{\unexpanded{#2}}][{#4}]%
         \noexpand\@pkgextension
        \fi
        \expandafter\reserved@b
      \fi}%
      \edef\reserved@a{\zap@space#3 \@empty}%
      \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
```

```
  \fi
  \reserved@a}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease⟩        {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
⟨latexrelease⟩\def\@fileswith@pti@ns#1[#2]#3[#4]{%
⟨latexrelease⟩  \ifx#1\@clsextension
⟨latexrelease⟩    \ifx\@classoptionslist\relax
⟨latexrelease⟩      \xdef\@classoptionslist{\zap@space#2 \@empty}%
⟨latexrelease⟩      \def\reserved@a{%
⟨latexrelease⟩        \@onefilewithoptions#3[{#2}][{#4}]#1%
⟨latexrelease⟩        \@documentclasshook}%
⟨latexrelease⟩    \else
⟨latexrelease⟩      \def\reserved@a{%
⟨latexrelease⟩        \@onefilewithoptions#3[{#2}][{#4}]#1}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \def\reserved@b##1,{%
⟨latexrelease⟩      \ifx\@nnil##1\relax\else
⟨latexrelease⟩        \ifx\@nnil##1\@nnil\else
⟨latexrelease⟩         \noexpand\@onefilewithoptions##1[{#2}][{#4}]%
⟨latexrelease⟩         \noexpand\@pkgextension
⟨latexrelease⟩        \fi
⟨latexrelease⟩        \expandafter\reserved@b
⟨latexrelease⟩      \fi}%
⟨latexrelease⟩      \edef\reserved@a{\zap@space#3 \@empty}%
⟨latexrelease⟩      \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \reserved@a}
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩        {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
⟨latexrelease⟩\def\@fileswith@pti@ns#1[#2]#3[#4]{%
⟨latexrelease⟩  \ifx#1\@clsextension
⟨latexrelease⟩    \ifx\@classoptionslist\relax
⟨latexrelease⟩      \xdef\@classoptionslist{\zap@space#2 \@empty}%
⟨latexrelease⟩      \def\reserved@a{%
⟨latexrelease⟩        \@onefilewithoptions#3[{#2}][{#4}]#1%
⟨latexrelease⟩        \@documentclasshook}%
⟨latexrelease⟩    \else
⟨latexrelease⟩      \def\reserved@a{%
⟨latexrelease⟩        \@onefilewithoptions#3[{#2}][{#4}]#1}%
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \else
⟨latexrelease⟩    \def\reserved@b##1,{%
⟨latexrelease⟩      \ifx\@nil##1\relax\else
⟨latexrelease⟩        \ifx\relax##1\relax\else
⟨latexrelease⟩         \noexpand\@onefilewithoptions##1[{#2}][{#4}]%
⟨latexrelease⟩         \noexpand\@pkgextension
⟨latexrelease⟩        \fi
⟨latexrelease⟩        \expandafter\reserved@b
⟨latexrelease⟩      \fi}%
⟨latexrelease⟩      \edef\reserved@a{\zap@space#3 \@empty}%
```

```
⟨latexrelease⟩       \edef\reserved@a{%
⟨latexrelease⟩         \expandafter\reserved@b\reserved@a,\@nil,}%
⟨latexrelease⟩   \fi
⟨latexrelease⟩   \reserved@a}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
\@onlypreamble\@fileswith@pti@ns
```

This macro is used when loading packages or classes.

`\load@onefilewithoptions` Have the main argument as `#1`, so we only need one `\expandafter` above.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩       {\@onefilewithoptions}{Hooks and unused options issue}%
```

Here this macro is called `\@onefilewithoptions`, but further ahead in this file it is renamed to `\load@onefilewithoptions`, and `\@onefilewithoptions` becomes a wrapper around this, used for bookkeeping when rolling back. Therefore, when in latexrelease, we need to define `\load@onefilewithoptions` instead, thus the extra guarded `\def` line below:

```
⟨*2ekernel⟩
\def\@onefilewithoptions#1[#2][#3]#4{%
⟨/2ekernel⟩
⟨latexrelease⟩\def\load@onefilewithoptions#1[#2][#3]#4{%
```

We have to sanitise file names, so that something like

```
\usepackage{some/local/path/array}
\usepackage{array}
```

won't load `array.sty` twice. It is remotely possible that those are two different files, but as a matter of principles, we will consider that the base file name uniquely identifies a package, regardless of where it lives. This assumption already holds for file hooks, for example, which address the hook to a file by its base name only.

We'll use `\@expl@@@filehook@set@curr@file@@nNN` to parse the file name and return the ⟨*path*⟩ and ⟨*base+ext*⟩ in separate token lists. Further ahead, most operations use `\@currname` which doesn't have a path attached to it; only few actions prepend `\@currpath` to `\@currname` (namely loading, as we have to respect the given path).

A file substitution isn't followed just yet because at this point we are parsing user input, so the file is still what the user asked for, and not the file actually loaded.

```
  \@expl@@@filehook@set@curr@file@@nNN{#1.#4}\reserved@a\reserved@b
  \edef\reserved@c{\def\noexpand\reserved@c####1%
    \detokenize\expandafter{\expanded{.#4}}%
    \noexpand\@nil{\def\noexpand\reserved@a{####1}}}\reserved@c
  \expandafter\reserved@c\reserved@a\@nil
  \@pushfilename
  \xdef\@currname{\string@makeletter\reserved@a}%
  \xdef\@currpath{\ifx\reserved@b\@empty\else\reserved@b/\fi}%
  \global\let\@currext#4%
```

The command `\ver@`⟨*file*⟩`.`⟨*ext*⟩ is used to signal that a package is already loaded, either because it is in fact loaded, or because it's loading was suppressed. In minimal installations, said package may not exist but still have its loading suppressed with `\ver@`⟨*file*⟩`.`⟨*ext*⟩, so before checking if the file exists we have to check that we do need

to load it with \@ifl@aded. If we don't, then there's no point in checking for a typo or load-disabling.

```
\@ifl@aded\@currext\@currname
```

In the current preferred approach, a key family name will exist for processing using ltkeys. In that case, we replace the previous package options with the new ones, then call the key handler. Otherwise, we use the more classical clash handler.

```
    {%
      \@ifundefined{opt@handler@\@currname.\@currext}
        {\@onefilewithoptions@clashchk{#2}}
        {%

          \expandafter\protected@edef
            \csname opt@\@currname.\@currext\endcsname
            {\zap@space#2 \@empty}%

          \expandafter\def
            \csname @raw@opt@\@currname.\@currext\expandafter\endcsname
            \expandafter{#2}%
          \@nameuse{opt@handler@\@currname.\@currext}%
        }%
    }%
    {\makeatletter
```

The next line seems to be necessary for 2.09 compatibility (the way the code is written there) This seems questionable and should be look at as in 2e it is definitely unnecessary at this point!

```
     \@reset@ptions
```

First we take the ⟨*name*⟩ and ⟨*ext*⟩ given in the argument and check if the file exists, and issue an error otherwise asking for a correction with \@missingfileerror. For checking if the file exists we use \@currpath (usually empty) before \@currname.

```
     \IfFileExists{\@currpath\@currname.\@currext}{}%
       {\@missing@onefilewithoptions{#2}}%
```

If \@currname is empty (the user replied to the "Enter file name" prompt with ⟨*RETURN*⟩), so stop here (do \@popfilename to pop the item just added above).

This \@gobble omits the date check at the end.

```
     \ifx\@currname\@empty
       \expandafter\@gobble
     \else
```

If the file exists, check if it was load-prevented, and otherwise do the bookkeeping with \@filehook@file@push then call \set@curr@file to set \@curr@file (and do any required substitution), then actually load the class/package with \load@onefile@withoptions. \set@curr@file also needs the file path.

```
       \@disable@packageload@do{\@currname.\@currext}%
         {\@expl@@@filehook@file@push@@
          \set@curr@file{\@currpath\@currname.\@currext}%
          \@filehook@set@CurrentFile
```

The \set@curr@file line above might have replaced the file, so \@currname and \@currext may no longer hold the actual package being loaded, so in that case we need to update these two token lists (\@curr@file holds the file name after replacement, so we parse that).

The requested file is saved in \@currpkg@reqd to be used in \InputIfFileExists later: if the updated \@currname and \@currext are used we lose track of the substitution, so \CurrentFile and \CurrentFileUsed will be (incorrectly) the same.

```
          \expandafter\@swaptwoargs\expandafter
            {\expandafter{\@currpkg@reqd}}%
            {% <
```

\@currpkg@reqd doesn't take a path because it is used later to assign \opt@... and \ver@....

```
          \edef\@currpkg@reqd{\@currname.\@currext}%
          \ifx\CurrentFile\CurrentFileUsed
          \else
            \filename@parse\@curr@file
            \edef\@currpath{\string@makeletter\filename@area}%
            \edef\@currname{\string@makeletter\filename@base}%
            \edef\@currext{\string@makeletter\filename@ext}%
          \fi
          \load@onefile@withoptions{#2}%
          \def\@currpkg@reqd%{\@currpkg@reqd}
            }% >
```

Now just clean up and exit.

```
          \@expl@@@filehook@file@pop@@}%
        \expandafter\@firstofone
      \fi}%
```

Except in the case where \@currname is empty, the date is checked against the date marked in the package file:

```
    {\@ifl@ter\@currext{\@currname}{#3}{}%
      {\@latex@warning@no@line
        {You have requested,\on@line,
         version\MessageBreak
           `#3' of \@cls@pkg\space \@currname,\MessageBreak
         but only version\MessageBreak
          `\csname ver@\@currname.\@currext\endcsname'\MessageBreak
         is available}}%

     \ifx\@currext\@clsextension\let\LoadClass\@twoloadclasserror\fi}%
    \@popfilename
    \@reset@ptions}
```

If the package is already loaded, check that there were no option clashes.

```
\def\@onefilewithoptions@clashchk#1{%
  \@if@ptions\@currext{\@currname}{#1}{}%
      {\@latex@error
        {Option clash for \@cls@pkg\space \@currname}%
        {The package \@currname\space has already been loaded
         with options:\MessageBreak
         \space\space[\@ptionlist{\@currname.\@currext}]\MessageBreak
         There has now been an attempt to load it
          with options\MessageBreak
         \space\space[#1]\MessageBreak
         Adding the global options:\MessageBreak
         \space\space
```

\@onefilewithoptions@clashchk

```
              \@ptionlist{\@currname.\@currext},#1\MessageBreak
         to your \noexpand\documentclass declaration may fix this.%
         \MessageBreak
         Try typing \space <return> \space to proceed.}}%
     \@firstofone}
```

```
\let\@currpkg@reqd\@empty
\@onlypreamble\@onefilewithoptions
```

The kernel no longer uses `\@unprocessedoptions`

```
\let\@unprocessedoptions\@undefined
```

Now the action taken when a file is not found. Path must be included here as it eventually leads to a file lookup.

`\@missing@onefilewithoptions`

```
\def\@missing@onefilewithoptions#1{%
  \@missingfileerror{\@currpath\@currname}\@currext
  \global\let\@currpath\@missingfile@area
  \global\let\@currname\@missingfile@base
  \global\let\@currext\@missingfile@ext}
```

Now the code that actually does the file loading:

`\load@onefile@withoptions`

```
\def\load@onefile@withoptions#1{%
  \let\CurrentOption\@empty
  \@reset@ptions
```

Grab everything in a macro, so the parameter stack is popped before any processing begins.

```
  \def\reserved@a{%
    \@pass@ptions\@currext{#1}{\@currname}%
    \expandafter\let
      \csname opt@\@currpkg@reqd\expandafter\endcsname
      \csname opt@\@currname.\@currext\endcsname
    \expandafter\let
      \csname @raw@opt@\@currpkg@reqd\expandafter\endcsname
      \csname @raw@opt@\@currname.\@currext\endcsname
    \global\expandafter
    \let\csname ver@\@currname.\@currext\endcsname\@empty
```

We initialize `\...-h@@k` here and only if we load the file so that it remains undefined otherwise.

```
    \expandafter\let\csname\@currname.\@currext-h@@k\endcsname\@empty
```

When the current extension is `\@pkgextension` we are loading a package otherwise, if it is `\@clsextension`, a class, so depending on that we execute different hooks. If the extension is neither, then it is another type of file without special hooks.

```
%-----------------------------------------
    \ifx\@currext\@pkgextension
      \UseHook{package/before}%
      \UseOneTimeHook{package/\@currname/before}%
    \else
      \ifx\@currext\@clsextension
        \UseHook{class/before}%
        \UseOneTimeHook{class/\@currname/before}%
      \fi
    \fi
```

Now actually load the file (at this point we are certain it exists, but use `\InputIfFileExists` so that file hooks are executed). `\@currpath` is needed here too.

```
    \InputIfFileExists{\@currpath\@currpkg@reqd}{}%
      {\@latex@error
        {The \@cls@pkg\space\@currpkg@reqd\space failed to load}\@ehd}%
%----------------------------------------
```

In older versions of the code `\@unprocessedoptions` would generate an error for each specified option in a package unless a `\ProcessOptions` has appeared in the package file.

This has changed in 2020. We now use a separate macro per package to avoid interference in case of nested packages. The whole code for handling this issue (GitHub 22) was provided by Hironobu Yamashita, thanks for that.

```
    \expandafter\let\csname unprocessedoptions-\@currname.\@currext\endcsname
                    \@@unprocessedoptions
    \csname\@currname.\@currext-h@@k\endcsname
    \expandafter\let\csname\@currname.\@currext-h@@k\endcsname
              \@undefined
```

Catch the case where the packages has handled the options and redefined `\@unprocessedoptions` to `\relax` (old interface). In that case no error should be produced.

```
    \ifx\@unprocessedoptions\relax
      \let\@unprocessedoptions\@undefined
```

Otherwise run the per package set of unused options.

```
    \else
      \csname unprocessedoptions-\@currname.\@currext\endcsname
    \fi
```

In either case we drop the macro afterwards as it is no longer needed.

```
    \expandafter\let
        \csname unprocessedoptions-\@currname.\@currext\endcsname
       \@undefined
```

And same procedure, James, when we are finished loading, except that the hook order is now reversed.

```
%----------------------------------------
    \ifx\@currext\@pkgextension
      \UseOneTimeHook{package/\@currname/after}%
      \UseHook{package/after}%
    \else
      \ifx\@currext\@clsextension
        \UseOneTimeHook{class/\@currname/after}%
        \UseHook{class/after}%
      \fi
    \fi}%
%----------------------------------------
  \@ifl@aded\@currext\@currname{}{\reserved@a}}
```

Now declare the non-generic package and class hooks used above:

```
\NewHook{package/before}
\NewHook{class/before}
\NewReversedHook{package/after}
\NewReversedHook{class/after}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩      {\@onefilewithoptions}{Hooks and unused options issue}%
⟨latexrelease⟩
```

Because of the way `\@onfilewithoptions` is changed for rollback handling below we have to define `\load@onefilewithoptions` when rolling back!

```
⟨latexrelease⟩\def\load@onefilewithoptions#1[#2][#3]#4{%
⟨latexrelease⟩  \@pushfilename
⟨latexrelease⟩  \xdef\@currname{#1}%
⟨latexrelease⟩  \global\let\@currext#4%
⟨latexrelease⟩  \let\CurrentOption\@empty
⟨latexrelease⟩  \@reset@ptions
⟨latexrelease⟩  \makeatletter
⟨latexrelease⟩  \def\reserved@a{%
⟨latexrelease⟩    \@ifl@aded\@currext{#1}%
⟨latexrelease⟩      {\@if@ptions\@currext{#1}{#2}{}%
⟨latexrelease⟩        {\@latex@error
⟨latexrelease⟩            {Option clash for \@cls@pkg\space #1}%
⟨latexrelease⟩            {The package #1 has already been loaded
⟨latexrelease⟩             with options:\MessageBreak
⟨latexrelease⟩             \space\space[\@ptionlist{#1.\@currext}]\MessageBreak
⟨latexrelease⟩             There has now been an attempt to load it
⟨latexrelease⟩              with options\MessageBreak
⟨latexrelease⟩             \space\space[#2]\MessageBreak
⟨latexrelease⟩             Adding the global options:\MessageBreak
⟨latexrelease⟩             \space\space
⟨latexrelease⟩                  \@ptionlist{#1.\@currext},#2\MessageBreak
⟨latexrelease⟩             to your \noexpand\documentclass declaration may fix this.%
⟨latexrelease⟩             \MessageBreak
⟨latexrelease⟩             Try typing \space <return> \space to proceed.}}}%
⟨latexrelease⟩      {\@pass@ptions\@currext{#2}{#1}%
⟨latexrelease⟩       \global\expandafter
⟨latexrelease⟩       \let\csname ver@\@currname.\@currext\endcsname\@empty
⟨latexrelease⟩       \expandafter\let\csname\@currname.\@currext-h@@k\endcsname\@empty
⟨latexrelease⟩       \InputIfFileExists
⟨latexrelease⟩         {\@currname.\@currext}%
⟨latexrelease⟩         {}%
⟨latexrelease⟩         {\@missingfileerror\@currname\@currext}%
⟨latexrelease⟩    \let\@unprocessedoptions\@@unprocessedoptions
⟨latexrelease⟩    \csname\@currname.\@currext-h@@k\endcsname
⟨latexrelease⟩    \expandafter\let\csname\@currname.\@currext-h@@k\endcsname
⟨latexrelease⟩              \@undefined
⟨latexrelease⟩    \@unprocessedoptions}%
⟨latexrelease⟩    \@ifl@ter\@currext{#1}{#3}{}%
⟨latexrelease⟩      {\@latex@warning@no@line
⟨latexrelease⟩         {You have requested,\on@line,
⟨latexrelease⟩          version\MessageBreak
⟨latexrelease⟩            `#3' of \@cls@pkg\space #1,\MessageBreak
⟨latexrelease⟩          but only version\MessageBreak
⟨latexrelease⟩           `\csname ver@#1.\@currext\endcsname'\MessageBreak
⟨latexrelease⟩          is available}}%
⟨latexrelease⟩    \ifx\@currext\@clsextension\let\LoadClass\@twoloadclasserror\fi
```

```
⟨latexrelease⟩     \@popfilename
⟨latexrelease⟩     \@reset@ptions}%
⟨latexrelease⟩   \reserved@a}
⟨latexrelease⟩
⟨latexrelease⟩\let \load@onefile@withoptions    \@undefined
⟨latexrelease⟩\let \@missing@onefilewithoptions \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@fileswithoptions` *and others.*)

`\@@fileswith@pti@ns` Save the definition (for error checking).

```
\let\@@fileswith@pti@ns\@fileswith@pti@ns
\@onlypreamble\@@fileswith@pti@ns
```

(*End of definition for* `\@@fileswith@pti@ns`.)

`\@reset@ptions` Reset the default option, and clear lists of declared options.

```
\def\@reset@ptions{%
  \global\ifx\@currext\@clsextension
    \let\default@ds\OptionNotUsed
   \else
    \let\default@ds\@unknownoptionerror
  \fi
  \global\let\ds@\@empty
  \global\let\@declaredoptions\@empty}
\@onlypreamble\@reset@ptions
```

(*End of definition for* `\@reset@ptions`.)

4.1 Hooks

Allow code to be saved to be executed at specific later times.

Here we save things in macros. I considered using toks registers (and `\addto@hook` from the NFSS code), but that would require stacking the contents in the case of required packages, so just generate a new macro for each package.

`\@begindocumenthook` `\@enddocumenthook` Stuff to appear at the beginning or end of the document.

```
\ifx\@begindocumenthook\@undefined
  \let\@begindocumenthook\@empty
\fi
\let\@enddocumenthook\@empty
```

(*End of definition for* `\@begindocumenthook` *and* `\@enddocumenthook`.)

`\AtEndOfPackage` `\AtEndOfClass` `\AtBeginDocument` `\AtEndDocument` The access functions.

```
\def\AtEndOfPackage{%
  \expandafter\g@addto@macro\csname\@currname.\@currext-h@@k\endcsname}
\let\AtEndOfClass\AtEndOfPackage
\@onlypreamble\AtEndOfPackage
\@onlypreamble\AtEndOfClass
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\AtBeginDocument}{Use hook system}%
\DeclareRobustCommand\AtBeginDocument{\AddToHook{begindocument}}
\DeclareRobustCommand\AtEndDocument  {\AddToHook{enddocument}}
%\DeclareRobustCommand\AtEndDocument {\AddToHook{env/document/end}} % alternative impl
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\AtBeginDocument}{Use hook system}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand\AtBeginDocument{\g@addto@macro\@begindocumenthook}
⟨latexrelease⟩\DeclareRobustCommand\AtEndDocument{\g@addto@macro\@enddocumenthook}
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
\@onlypreamble\AtBeginDocument
```

(*End of definition for* `\AtEndOfPackage` *and others.*)

`\@cls@pkg` The current file type.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2024/11/01}%
⟨latexrelease⟩                  {\@cls@pkg}{Allow for more extensions}%
\def\@cls@pkg{%
  \ifx\@currext\@clsextension
    document class%
  \else
    \ifx\@currext\@pkgextension
      package%
    \else
      file%
    \fi
  \fi}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@cls@pkg}{Allow for more extensions}%
⟨latexrelease⟩
⟨latexrelease⟩\def\@cls@pkg{%
⟨latexrelease⟩  \ifx\@currext\@clsextension
⟨latexrelease⟩    document class%
⟨latexrelease⟩  \else
⟨latexrelease⟩    package%
⟨latexrelease⟩  \fi}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
\@onlypreamble\@cls@pkg
```

(*End of definition for* `\@cls@pkg`.)

\@unknownoptionerror Bad option.

```
\def\@unknownoptionerror{%
  \@latex@error
    {Unknown option `\CurrentOption' for \@cls@pkg\space`\@currname'}%
    {The option `\CurrentOption' was not declared in
     \@cls@pkg\space`\@currname', perhaps you\MessageBreak
      misspelled its name.
     Try typing \space <return>
     \space to proceed.}}
\@onlypreamble\@unknownoptionerror
```

(*End of definition for* `\@unknownoptionerror`.)

\@@unprocessedoptions Declare an error for each option, unless a \ProcessOptions occurred.

```
\def\@@unprocessedoptions{%
  \ifx\@currext\@pkgextension
    \protected@edef\@curroptions{\@ptionlist{\@currname.\@currext}}%
    \@for\CurrentOption:=\@curroptions\do{%
        \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi}%
  \fi}
\@onlypreamble\@unprocessedoptions
\@onlypreamble\@@unprocessedoptions
```

(*End of definition for* `\@@unprocessedoptions`.)

\@badrequireerror \RequirePackage or \LoadClass occurs in the options section.

```
\def\@badrequireerror#1[#2]#3[#4]{%
  \@latex@error
    {\noexpand\RequirePackage or \noexpand\LoadClass
         in Options Section}%
    {The \@cls@pkg\space `\@currname' is defective.\MessageBreak
     It attempts to load `#3' in the options section, i.e.,\MessageBreak
     between \noexpand\DeclareOption and \string\ProcessOptions.}}
\@onlypreamble\@badrequireerror
```

(*End of definition for* `\@badrequireerror`.)

\@twoloadclasserror Two \LoadClass in a class.

```
\def\@twoloadclasserror{%
  \@latex@error
    {Two \noexpand\LoadClass commands}%
    {You may only use one \noexpand\LoadClass in a class file}}
\@onlypreamble\@twoloadclasserror
```

(*End of definition for* `\@twoloadclasserror`.)

\@twoclasseserror Two \documentclass or \documentstyle.

```
\def\@twoclasseserror#1#{%
  \@latex@error
    {Two \noexpand\documentclass or \noexpand\documentstyle commands}%
    {The document may only declare one class.}\@gobble}
\@onlypreamble\@twoclasseserror
```

(*End of definition for* `\@twoclasseserror`.)

4.2 Providing shipment

`\two@digits` Prefix a number less than 10 with '0'.

```
\def\two@digits#1{\ifnum#1<10 0\fi\number#1}
```

(*End of definition for* `\two@digits`.)

`filecontents` (*env.*) `\filecontents` `\endfilecontents` This environment implements inline files. The star-form does not write extra comments into the file.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\filec@ntents}{Define \q@curr@file directly (gh/220)}%
%
```

We use `@tempswa` to mean no preamble writing and reuse `@filesw` to indicate no overwriting:

```
\def\filecontents{\@tempswatrue\@fileswtrue
  \@ifnextchar[\filec@ntents@opt\filec@ntents
}
\@namedef{filecontents*}{\@tempswafalse\@fileswtrue
  \@ifnextchar[\filec@ntents@opt\filec@ntents
}
```

To handle the optional argument we execute for each option the command `\filec@ntents@OPTION` if it exist or complain about unknown option.

```
\def\filec@ntents@opt[#1]{%
  \edef\@fortmp{\zap@space#1 \@empty}%
  \@for\reserved@a:=\@fortmp\do{%
    \ifcsname filec@ntents@\reserved@a\endcsname
      \csname filec@ntents@\reserved@a\endcsname
    \else
    \@latex@error{Unknown filecontents option \reserved@a}%
       {Valid options are force (or overwrite), nosearch, noheader, nowarn}%
    \fi}%
  \filec@ntents
}
```

Option `force` (or `overwrite`) changes the overwriting switch

```
\let\filec@ntents@force\@fileswfalse
\let\filec@ntents@overwrite\@fileswfalse  % alternative name
```

and option `noheader` the preamble switch (which is equivalent to using the star form of the environment).

```
\let\filec@ntents@noheader\@tempswafalse
```

Option `nosearch` only checks the current directory not the whole TEX tree for the existence of the file to write.

```
\def\filec@ntents@nosearch{%
  \let\filec@ntents@checkdir\@currdir
  \def\filec@ntents@where{in current directory}}
```

By default we search the whole tree:

```
\let\filec@ntents@checkdir\@empty
\def\filec@ntents@where{exists on the system}
```

Option `nowarn` does not show any warning on the terminal but still writes it to the `.log`.

```
\def\filec@ntents@nowarn{%
  \let\filec@ntents@warning\@latex@note@no@line
}
```

By default we show terminal warnings.

```
\let\filec@ntents@warning\@latex@warning@no@line
```

```
\begingroup%
\@tempcnta=1
\loop
  \catcode\@tempcnta=12  %
  \advance\@tempcnta\@ne %
\ifnum\@tempcnta<32      %
\repeat                  %
\catcode`\*=11 %
\catcode`\^^M\active%
\catcode`\^^L\active\let^^L\relax%
\catcode`\^^I\active%
```

```
\gdef\filec@ntents#1{%
  \set@curr@file{\filec@ntents@checkdir#1}%
  \edef\q@curr@file{"\@curr@file"}%
```

LuaTEX has more writes (and 18 is safe here).

```
  \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
  \openin\@inputcheck\q@curr@file \space %
  \ifeof\@inputcheck%
    \@latex@note@no@line%
        {Writing file `\@currdir\@curr@file'}%
```

```
    \ch@ck7\reserved@c\write\relax%
    \immediate\openout\reserved@c\q@curr@file\relax%
  \else%
```

```
    \if@filesw%
      \@latex@note@no@line%
          {File `\@curr@file' already \filec@ntents@where.\MessageBreak%
             Not generating it from this source}%
      \let\write\@gobbletwo%
      \let\closeout\@gobble%
    \else%
```

If we are overwriting, we try to make sure that the user is not by mistake overwriting the input file (`\jobname`). Of course, this only works for input files ending in `.tex`. If a different extension is used there is no way to see that we are overwriting ourselves!

```
      \edef\reserved@b{\detokenize\expandafter{\jobname}}%
      \ifx\@curr@file\reserved@b%
        \@fileswtrue%
      \else%
        \edef\reserved@b{\reserved@b\detokenize{.tex}}%
        \ifx\@curr@file\reserved@b
          \@fileswtrue%
        \fi%
      \fi%
```

We allocate a write channel but we open it only if it is (hopefully) safe. If not opened that means we are going to write on the terminal.

```
      \ch@ck7\reserved@c\write\relax%
      \if@filesw%  % Foul ... trying to overwrite \jobname!
      \@latex@error{Trying to overwrite `\jobname.tex'}{You can't %
        write to the file you are reading from!\MessageBreak%
        Data is written to screen instead.}%
      \else%
        \filec@ntents@warning%
           {Writing or overwriting file `\@curr@file'}%
        \immediate\openout\reserved@c\q@curr@file\relax%
      \fi%
    \fi%
  \fi%
```

Closing the `\@inputcheck` is done here to avoid having to do this in each branch.

```
  \closein\@inputcheck%
  \if@tempswa%

    \immediate\write\reserved@c{%
      \@percentchar\@percentchar\space%
          \expandafter\@gobble\string\LaTeX2e file `\@curr@file'^^J%
      \@percentchar\@percentchar\space  generated by the %
        `\@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
      \@percentchar\@percentchar\space from source `\jobname' on %
         \number\year/\two@digits\month/\two@digits\day.^^J%
      \@percentchar\@percentchar}%
  \fi%
  \let\do\@makeother\dospecials%
```

If there are active characters in the upper half (e.g., from `inputenc`) there would be confusion so we render everything harmless.

```
  \count@ 128\relax%
  \loop%
    \catcode\count@ 11\relax%
    \advance\count@ \@ne%
    \ifnum\count@<\@cclvi%
  \repeat%

  \edef\E{\@backslashchar end\string{\@currenvir\string}}%
  \edef\reserved@b{%
    \def\noexpand\reserved@b%
         ####1\E####2\E####3\relax}%
  \reserved@b{%
    \ifx\relax##3\relax%
```

There was no `\end{filecontents}`

```
      \immediate\write\reserved@c{##1}%
    \else%
```

There was a `\end{filecontents}`, so stop this time.

```
      \edef^^M{\noexpand\end{\@currenvir}}%
      \ifx\relax##1\relax%
      \else%
```

Text before the \end, write it with a warning.

```
          \@latex@warning{Writing text `##1' before %
            \string\end{\@currenvir}\MessageBreak
            as last line of \@curr@file}%
        \immediate\write\reserved@c{##1}%
      \fi%
      \ifx\relax##2\relax%
      \else%
```

Text after the \end, ignore it with a warning.

```
          \@latex@warning{%
            Ignoring text `##2' after \string\end{\@currenvir}}%
      \fi%
    \fi%
    ^^M}%

  \catcode`\^^L\active%
  \let\L\@undefined%
  \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
  \catcode`\^^I\active%
  \let\I\@undefined%
  \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
  \catcode`\^^M\active%
  \edef^^M##1^^M{%
    \noexpand\reserved@b##1\E\E\relax}}%
\endgroup%
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩    {\filec@ntents}{Spaces in file names + optional arg}%
⟨latexrelease⟩
⟨latexrelease⟩\def\filecontents{\@tempswatrue\@fileswtrue
⟨latexrelease⟩  \@ifnextchar[\filec@ntents@opt\filec@ntents
⟨latexrelease⟩}
⟨latexrelease⟩\@namedef{filecontents*}{\@tempswafalse\@fileswtrue
⟨latexrelease⟩  \@ifnextchar[\filec@ntents@opt\filec@ntents
⟨latexrelease⟩}
⟨latexrelease⟩\def\filec@ntents@opt[#1]{%
⟨latexrelease⟩  \edef\@fortmp{\zap@space#1 \@empty}%
⟨latexrelease⟩  \@for\reserved@a:=\@fortmp\do{%
⟨latexrelease⟩    \ifcsname filec@ntents@\reserved@a\endcsname
⟨latexrelease⟩      \csname filec@ntents@\reserved@a\endcsname
⟨latexrelease⟩    \else
⟨latexrelease⟩    \@latex@error{Unknown filecontents option \reserved@a}%
⟨latexrelease⟩       {Valid options are force (or overwrite), nosearch, noheader}%
⟨latexrelease⟩    \fi}%
⟨latexrelease⟩  \filec@ntents
⟨latexrelease⟩}
⟨latexrelease⟩\let\filec@ntents@force\@fileswfalse
⟨latexrelease⟩\let\filec@ntents@overwrite\@fileswfalse  % alternative name
⟨latexrelease⟩\let\filec@ntents@noheader\@tempswafalse
⟨latexrelease⟩\def\filec@ntents@nosearch{%
⟨latexrelease⟩  \let\filec@ntents@checkdir\@currdir
⟨latexrelease⟩  \def\filec@ntents@where{in current directory}}
```

```
⟨latexrelease⟩\let\filec@ntents@checkdir\@empty
⟨latexrelease⟩\def\filec@ntents@where{exists on the system}
⟨latexrelease⟩\begingroup%
⟨latexrelease⟩\@tempcnta=1
⟨latexrelease⟩\loop
⟨latexrelease⟩  \catcode\@tempcnta=12  %
⟨latexrelease⟩  \advance\@tempcnta\@ne %
⟨latexrelease⟩\ifnum\@tempcnta<32      %
⟨latexrelease⟩\repeat                  %
⟨latexrelease⟩\catcode`\*=11 %
⟨latexrelease⟩\catcode`\^^M\active%
⟨latexrelease⟩\catcode`\^^L\active\let^^L\relax%
⟨latexrelease⟩\catcode`\^^I\active%
⟨latexrelease⟩\gdef\filec@ntents#1{%
⟨latexrelease⟩  \set@curr@file{\filec@ntents@checkdir#1}%
⟨latexrelease⟩  \edef\q@curr@file{\expandafter\quote@name\expandafter{\@curr@file}}%
⟨latexrelease⟩  \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
⟨latexrelease⟩  \openin\@inputcheck\q@curr@file \space %
⟨latexrelease⟩  \ifeof\@inputcheck%
⟨latexrelease⟩    \@latex@warning@no@line%
⟨latexrelease⟩        {Writing file `\@currdir\@curr@file'}%
⟨latexrelease⟩    \ch@ck7\reserved@c\write\relax%
⟨latexrelease⟩    \immediate\openout\reserved@c\q@curr@file\relax%
⟨latexrelease⟩  \else%
⟨latexrelease⟩    \if@filesw%
⟨latexrelease⟩      \@latex@warning@no@line%
⟨latexrelease⟩          {File `\@curr@file' already \filec@ntents@where.\MessageBreak%
⟨latexrelease⟩             Not generating it from this source}%
⟨latexrelease⟩      \let\write\@gobbletwo%
⟨latexrelease⟩      \let\closeout\@gobble%
⟨latexrelease⟩    \else%
⟨latexrelease⟩      \edef\reserved@a{#1}%
⟨latexrelease⟩      \edef\reserved@a{\detokenize\expandafter{\reserved@a}}%
⟨latexrelease⟩      \edef\reserved@b{\detokenize\expandafter{\jobname}}%
⟨latexrelease⟩      \ifx\reserved@a\reserved@b%
⟨latexrelease⟩        \@fileswtrue%
⟨latexrelease⟩      \else%
⟨latexrelease⟩        \edef\reserved@b{\reserved@b\detokenize{.tex}}%
⟨latexrelease⟩        \ifx\reserved@a\reserved@b
⟨latexrelease⟩          \@fileswtrue%
⟨latexrelease⟩        \fi%
⟨latexrelease⟩      \fi%
⟨latexrelease⟩      \ch@ck7\reserved@c\write\relax%
⟨latexrelease⟩      \if@filesw%  % Foul ... trying to overwrite \jobname!
⟨latexrelease⟩      \@latex@error{Trying to overwrite `\jobname.tex'}{You can't %
⟨latexrelease⟩        write to the file you are reading from!\MessageBreak%
⟨latexrelease⟩        Data is written to screen instead.}%
⟨latexrelease⟩      \else%
⟨latexrelease⟩        \@latex@warning@no@line%
⟨latexrelease⟩           {Writing or overwriting file `\@currdir\@curr@file'}%
⟨latexrelease⟩        \immediate\openout\reserved@c\q@curr@file\relax%
⟨latexrelease⟩      \fi%
⟨latexrelease⟩    \fi%
⟨latexrelease⟩  \fi%
```

```
⟨latexrelease⟩  \closein\@inputcheck%
⟨latexrelease⟩  \if@tempswa%
⟨latexrelease⟩    \immediate\write\reserved@c{%
⟨latexrelease⟩      \@percentchar\@percentchar\space%
⟨latexrelease⟩          \expandafter\@gobble\string\LaTeX2e file `\@curr@file'^^J%
⟨latexrelease⟩      \@percentchar\@percentchar\space  generated by the %
⟨latexrelease⟩        `\@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
⟨latexrelease⟩      \@percentchar\@percentchar\space from source `\jobname' on %
⟨latexrelease⟩         \number\year/\two@digits\month/\two@digits\day.^^J%
⟨latexrelease⟩      \@percentchar\@percentchar}%
⟨latexrelease⟩  \fi%
⟨latexrelease⟩  \let\do\@makeother\dospecials%
⟨latexrelease⟩  \count@ 128\relax%
⟨latexrelease⟩  \loop%
⟨latexrelease⟩    \catcode\count@ 11\relax%
⟨latexrelease⟩    \advance\count@ \@ne%
⟨latexrelease⟩    \ifnum\count@<\@cclvi%
⟨latexrelease⟩  \repeat%
⟨latexrelease⟩  \edef\E{\@backslashchar end\string{\@currenvir\string}}%
⟨latexrelease⟩  \edef\reserved@b{%
⟨latexrelease⟩    \def\noexpand\reserved@b%
⟨latexrelease⟩         ####1\E####2\E####3\relax}%
⟨latexrelease⟩  \reserved@b{%
⟨latexrelease⟩    \ifx\relax##3\relax%
⟨latexrelease⟩      \immediate\write\reserved@c{##1}%
⟨latexrelease⟩    \else%
⟨latexrelease⟩      \edef^^M{\noexpand\end{\@currenvir}}%
⟨latexrelease⟩      \ifx\relax##1\relax%
⟨latexrelease⟩      \else%
⟨latexrelease⟩          \@latex@warning{Writing text `##1' before %
⟨latexrelease⟩             \string\end{\@currenvir}\MessageBreak as last line of \@curr@file}%
⟨latexrelease⟩        \immediate\write\reserved@c{##1}%
⟨latexrelease⟩      \fi%
⟨latexrelease⟩      \ifx\relax##2\relax%
⟨latexrelease⟩      \else%
⟨latexrelease⟩         \@latex@warning{%
⟨latexrelease⟩           Ignoring text `##2' after \string\end{\@currenvir}}%
⟨latexrelease⟩      \fi%
⟨latexrelease⟩    \fi%
⟨latexrelease⟩    ^^M}%
⟨latexrelease⟩  \catcode`\^^L\active%
⟨latexrelease⟩  \let\L\@undefined%
⟨latexrelease⟩  \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
⟨latexrelease⟩  \catcode`\^^I\active%
⟨latexrelease⟩  \let\I\@undefined%
⟨latexrelease⟩  \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
⟨latexrelease⟩  \catcode`\^^M\active%
⟨latexrelease⟩  \edef^^M##1^^M{%
⟨latexrelease⟩    \noexpand\reserved@b##1\E\E\relax}}%
⟨latexrelease⟩\endgroup%
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩    {\filec@ntents}{Spaces in file names + optional arg}%
⟨latexrelease⟩
```

```
⟨latexrelease⟩\let\filec@ntents@opt        \@undefined
⟨latexrelease⟩\let\filec@ntents@force      \@undefined
⟨latexrelease⟩\let\filec@ntents@overwrite  \@undefined
⟨latexrelease⟩\let\filec@ntents@noheader   \@undefined
⟨latexrelease⟩\let\filec@ntents@nosearch   \@undefined
⟨latexrelease⟩\let\filec@ntents@checkdir   \@undefined
⟨latexrelease⟩\let\filec@ntents@where      \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\begingroup%
⟨latexrelease⟩\@tempcnta=1
⟨latexrelease⟩\loop
⟨latexrelease⟩  \catcode\@tempcnta=12  %
⟨latexrelease⟩  \advance\@tempcnta\@ne %
⟨latexrelease⟩\ifnum\@tempcnta<32      %
⟨latexrelease⟩\repeat                  %
⟨latexrelease⟩\catcode`\*=11 %
⟨latexrelease⟩\catcode`\^^M\active%
⟨latexrelease⟩\catcode`\^^L\active\let^^L\relax%
⟨latexrelease⟩\catcode`\^^I\active%
⟨latexrelease⟩
⟨latexrelease⟩\gdef\filec@ntents#1{%
⟨latexrelease⟩  \openin\@inputcheck#1 %
⟨latexrelease⟩  \ifeof\@inputcheck%
⟨latexrelease⟩    \@latex@warning@no@line%
⟨latexrelease⟩        {Writing file `\@currdir#1'}%
⟨latexrelease⟩    \chardef\reserved@c15 %
⟨latexrelease⟩    \ch@ck7\reserved@c\write%
⟨latexrelease⟩    \immediate\openout\reserved@c#1\relax%
⟨latexrelease⟩  \else%
⟨latexrelease⟩    \closein\@inputcheck%
⟨latexrelease⟩    \@latex@warning@no@line%
⟨latexrelease⟩            {File `#1' already exists on the system.\MessageBreak%
⟨latexrelease⟩             Not generating it from this source}%
⟨latexrelease⟩    \let\write\@gobbletwo%
⟨latexrelease⟩    \let\closeout\@gobble%
⟨latexrelease⟩  \fi%
⟨latexrelease⟩  \if@tempswa%
⟨latexrelease⟩    \immediate\write\reserved@c{%
⟨latexrelease⟩      \@percentchar\@percentchar\space%
⟨latexrelease⟩          \expandafter\@gobble\string\LaTeX2e file `#1'^^J%
⟨latexrelease⟩      \@percentchar\@percentchar\space  generated by the %
⟨latexrelease⟩        `\@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
⟨latexrelease⟩      \@percentchar\@percentchar\space from source `\jobname' on %
⟨latexrelease⟩         \number\year/\two@digits\month/\two@digits\day.^^J%
⟨latexrelease⟩      \@percentchar\@percentchar}%
⟨latexrelease⟩  \fi%
⟨latexrelease⟩  \let\do\@makeother\dospecials%
⟨latexrelease⟩  \count@ 128\relax%
⟨latexrelease⟩  \loop%
⟨latexrelease⟩    \catcode\count@ 11\relax%
⟨latexrelease⟩    \advance\count@ \@ne%
⟨latexrelease⟩    \ifnum\count@<\@cclvi%
⟨latexrelease⟩  \repeat%
⟨latexrelease⟩  \edef\E{\@backslashchar end\string{\@currenvir\string}}%
```

```
⟨latexrelease⟩  \edef\reserved@b{%
⟨latexrelease⟩    \def\noexpand\reserved@b%
⟨latexrelease⟩         ####1\E####2\E####3\relax}%
⟨latexrelease⟩  \reserved@b{%
⟨latexrelease⟩    \ifx\relax##3\relax%
⟨latexrelease⟩      \immediate\write\reserved@c{##1}%
⟨latexrelease⟩    \else%
⟨latexrelease⟩      \edef^^M{\noexpand\end{\@currenvir}}%
⟨latexrelease⟩      \ifx\relax##1\relax%
⟨latexrelease⟩      \else%
⟨latexrelease⟩          \@latex@warning{Writing text ‘##1’ before %
⟨latexrelease⟩             \string\end{\@currenvir}\MessageBreak as last line of #1}%
⟨latexrelease⟩        \immediate\write\reserved@c{##1}%
⟨latexrelease⟩      \fi%
⟨latexrelease⟩      \ifx\relax##2\relax%
⟨latexrelease⟩      \else%
⟨latexrelease⟩         \@latex@warning{%
⟨latexrelease⟩           Ignoring text ‘##2’ after \string\end{\@currenvir}}%
⟨latexrelease⟩      \fi%
⟨latexrelease⟩    \fi%
⟨latexrelease⟩    ^^M}%
⟨latexrelease⟩
⟨latexrelease⟩  \catcode‘\^^L\active%
⟨latexrelease⟩  \let\L\@undefined%
⟨latexrelease⟩  \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
⟨latexrelease⟩  \catcode‘\^^I\active%
⟨latexrelease⟩  \let\I\@undefined%
⟨latexrelease⟩  \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
⟨latexrelease⟩  \catcode‘\^^M\active%
⟨latexrelease⟩  \edef^^M##1^^M{%
⟨latexrelease⟩    \noexpand\reserved@b##1\E\E\relax}}%
⟨latexrelease⟩\endgroup%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\begingroup
\catcode‘|=\catcode‘\%
\catcode‘\%=12
\catcode‘\*=11
\gdef\@percentchar{%}
\gdef\endfilecontents{|
  \immediate\closeout\reserved@c
  \def\T##1##2##3{|
  \ifx##1\@undefined\else
    \@latex@warning@no@line{##2 has been converted to Blank ##3e}|
  \fi}|
  \T\L{Form Feed}{Lin}|
  \T\I{Tab}{Spac}|
  \immediate\write\@unused{}}
\global\let\endfilecontents*\endfilecontents
```

We no longer prevent the code to be used after begin document (no rollback needed for this change).

```
%\@onlypreamble\filecontents
%\@onlypreamble\endfilecontents
```

```
%\@onlypreamble\filecontents*
%\@onlypreamble\endfilecontents*
\endgroup
%\@onlypreamble\filec@ntents
```

(*End of definition for* `\filecontents` *and* `\endfilecontents`.)

5 Package/class rollback mechanism

```
⟨/2ekernel⟩
⟨*2ekernel | latexreleasefirst⟩
```

`\pkgcls@debug` For testing we have a few extra lines of code that by default do nothing but one can set `\pkgcls@debug` to `\typeout` to get extra info. Sometime in the future this will be dropped.

```
⟨*tracerollback⟩
%\let\pkgcls@debug\typeout
\let\pkgcls@debug\@gobble
⟨/tracerollback⟩
```

(*End of definition for* `\pkgcls@debug`.)

`\requestedLaTeXdate` The macro (!) `\requestedLaTeXdate` holds the globally requested rollback date (via `latexrelease`) or zero if no such request was made.

```
\def\requestedLaTeXdate{0}
```

(*End of definition for* `\requestedLaTeXdate`.)

`\pkgcls@targetdate` `\pkgcls@targetlabel` `\pkgcls@innerdate` If a rollback for a package or class is requested then `\pkgcls@targetdate` holds the requested date as a number YYYYMMDD (if there was one, otherwise the value of `\requestedLaTeXdate`) and `\pkgcls@targetlabel` will be empty. If there was a request for a named version then `\pkgcls@targetlabel` holds the version name and `\pkgcls@targetdate` is set to `1`.

`\pkgcls@targetdate=0` is used to indicate that there was no rollback request. While loading an old release `\pkgcls@targetdate` is also reset to zero so that `\DeclareRelease` declarations are bypassed.

In contrast `\pkgcls@innerdate` will always hold the requested date (in a macro not a counter) if there was one, otherwise, e.g., if there was no request or a request to a version name it will contain TEX largest legal number. While loading a file this can be used to provide conditionals that select code based on the request.

```
\ifx\pkgcls@targetdate\@undefined
  \newcount\pkgcls@targetdate
\fi
\let\pkgcls@targetlabel\@empty
\def\pkgcls@innerdate{\maxdimen}
```

(*End of definition for* `\pkgcls@targetdate`, `\pkgcls@targetlabel`, *and* `\pkgcls@innerdate`.)

`\pkgcls@candidate` `\pkgcls@releasedate` When looping through the `\DeclareRelease` declarations we record if the release is the best candidate we have seen so far. This is recorded in `\pkgcls@candidate` and we update it whenever we see a better one.

In `\pkgcls@releasedate` we keep track of the release date of that candidate.

```
\let\pkgcls@candidate\@empty
\let\pkgcls@releasedate\@empty
```

(*End of definition for* `\pkgcls@candidate` *and* `\pkgcls@releasedate`.)

`\load@onefilewithoptions`
`\@onefilewithoptions`

the best place to add the rollback code is at the point where `\@onefilewithoptions` is called to load a single class or package.

To make things easy we save the old definition as `\load@onefilewithoptions` and then provide a new interface.

Important: as this code is also unconditionally placed into latexrelease we can only do this name change once otherwise both macros will contain the same code.

```
\ifx\load@onefilewithoptions\@undefined
 \let\load@onefilewithoptions\@onefilewithoptions
 \def\@onefilewithoptions#1[#2][#3]#4{%
```

First a bit of tracing normally disabled.

```
⟨*tracerollback⟩
  \pkgcls@debug{--- File loaded request (\noexpand\usepackage or ...)}%
  \pkgcls@debug{\@spaces 1: #1}%
  \pkgcls@debug{\@spaces 2: #2}%
  \pkgcls@debug{\@spaces 3: #3}%
  \pkgcls@debug{\@spaces 4: #4}%
⟨/tracerollback⟩
```

Three of the arguments are needed later on in error/warning messages so we save them.

```
  \def\pkgcls@name{#1}%                  % for info message
  \def\pkgcls@arg {#3}%                  % for info message
  \edef\pkgcls@ext{%
    \ifx#4\@clsextension document class\else
      \ifx#4\@pkgextension package\else
        file
      \fi
    \fi
  }%                                     % for info message
```

then we parse the final optional argument to determine if there is a specific rollback request for the current file. This will set `\pkgcls@targetdate`, `\pkgcls@targetlabel` and `\pkgcls@mindate`.

```
  \pkgcls@parse@date@arg{#3}%
```

When determining the correct release to load we keep track of candidates in `\pkgcls@candidate` and initially we don't have any:

```
  \let\pkgcls@candidate\@empty
```

If we had a rollback request then `#3` may contain data but not necessarily a "minimal date" so instead of passing it on we pass on the content of `\pkgcls@mindate`. We need to pass the value not the command, otherwise nested packages may pick up the wrong information.

```
  \begingroup
  \edef\reserved@a{%
    \endgroup
    \unexpanded{\load@onefilewithoptions#1[#2]}%
    [\pkgcls@mindate]%
    \unexpanded{#4}}%
   \reserved@a
 }
\fi
```

(*End of definition for* `\load@onefilewithoptions` *and* `\@onefilewithoptions`.)

`\pkgcls@parse@date@arg` The `\pkgcls@parse@date@arg` command parses the second optional argument of `\usepackage`, `\RequirePackage` or `\documentclass` for a rollback request setting the values of `\pkgcls@targetdate` and `\pkgcls@targetlabel`.

This optional argument has a dual purpose: If it just contains a date string then this means that the package should have at least that date (to ensure that a certain feature is actually available, or a certain bug has been fixed). When the package gets loaded the information in `\Provides...` will then be checked against this request.

But if it starts with an equal sign followed by a date string or followed by a version name then this means that we should roll back to the state of the package at that date or to the version with the requested name.

If there was no optional argument or the optional argument does not start with "`=`" then the `\pkgcls@targetdate` is set to the date of the overall rollback request (via `latexrelease`) or if that was not given it is set to `0`. In either case `\pkgcls@targetlabel` will be made empty.

If the argument doesn't start with "`=`" then it is supposed to be a "minimal date" and we therefore save the value in `\pkgcls@mindate`, otherwise this macro is made empty.

So in summary we have:

Input		`\pkgcls@targetdate`	`\pkgcls@targetlabel`	`\pkgcls@mindate`
⟨*empty*⟩	→	⟨*global-rollbackdate-as-number*⟩	⟨*empty*⟩	⟨*empty*⟩
⟨*date*⟩	→	⟨*global-rollbackdate-as-number*⟩	⟨*empty*⟩	⟨*date*⟩
=⟨*date*⟩	→	⟨*date-as-number*⟩	⟨*empty*⟩	⟨*empty*⟩
=⟨*version*⟩	→	1	⟨*version*⟩	⟨*empty*⟩
⟨*other*⟩	→	⟨*global-rollbackdate-as-number*⟩	⟨*empty*⟩	⟨*other*⟩

where ⟨*global-rollbackdate-as-number*⟩ is a date request given via `latexrelease` or if there wasn't one `0`.

```
\def\pkgcls@parse@date@arg #1{%
```

If the argument is empty we use the rollback date from `latexrelease` which has the value of zero if there was no rollback request. The label and the minimal date is made empty in that case.

```
  \ifx\@nil#1\@nil
    \pkgcls@targetdate\requestedLaTeXdate\relax
    \let\pkgcls@targetlabel\@empty
    \let\pkgcls@mindate\@empty
```

Otherwise we parse the argument further, checking for a `=` as the first character. We append a `=` at the end so that there is at least one such character in the argument.

```
  \else
    \pkgcls@parse@date@arg@#1=\@nil\relax
  \fi
}
```

The actual parsing work then happens in `\pkgcls@parse@date@arg@`:

```
\def\pkgcls@parse@date@arg@#1=#2\@nil{%
```

We set `\pkgcls@targetdate` depending on the parsing result; the code is expandable so we can do the parsing as part of the assignment.

```
  \pkgcls@targetdate
```

If a = was in first position then #1 will be empty. In that case #2 will be the original argument with a = appended.

This can be parsed with \@parse@version, the trailing character is simply ignored. This macro returns the parsed date as a number (or zero if it wasn't a date) and accepts both YYYY/MM/DD and YYYY-MM-DD formats.

```
    \ifx\@nil#1\@nil
      \@parse@version0#2//00\@nil\relax
```

Whatever is returned is thus assigned to \pkgcls@targetdate and therefore we can now test its value. If the value is zero we assume that the remaining argument string represents a version and change \pkgcls@targetdate and set \pkgcls@targetlabel to the version name (after stripping off the trailing =).

```
      \ifnum \pkgcls@targetdate=\z@
        \pkgcls@targetdate\@ne
        \def\pkgcls@innerdate{\maxdimen}%
        \pkgcls@parse@date@arg@version#2%
      \else
        \edef\pkgcls@innerdate{\the\pkgcls@targetdate}%
      \fi
      \let\pkgcls@mindate\@empty
    \else
```

If #1 was not empty then there wasn't a = character in first position so we are dealing either with a "minimum date" or with some incorrect data. We assume the former and make the following assignments (the first one finishing the assignment of \pkgcls@targetdate):

```
      \requestedLaTeXdate\relax
      \let\pkgcls@targetlabel\@empty
      \def\pkgcls@innerdate{\maxdimen}%
      \def\pkgcls@mindate{#1}%
```

If the min-date is after the requested rollback date (if there is any, i.e., if it is not zero) then we have a conflict and therefore issue a warning.

```
      \ifnum \pkgcls@targetdate > \z@
        \ifnum \@parse@version0#1//00\@nil > \pkgcls@targetdate
          \@latex@warning@no@line{Suspicious rollback/min-date date given\MessageBreak
            A minimal date of #1 has been specified for
             \pkgcls@ext\MessageBreak '\pkgcls@name'.\MessageBreak
             But this is in conflict
             with a rollback request to \requestedpatchdate}
        \fi
      \fi
    \fi
}
```

Strip off the trailing = and assign the version name to \pkgcls@targetlabel.

```
\def\pkgcls@parse@date@arg@version#1={%
  \def\pkgcls@targetlabel{#1}}
```

(*End of definition for* \pkgcls@parse@date@arg.)

\DeclareRelease First argument is the "name" of the release and it can be left empty if one doesn't like to give a name to the release. The second argument is that from which on this release was available (or should be used in case of minor updates). The final argument is the external file name of this release, by convention this should be

⟨*pkg/cls-name*⟩-⟨*date*⟩.⟨*extension*⟩ but this is not enforced and through this argument one can overwrite it.

```
\def\DeclareRelease#1#2#3{%
  \ifnum\pkgcls@targetdate>\z@  % some sort of rollback request
⟨*tracerollback⟩
    \pkgcls@debug{---\string\DeclareRelease:}%
    \pkgcls@debug{\@spaces 1: #1}%
    \pkgcls@debug{\@spaces 2: #2}%
    \pkgcls@debug{\@spaces 3: #3}%
⟨/tracerollback⟩
```

If the date argument #2 is empty we are dealing with a special release that should be only accessible via its name; a typical use case would be a "beta" release. So if we are currently processing a date request we ignore it and otherwise we check if we can match the name and if so load the corresponding release file.

```
    \ifx\@nil#2\@nil
      \ifnum\pkgcls@targetdate=\@ne  % named request
        \def\reserved@a{#1}%
        \ifx\pkgcls@targetlabel\reserved@a
          \pkgcls@use@this@release{#3}{}%
⟨*tracerollback⟩
        \else
          \pkgcls@debug{Label doesn't match}%
⟨/tracerollback⟩
        \fi
⟨*tracerollback⟩
      \else
        \pkgcls@debug{Date request: ignored}%
⟨/tracerollback⟩
      \fi
    \else
```

If the value of \pkgcls@targetdate is greater than 1 (or in reality greater than something like 19930101) we are dealing with a rollback request to a specific date.

```
      \ifnum\pkgcls@targetdate>\@ne  % a real request
```

So we parse the date of this release to check if it is before or after the request date.

```
        \ifnum\@parse@version#2//00\@nil
             >\pkgcls@targetdate
```

If it is after we have to distinguish between two cases: If there was an earlier candidate we use that one because the other is too late, but if there wasn't one (i.e., if current release is the oldest that exists) we use it as the best choice. However in that case something is wrong (as there shouldn't be a rollback to a date when a package used didn't yet exists). So we make a complained to the user.

```
          \ifx\pkgcls@candidate\@empty
            \pkgcls@rollbackdate@error{#2}%
            \pkgcls@use@this@release{#3}{#2}%
          \else
            \pkgcls@use@this@release\pkgcls@candidate
                                     \pkgcls@releasedate
          \fi
        \else
```

Otherwise, if the release date of this version is before the target rollback and we record it as a candidate. But we don't use it yet as there may be another release which is still before the target rollback.

```
          \def\pkgcls@candidate{#3}%
          \def\pkgcls@releasedate{#2}%
⟨*tracerollback⟩
          \pkgcls@debug{New candidate: #3}%
⟨/tracerollback⟩
        \fi
      \else
```

If we end up in this branch we have a named version request. So we check if `\pkgcls@targetlabel` matches the current name and if yes we use this release immediately, otherwise we do nothing as a later declaration may match it.

```
        \def\reserved@a{#1}%
        \ifx\pkgcls@targetlabel\reserved@a
          \pkgcls@use@this@release{#3}{#2}%
⟨*tracerollback⟩
        \else
          \pkgcls@debug{Label doesn't match}%
⟨/tracerollback⟩
        \fi
      \fi
    \fi
  \fi
}
```

(*End of definition for* `\DeclareRelease`.)

`\pkgcls@use@this@release` If a certain release has been selected (stored in the external file given in `#1`) we need to input it and afterwards stop reading the current file.

```
\def\pkgcls@use@this@release#1#2{%
```

Before that we record the selection made inside the transcript.

```
  \pkgcls@show@selection{#1}{#2}%
```

We then set the `\pkgcls@targetdate` to zero so that any `\DeclareRelease` or `\DeclareCurrentRelease` in the file we now load are bypassed[49] and then we finally load the correct release.

After loading that file we need to stop reading the current file so we issue `\endinput`. Note that the `\relax` before that is essential to ensure that the `\endinput` is only happening after the file has been fully processed, otherwise it would act after the first line of the `\@@input`!

```
  \pkgcls@targetdate\z@
  \@addtofilelist{#1}%
  \@@input #1\relax
  \endinput
}
```

(*End of definition for* `\pkgcls@use@this@release`.)

[49] The older release may also have such declarations inside if it was a simply copy of the `.sty` or `.cls` file current at that date. Removing these declarations would make the file load a tiny bit faster, but this way it works in any case.

\pkgcls@show@selection This command records what selection was made. As that is needed in two places (and it is rather lengthy) it was placed in a separate command. The first argument is the name of the external file that is being loaded and is only needed for debugging. The second argument is the date that corresponds to this file and it is used as part of the message.

```
\def\pkgcls@show@selection#1#2{%
⟨*tracerollback⟩
  \pkgcls@debug{Result: use  #1}%
⟨/tracerollback⟩
  \GenericInfo
   {\@spaces\@spaces\space}{Rollback for
    \@cls@pkg\space'\@currname' requested ->
    \ifnum\pkgcls@targetdate>\@ne
       date
       \ifnum\requestedLaTeXdate=\pkgcls@targetdate
          \requestedpatchdate
       \else
          \expandafter\@gobble\pkgcls@arg
       \fi.\MessageBreak
```

Instead of "best approximation" we could say that we have been able to exactly match the date (if it is exact), but that would mean extra tests without much gain, so not done.

```
       Best approximation is
    \else
       version '\pkgcls@targetlabel'.\MessageBreak
       This corresponds to
    \fi
    \ifx\@nil#2\@nil
       a special release%
    \else
       the release introduced on #2%
    \fi
    \@gobble}%
}
```

(*End of definition for* \pkgcls@show@selection.)

\pkgcls@rollbackdate@error This is called if the requested rollback date is earlier than the earliest known release of a package or class.

A similar error is given if global rollback date and min-date on a specific package conflict with each other, but that case is happens only once so it is inlined.

```
\def\pkgcls@rollbackdate@error#1{%
  \@latex@error{Suspicious rollback date given}%
     {The \@cls@pkg\space'\@currname'  has no rollback data
      before #1 which\MessageBreak
      is after your requested rollback date --- so
      something may be wrong here.\MessageBreak
      Continue and we use the earliest known release.}}
```

(*End of definition for* \pkgcls@rollbackdate@error.)

\DeclareCurrentRelease This declares the date (and possible name) of the current version of a package or class.

```
\def\DeclareCurrentRelease#1#2{%
```

First we test if \pkgcls@targetdate is greater than zero, otherwise this code is bypassed (as there is no rollback request).

```
  \ifnum\pkgcls@targetdate>\z@  % some sort of rollback request
⟨*tracerollback⟩
    \pkgcls@debug{---DeclareCurrentRelease}%
    \pkgcls@debug{   1: #1}%
    \pkgcls@debug{   2: #2}%
⟨/tracerollback⟩
```

If the value is greater than 1 we have to deal with a date request, so we parse #2 as a date and compare it with \pkgcls@targetdate.

```
    \ifnum\pkgcls@targetdate>\@ne  % a date request
      \ifnum\@parse@version#2//00\@nil
           >\pkgcls@targetdate
```

If it is greater that means the release date if this file is later than the requested rollback date. Again we have two cases: If there was a previous candidate release we use that one as the current release is too young, but if there wasn't we have to use this release nevertheless as there isn't any alternative.

However this case can only happen if there is a \DeclareCurrentRelease but no declared older releases (so basically the use of the declaration is a bit dubious).

```
        \ifx\pkgcls@candidate\@empty
          \pkgcls@rollbackdate@error{#2}%
        \else
          \pkgcls@use@this@release\pkgcls@candidate
                                   \pkgcls@releasedate
        \fi
```

Otherwise the current file is the right release, so we record that in the transcript and then carry on.

```
      \else
        \pkgcls@show@selection{current version}{#2}%
      \fi
    \else % a label request
```

Otherwise we have a rollback request to a named version so we check if that fits the current name and if not give an error as this was the last possible opportunity.

```
      \def\reserved@a{#1}%
      \ifx\pkgcls@targetlabel\reserved@a
        \pkgcls@show@selection{current version}{#2}%
      \else
        \@latex@error{Requested version '\pkgcls@targetlabel' for
          \@cls@pkg\space'\@currname' is unknown}\@ehc
      \fi
    \fi
  \fi
}
```

(*End of definition for* \DeclareCurrentRelease.)

\IfTargetDateBefore This enables a simple form of conditional code inside a class or package file. If there is a date request and the request date is earlier than the first argument the code in the second argument is processed otherwise the code in the third argument is processed. If there was no date request then we also execute the third argument, i.e., we will get the "latest" version of the file.

Most often the second argument (before-date-code) will be empty.

```
\DeclareRobustCommand\IfTargetDateBefore[1]{%
  \ifnum\pkgcls@innerdate <%
        \expandafter\@parse@version\expandafter0#1//00\@nil
    \typeout{Exclude code introduced on #1}%
    \expandafter\@firstoftwo
  \else
    \typeout{Include code introduced on #1}%
    \expandafter\@secondoftwo
  \fi
}
```

(*End of definition for* `\IfTargetDateBefore`.)

```
⟨/2ekernel | latexreleasefirst⟩
```

6 After Preamble

Finally we declare a package that allows all the commands declared above to be `\@onlypreamble` to be used after `\begin{document}`.

```
⟨*afterpreamble⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{pkgindoc}
         [2020-08-08 v1.3m Package Interface in Document (DPC)]
\def\reserved@a#1\do\@classoptionslist#2\do\filec@ntents#3\relax{%
  \gdef\@preamblecmds{#1#3}}
\expandafter\reserved@a\@preamblecmds\relax
⟨/afterpreamble⟩
```

File 51
ltkeys.dtx

1 Creating and using keyval options

As with any key–value input, using key–value pairs as package or class options has two parts: creating the key options and setting (using) them. Options created in this way *may* be used after package loading as general key–value settings: this will depend on the nature of the underlying code.

\DeclareKeys

`\DeclareKeys [⟨family⟩] {⟨declarations⟩}`

Creates a series of options from a comma-separated ⟨*declarations*⟩ list. Each entry in this list is a key–value pair, with the ⟨*key*⟩ having one or more ⟨*properties*⟩. A small number of "basic" ⟨*properties*⟩ are described below. The full range of properties, provided by l3keys, can also be used for more powerful processing. See `interface3` for the full details.

The basic properties provided here are

- `.code` — execute arbitrary code
- `.if` — sets a TEX `\if...` switch
- `.ifnot` — sets an inverted TEX `\if...` switch
- `.pass-to-packages` — for class options, this specifies whether the option should be treated "global" (read by packages from the global list); for package options this property has no effect
- `.store` — stores a value in a macro
- `.usage` – defines whether the option can be given only when loading (`load`), in the preamble (`preamble`) or has no limitation on scope (`general`)

The part of the ⟨*key*⟩ before the ⟨*property*⟩ is the ⟨*name*⟩, with the ⟨*value*⟩ working with the ⟨*property*⟩ to define the behaviour of the option.

For example, with

```
\DeclareKeys[mypkg]
  {
    draft.if          = @mypkg@draft      ,
    draft.usage       = preamble          ,
    name.store        = \@mypkg@name      ,
    name.usage        = load              ,
    second-name.store = \@mypkg@other@name
  }
```

three options would be create. The option `draft` can be given anywhere in the preamble, and will set a switch called `\if@mypkg@draft`. The option `name` can only be given during package loading, and will save whatever value it is given in `\@mypkg@name`. Finally, the option `second-name` can be given anywhere, and will save its value in `\@mypkg@other@name`.

Keys created *before* the use of `\ProcessKeyOptions` act as package options.

\DeclareUnknownKeyHandler

\DeclareUnknownKeyHandler [⟨*family*⟩] {⟨*code*⟩}

The function \DeclareUnknownKeyHandler may be used to define the behavior when an undefined key is encountered. The ⟨*code*⟩ will receive the unknown key name as #1 and the value as #2. These can then be processed as appropriate, e.g. by forwarding to another package.

\ProcessKeyOptions

\ProcessKeyOptions [⟨*family*⟩]

The \ProcessKeyOptions function is used to check the current option list against the keys defined for ⟨*family*⟩. Global (class) options and local (package) options are checked when this function is called in a package.

\SetKeys

\SetKeys [⟨*family*⟩] {⟨*keyvals*⟩}

Sets (applies) the explicit list of ⟨*keyvals*⟩ for the ⟨*family*⟩: it the latter is not given, the value of \@currname used. This command may be used within a package to set options before or after using \ProcessKeyOptions.

1.1 Implementation of ltkeys

```
⟨@@=keys⟩
⟨*2ekernel⟩
\ExplSyntaxOn
```

1.2 Key properties

.code
.if
.ifnot
.store
.usage

```
\group_begin:
  \cs_set_protected:Npn \__keys_tmp:nn #1#2
    {
      \quark_if_recursion_tail_stop:n {#1}
      \cs_new_eq:cc
        { \c__keys_props_root_str . #2 }
        { \c__keys_props_root_str . #1 }
      \__keys_tmp:nn
    }
  \__keys_tmp:nn
    { code:n }          { code }
    { legacy_if_set:n } { if }
    { legacy_if_set_inverse:n } { ifnot }
    { tl_set:N }        { store }
    { usage:n }         { usage }
    { \q_recursion_tail } { }
    \q_recursion_stop
\group_end:
```

(*End of definition for* .code *and others.*)

.pass-to-packages
__keys_scope:n
__keys_scope:N

Used to force options to be global: as this property (uniquely) has an *optional* value, there is a bit of work to do.

```
\cs_new_protected:cpn { \c__keys_props_root_str .pass-to-packages }
  {
    \bool_if:NTF \l__keys_no_value_bool
```

```
      { \__keys_scope:n { true } }
      { \__keys_scope:n }
  }
\cs_new_protected:Npn \__keys_scope:n #1
  {
    \str_case:nnF {#1}
      {
        { true }
          { \__keys_scope:N \clist_put_right:NV }
        { false }
          { \__keys_scope:N \clist_remove_all:NV }
      }
      {
        \msg_error:nnnn { keys }
          { choice-unknown }
          { .pass-to-packages }
          {#1}
      }
  }
\cs_new_protected:Npn \__keys_scope:N #1
  {
    \exp_after:wN \__keys_find_key_module:wNN
      \l_keys_path_str \s__keys_stop
        \l_keys_key_tl \l_keys_key_str
    #1 \l__keys_forced_global_clist \l_keys_key_str
  }
```

(*End of definition for* `.pass-to-packages`, `\__keys_scope:n`, *and* `\__keys_scope:N`.)

1.3 Main mechanism

```
\cs_generate_variant:Nn \clist_if_in:NnT { Ne }
\cs_generate_variant:Nn \clist_if_in:NnTF { Ne }
\cs_generate_variant:Nn \clist_put_right:Nn { Nv }
```

`\l__keys_class_only_clist` Used to track class-only options.

```
\clist_new:N \l__keys_class_only_clist
```

(*End of definition for* `\l__keys_class_only_clist`.)

`\l__keys_forced_global_clist` Used to force options to be global.

```
\clist_new:N \l__keys_forced_global_clist
```

(*End of definition for* `\l__keys_forced_global_clist`.)

`\l__keys_options_clist` A single list is used for all options, into which they are collected before processing.

```
\clist_new:N \l__keys_options_clist
```

(*End of definition for* `\l__keys_options_clist`.)

`\l__keys_options_loading_bool`

Used to indicate we are in the loading phase: controls the outcome of warnings.

```
\bool_new:N \l__keys_options_loading_bool
```

__keys_options:n
__keys_options_aux:n
__keys_options_end:

The main function calls functions to collect up the global and local options into `\l__keys_options_clist` before calling the underlying functions to actually do the processing. So that a suitable message is produced if the option is unknown, the special `unknown` key is set if it does not already exist for the current family, and is cleaned up afterwards if required. To allow the LaTeX 2ε layer to know this mechanism is active, and to deal with the key family not matching the file name, we store the family in all cases.

```
\cs_new_protected:Npn \__keys_options:n #1
  { \__keys_options_expand_module:Nn \__keys_options_aux:n {#1} }
\cs_new_protected:Npn \__keys_options_aux:n #1
  {
    \cs_gset_protected:cpn { opt@handler@\@currname.\@currext }
      { \ProcessKeyOptions [ #1 ] }
    \cs_set_protected:Npn \__keys_option_end: { }
    \clist_clear:N \l__keys_options_clist
    \__keys_options_global:n {#1}
    \__keys_options_local:
    \keys_if_exist:nnF {#1} { unknown }
      {
        \keys_define:nn {#1}
          {
            unknown .code:n =
              {
                \msg_error:nnxx { keys } { option-unknown }
                  { \l_keys_key_str } { \@currname }
              }
          }
        \cs_set_protected:Npn \__keys_option_end:
          { \keys_define:nn {#1} { unknown .undefine: } }
      }
    \bool_set_true:N \l__keys_options_loading_bool
    \clist_map_variable:NNn \l__keys_options_clist \CurrentOption
      { \keys_set:nV {#1} \CurrentOption }
    \bool_set_false:N \l__keys_options_loading_bool
    \AtEndOfPackage { \cs_set_eq:NN \@unprocessedoptions \scan_stop: }
    \__keys_option_end:
    \__keys_options_loaded:n {#1}
  }
```

```
\msg_new:nnnn { keys } { option-unknown }
  { Unknown~option~'#1'~for~package~#2. }
  {
    LaTeX~has~been~asked~to~set~an~option~called~'#1'~
    but~the~package~"\msg_module_name:n {#2}"~has~not~created~an~option~with~this~name.
  }
```

(*End of definition for* `\__keys_options:n`, `\__keys_options_aux:n`, *and* `\__keys_options_end:`.)

__keys_options_global:n

Global (class) options are handled differently for LaTeX 2ε packages and classes. Hence this function is essentially a check on the current file type. The initial test is needed as LaTeX 2ε allows variables to be equal to `\scan_stop:`, which is usually forbidden in expl3 code.

```
\cs_new_protected:Npn \__keys_options_global:n #1
  {
    \cs_if_eq:NNF \@raw@classoptionslist \scan_stop:
```

```
      {
        \cs_if_eq:NNTF \@currext \@clsextension
          { \__keys_options_class:n {#1} }
          { \__keys_options_package:n {#1} }
      }
  }
```

(*End of definition for* `\__keys_options_global:n`.)

`\__keys_options_class:n`
`\__keys_options_class:nnn`
`\__keys_options_class:nn`

For classes, each option (stripped of any content after `=`) is checked for existence as a key. If found, the option is added to the combined list for processing. On the other hand, unused options are stored up in `\@unusedoptionlist`. An earlier version of this code checked for the `unknown` key just once and if found short-cutted the loop: that though makes handling more complex situations harder, so we take the performance hit instead. Options used by classes are tracked but the catch-all `unknown` is excluded (hence not using a lazy evaluation for the key testing).

```
\cs_new_protected:Npn \__keys_options_class:n #1
  {
    \cs_if_free:cF { @raw@opt@ \@currname . \@currext }
      {
        \clist_map_inline:cn { @raw@opt@ \@currname . \@currext }
          {
            \exp_args:Ne \__keys_options_class:nnn
              { \tl_trim_spaces:e { \__keys_remove_equals:n {##1} } }
                {##1} {#1}
          }
      }
  }
\cs_new_protected:Npn \__keys_options_class:nnn #1#2#3
  {
    \keys_if_exist:nnTF {#3} {#1}
      {
        \__keys_options_class:nn {#1} {#2}
        \clist_put_right:Ne \l__keys_class_only_clist { \tl_to_str:n {#1} }
      }
      {
        \keys_if_exist:nnTF {#3} { unknown }
          { \__keys_options_class:nn {#1} {#2} }
          {
            \clist_if_in:NnF \@unusedoptionlist {#1}
              { \clist_put_right:Nn \@unusedoptionlist {#1} }
          }
      }
  }
\cs_new_protected:Npn \__keys_options_class:nn #1#2
  {
    \clist_remove_all:Nn \@unusedoptionlist {#1}
    \clist_put_right:Nn \l__keys_options_clist {#2}
  }
```

(*End of definition for* `\__keys_options_class:n`, `\__keys_options_class:nnn`, *and* `\__keys_options_class:nn`.)

__keys_options_package:n
__keys_options_package:nnn
__keys_options_package:nn

For global options when processing a package, the tasks are slightly different from those for a class. The check is the same, but here there is nothing to do if the option is not applicable. Each valid option also needs to be removed from \@unusedoptionlist.

```
\cs_new_protected:Npn \__keys_options_package:n #1
  {
    \clist_map_inline:Nn \@raw@classoptionslist
      {
        \exp_args:Ne \__keys_options_package:nnn
          { \tl_trim_spaces:e { \__keys_remove_equals:n {##1} } }
          {##1} {#1}
      }
  }
```

The forced-global test here needs to use \tl_to_str:n as the data come from a key name, which is always a string.

```
\cs_new_protected:Npn \__keys_options_package:nnn #1#2#3
  {
    \keys_if_exist:nnT {#3} {#1}
      {
        \clist_if_in:NeTF \l__keys_class_only_clist { \tl_to_str:n {#1} }
          {
            \clist_if_in:NeT \l__keys_forced_global_clist { \tl_to_str:n {#1} }
              { \__keys_options_package:nn {#1} {#2} }
          }
          { \__keys_options_package:nn {#1} {#2} }
      }
   }
\cs_new_protected:Npn \__keys_options_package:nn #1#2
  {
    \clist_put_right:Nn \l__keys_options_clist {#2}
    \clist_remove_all:Nn \@unusedoptionlist {#1}
  }
```

(*End of definition for* __keys_options_package:n*,* __keys_options_package:nnn*, and* __keys_options_package:nn.)

__keys_options_local:

If local options are found, they are added to the processing list. LaTeX 2_ε stores options for each file in a macro which may or may not exist, hence the need to use \cs_if_exist:c.

```
\cs_new_protected:Npn \__keys_options_local:
  {
    \cs_if_eq:NNF \@currext \@clsextension
      {
        \cs_if_exist:cT { @raw@opt@ \@currname . \@currext }
          {
            \clist_put_right:Nv \l__keys_options_clist
              { @raw@opt@ \@currname . \@currext }
          }
      }
  }
```

(*End of definition for* __keys_options_local:.)

__keys_remove_equals:n
__keys_remove_equals:w

As the name suggests, this is a simple function to remove an equals sign from the input. This is all wrapped up in an n function so that there will always be a sign available.

```
\cs_new:Npn \__keys_remove_equals:n #1
  { \__keys_remove_equals:w #1 = \s__keys_stop }
\cs_new:Npn \__keys_remove_equals:w #1 = #2 \s__keys_stop { \exp_not:n {#1} }
```

(*End of definition for* `\__keys_remove_equals:n` *and* `\__keys_remove_equals:w`.)

1.4 The document interfaces

```
\cs_generate_variant:Nn \keys_define:nn { nx }
```

`\__keys_options_expand_module:Nn`
`\__keys_options_expand_module:nN`

To deal with active characters inside the module argument whilst also expanding that argument, we use a combination of `c`- and `f`-type expansion. This works as the definitions for active UTF-8 bytes contain an `\ifincsname` test.

```
\cs_new_protected:Npn \__keys_options_expand_module:Nn #1#2
  {
    \cs:w __keys_options_expand_module:nN \use:e { \cs_end: {#2} } #1
  }
\cs_new_protected:Npn \__keys_options_expand_module:nN #1#2
  { #2 {#1} }
```

(*End of definition for* `\__keys_options_expand_module:Nn` *and* `\__keys_options_expand_module:nN`.)

`\DeclareKeys`

Defining key options is quite straight-forward: we have an intermediate function to allow for potential set-up steps.

```
\NewDocumentCommand \DeclareKeys { O { \@currname } +m }
  { \__keys_options_expand_module:Nn \keys_define:nn {#1} {#2} }
```

(*End of definition for* `\DeclareKeys`. *This function is documented on page* 1027.)

`\DeclareUnknownKeyHandler`

```
\NewDocumentCommand \DeclareUnknownKeyHandler { O { \@currname } +m }
  {
    \cs_set_protected:cpn { __keys_unknown_handler_ #1 :nn } ##1##2 {#2}
    \__keys_options_expand_module:Nn \keys_define:nx {#1}
      {
        unknown .code:n =
          \exp_not:N \exp_args:NV
            \exp_not:c { __keys_unknown_handler_ #1 :nn }
            \exp_not:N \l_keys_key_str {####1}
      }
  }
```

(*End of definition for* `\DeclareUnknownKeyHandler`. *This function is documented on page* 1028.)

`\ProcessKeyOptions`

We need to deal with the older interface from l3keys2e here: it had a mandatory argument. We can mop that up using a look-ahead, and then exploit that information to determine whether the package option handling is set up for the new approach for clash handling.

```
\NewDocumentCommand \ProcessKeyOptions { O { \@currname } }
  { \__keys_options:n {#1} }
\@onlypreamble \ProcessKeyOptions
```

(*End of definition for* `\ProcessKeyOptions`. *This function is documented on page* 1028.)

1.5 Option usage scope

__keys_options_loaded:n
__keys_options_loaded:nn

Indicates that the load-time options for a package have been processed: once this has happened, make them unavailable either with a warning or an error.

```
\cs_new_protected:Npn \__keys_options_loaded:n #1
  {
    \prop_get:NnNT \l_keys_usage_load_prop {#1} \l__keys_tmpa_tl
      {
        \clist_map_inline:Nn \l__keys_tmpa_tl
          {
            \keys_define:nn {#1}
              {
                ##1 .code:n =
                  \__keys_options_loaded:nn {#1} {##1}
              }
          }
      }
  }
\cs_new_protected:Npn \__keys_options_loaded:nn #1#2
  {
    \bool_if:NTF \l__keys_options_loading_bool
      { \msg_warning:nnnn { keys } { load-option-ignored } }
      { \msg_error:nnnn { keys } { load-only } }
        {#1} {#2}
  }
```

```
\msg_new:nnn { keys } { load-option-ignored }
  {
    Package~"\msg_module_name:n {#1}"~has~already~been~loaded:~
    ignoring~load-time~option~"#2".
  }
```

```
\msg_new:nnnn { keys } { load-only }
  {
    Key~"#2"~may~only~be~used~during~loading~of~package~
    "\msg_module_name:n {#1}".
  }
  {
    LaTeX~was~asked~to~set~a~key~called~"#2",~but~this~is~only~allowed~
    in~the~optional~argument~when~loading~package~"\msg_module_name:n{#1}".
  }
```

(*End of definition for* `\__keys_options_loaded:n` *and* `\__keys_options_loaded:nn`.)

Disable all preamble options in one shot.

```
\tl_gput_left:Nn \@kernel@after@begindocument
  {
    \prop_map_inline:Nn \l_keys_usage_preamble_prop
      {
        \clist_map_inline:nn {#2}
          {
            \keys_define:nn {#1}
              {
                ##1 .code:n =
                  \msg_error:nnn { keys } { preamble-only } {##1}
              }
```

```
          }
      }
  }
\msg_new:nnnn { keys } { preamble-only }
  { Key~"#1"~may~only~be~used~in~the~preamble. }
  {
    LaTeX~was~asked~to~set~a~key~called~"#1",~but~this~is~only~allowed~
    before~\begin{document}.~You~will~need~to~set~the~key~earlier.
  }
```

1.6 General key setting

\SetKeys A simple wrapper.

```
\NewDocumentCommand \SetKeys { O { \@currname } +m }
  { \__keys_options_expand_module:Nn \keys_set:nn {#1} {#2} }
```

(*End of definition for* \SetKeys*. This function is documented on page 1028.*)

```
\ExplSyntaxOff
```

⟨/2ekernel⟩

File 52
ltfilehook.dtx

1 Introduction

1.1 Provided hooks

The code offers a number of hooks into which packages (or the user) can add code to support different use cases. Many hooks are offered as pairs (i.e., the second hook is reversed. Also important to know is that these pairs are properly nested with respect to other pairs of hooks.

There are hooks that are executed for all files of a certain type (if they contain code), e.g., for all "include files" or all "packages", and there are also hooks that are specific to a single file, e.g., do something after the package `foo.sty` has been loaded.

1.2 General hooks for file reading

There are four hooks that are called for each file that is read using document-level commands such as `\input`, `\include`, `\usepackage`, etc. They are not called for files read using internal low-level methods, such as `\@input` or `\openin`.

`file/before`
`file/.../before`
`file/.../after`
`file/after`

These are:

`file/before`, `file/`⟨*file-name*⟩`/before` These hooks are executed in that order just before the file is loaded for reading. The code of the first hook is used with every file, while the second is executed only for the file with matching ⟨*file-name*⟩ allowing you to specify code that only applies to one file.

`file/`⟨*file-name*⟩`/after`, `file/after` These hooks are after the file with name ⟨*file-name*⟩ has been fully consumed. The order is swapped (the specific one comes first) so that the `/before` and `/after` hooks nest properly, which is important if any of them involve grouping (e.g., contain environments, for example). Furthermore both hooks are reversed hooks to support correct nesting of different packages adding code to both `/before` and `/after` hooks.

So the overall sequence of hook processing for any file read through the user interface commands of LaTeX is:

```
\UseHook{⟨file/before⟩}
\UseHook{⟨file/⟨file name⟩/before⟩}
   ⟨file contents⟩
\UseHook{⟨file/⟨file name⟩/after⟩}
\UseHook{⟨file/after⟩}
```

The file hooks only refer to the file by its name and extension, so the ⟨*file name*⟩ should be the file name as it is on the filesystem with extension (if any) and without paths. Different from `\input` and similar commands, the `.tex` extension is not assumed in hook ⟨*file name*⟩, so `.tex` files must be specified with their extension to be recognized. Files within subfolders should also be addressed by their name and extension only.

Extensionless files also work, and should then be given without extension. Note however that TEX prioritizes `.tex` files, so if two files `foo` and `foo.tex` exist in the search path, only the latter will be seen.

When a file is input, the ⟨*file name*⟩ is available in `\CurrentFile`, which is then used when accessing the `file/`⟨*file name*⟩`/before` and `file/`⟨*file name*⟩`/after`.

`\CurrentFile`

The name of the file about to be read (or just finished) is available to the hooks through `\CurrentFile` (there is no `expl3` name for it for now). The file is always provided with its extension, i.e., how it appears on your hard drive, but without any specified path to it. For example, `\input{sample}` and `\input{app/sample.tex}` would both have `\CurrentFile` being `sample.tex`.

`\CurrentFilePath`

The path to the current file (complement to `\CurrentFile`) is available in `\CurrentFilePath` if needed. The paths returned in `\CurrentFilePath` are only user paths, given through `\input@path` (or `expl3`'s equivalent `\l_file_search_path_seq`) or by directly typing in the path in the `\input` command or equivalent. Files located by `kpsewhich` get the path added internally by the TEX implementation, so at the macro level it looks as if the file were in the current folder, so the path in `\CurrentFilePath` is empty in these cases (package and class files, mostly).

`\CurrentFileUsed`
`\CurrentFilePathUsed`

In normal circumstances these are identical to `\CurrentFile` and `\CurrentFilePath`. They will differ when a file substitution has occurred for `\CurrentFile`. In that case, `\CurrentFileUsed` and `\CurrentFilePathUsed` will hold the actual file name and path loaded by LATEX, while `\CurrentFile` and `\CurrentFilePath` will hold the names that were *asked for*. Unless doing very specific work on the file being read, `\CurrentFile` and `\CurrentFilePath` should be enough.

1.3 Hooks for package and class files

Commands to load package and class files (e.g., `\usepackage`, `\RequirePackage`, `\LoadPackageWithOptions`, etc.) offer the hooks from section 1.2 when they are used to load a package or class file, e.g., `file/array.sty/after` would be called after the `array` package got loaded. But as packages and classes form as special group of files, there are some additional hooks available that only apply when a package or class is loaded.

`package/before`
`package/after`
`package/.../before`
`package/.../after`
`class/before`
`class/after`
`class/.../before`
`class/.../after`

These are:

`package/before`, `package/after` These hooks are called for each package being loaded.

`package/`⟨*name*⟩`/before`, `package/`⟨*name*⟩`/after` These hooks are additionally called if the package name is ⟨*name*⟩ (without extension).

`class/before`, `class/after` These hooks are called for each class being loaded.

`class/`⟨*name*⟩`/before`, `class/`⟨*name*⟩`/after` These hooks are additionally called if the class name is ⟨*name*⟩ (without extension).

All `/after` hooks are implemented as reversed hooks.
The overall sequence of execution for `\usepackage` and friends is therefore:

```
\UseHook{⟨package/before⟩}
\UseHook{⟨package/⟨package name⟩/before⟩}
    \UseHook{⟨file/before⟩}
    \UseHook{⟨file/⟨package name⟩.sty/before⟩}
        ⟨package contents⟩
    \UseHook{⟨file/⟨package name⟩.sty/after⟩}
    \UseHook{⟨file/after⟩}
    code from \AtEndOfPackage if used inside the package
\UseHook{⟨package/⟨package name⟩/after⟩}
\UseHook{⟨package/after⟩}
```

and similar for class file loading, except that `package/` is replaced by `class/` and `\AtEndOfPackage` by `\AtEndOfClass`.

If a package or class is not loaded (or it was loaded before the hooks were set) none of the hooks are executed!

All class or package hooks involving the name of the class or package are implemented as one-time hooks, whereas all other such hooks are normal hooks. This allows for the following use case

```
\AddToHook{package/varioref/after}
      { ... apply my customizations if the package gets
        loaded (or was loaded already) ... }
```

without the need to first test if the package is already loaded.

1.4 Hooks for `\include` files

To manage `\include` files, LaTeX issues a `\clearpage` before and after loading such a file. Depending on the use case one may want to execute code before or after these `\clearpage`s especially for the one that is issued at the end.

Executing code before the final `\clearpage`, means that the code is processed while the last page of the included material is still under construction. Executing code after it means that all floats from inside the include file are placed (which might have added further pages) and the final page has finished.

Because of these different scenarios we offer hooks in three places.[50] None of the hooks are executed when an `\include` file is bypassed because of an `\includeonly` declaration. They are, however, all executed if LaTeX makes an attempt to load the `\include` file (even if it doesn't exist and all that happens is "`No file ⟨filename⟩.tex`").

[50] If you want to execute code before the first `\clearpage` there is no need to use a hook—you can write it directly in front of the `\include`.

include/before
include/.../before
include/end
include/.../end
include/after
include/.../after

These are:

include/before, include/⟨*name*⟩/before These hooks are executed (in that order) after the initial `\clearpage` and after `.aux` file is changed to use ⟨*name*⟩`.aux`, but before the ⟨*name*⟩`.tex` file is loaded. In other words they are executed at the very beginning of the first page of the `\include` file.

include/⟨*name*⟩/end, include/end These hooks are executed (in that order) after LaTeX has stopped reading from the `\include` file, but before it has issued a `\clearpage` to output any deferred floats.

include/⟨*name*⟩/after, include/after These hooks are executed (in that order) after LaTeX has issued the `\clearpage` but before is has switched back writing to the main `.aux` file. Thus technically we are still inside the `\include` and if the hooks generate any further typeset material including anything that writes to the `.aux` file, then it would be considered part of the included material and bypassed if it is not loaded because of some `\includeonly` statement.[51]

include/excluded, include/⟨*name*⟩/excluded The above hooks for `\include` files are only executed when the file is loaded (or more exactly the load is attempted). If, however, the `\include` file is explicitly excluded (through an `\includeonly` statement) the above hooks are bypassed and instead the `include/excluded` hook followed by the `include/`⟨*name*⟩`/excluded` hook are executed. This happens after LaTeX has loaded the `.aux` file for this include file, i.e., after LaTeX has updated its counters to pretend that the file was seen.

All `include` hooks involving the name of the included file are implemented as one-time hooks (whereas all other such hooks are normal hooks).

If you want to execute code that is run for every `\include` regardless of whether or not it is excluded, use the `cmd/include/before` or `cmd/include/after` hooks.

1.5 High-level interfaces for LaTeX

We do not provide any additional wrappers around the hooks (like filehook or scrlfile do) because we believe that for package writers the high-level commands from the hook management, e.g., `\AddToHook`, etc. are sufficient and in fact easier to work with, given that the hooks have consistent naming conventions.

[51] For that reason another `\clearpage` is executed after these hooks which normally does nothing, but starts a new page if further material got added this way.

1.6 Kernel, class, and package interfaces for LaTeX

`\declare@file@substitution`	`\declare@file@substitution {⟨file⟩} {⟨replacement-file⟩}`
`\undeclare@file@substitution`	`\undeclare@file@substitution {⟨file⟩}`

If ⟨*file*⟩ is requested for loading replace it with ⟨*replacement-file*⟩. `\CurrentFile` remains pointing to ⟨*file*⟩ but `\CurrentFileUsed` will show the file actually loaded.

The main use case for this declaration is to provide a corrected version of a package that can't be changed (due to its license) but no longer functions because of LaTeX kernel changes, for example, or to provide a version that makes use of new kernel functionality while the original package remains available for use with older releases. As such it is mainly meant for use in the LaTeX kernel but other use cases are conceivable.

The `\undeclare@file@substitution` declaration undoes a substitution made earlier.

> *Please do not misuse this functionality and replace a file with another unless if really needed and only if the new version is implementing the same functionality as the original one!*

`\disable@package@load`	`\disable@package@load {⟨package⟩} {⟨alternate-code⟩}`
`\reenable@package@load`	`\reenable@package@load {⟨package⟩}`

If ⟨*package*⟩ is requested, do not load it but instead run ⟨*alternate-code*⟩ which could issue a warning, error or any other code.

The main use case is for classes that want to restrict the set of supported packages or contain code that make the use of some packages impossible. So rather than waiting until the document breaks they can set up informative messages why certain packages are not available.

The function is only implemented for packages not for arbitrary files and again it should only be applied if there are good reasons for doing this.[52]

1.7 A sample package for structuring the log output

As an application we provide the package structuredlog that adds lines to the `.log` when a file is opened and closed for reading keeping track of nesting level es well. For example, for the current document it adds the lines

```
= (LEVEL 1 START) t1lmr.fd
= (LEVEL 1 STOP) t1lmr.fd
= (LEVEL 1 START) supp-pdf.mkii
= (LEVEL 1 STOP) supp-pdf.mkii
= (LEVEL 1 START) nameref.sty
== (LEVEL 2 START) refcount.sty
== (LEVEL 2 STOP) refcount.sty
== (LEVEL 2 START) gettitlestring.sty
== (LEVEL 2 STOP) gettitlestring.sty
= (LEVEL 1 STOP) nameref.sty
= (LEVEL 1 START) ltfilehook-doc.out
```

[52] Just to be sure: "I don't like this package by somebody else" is not a good one :-)

```
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.out
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.hd
= (LEVEL 1 STOP) ltfilehook-doc.hd
= (LEVEL 1 START) ltfilehook.dtx
== (LEVEL 2 START) ot1lmr.fd
== (LEVEL 2 STOP) ot1lmr.fd
== (LEVEL 2 START) omllmm.fd
== (LEVEL 2 STOP) omllmm.fd
== (LEVEL 2 START) omslmsy.fd
== (LEVEL 2 STOP) omslmsy.fd
== (LEVEL 2 START) omxlmex.fd
== (LEVEL 2 STOP) omxlmex.fd
== (LEVEL 2 START) umsa.fd
== (LEVEL 2 STOP) umsa.fd
== (LEVEL 2 START) umsb.fd
== (LEVEL 2 STOP) umsb.fd
== (LEVEL 2 START) ts1lmr.fd
== (LEVEL 2 STOP) ts1lmr.fd
== (LEVEL 2 START) t1lmss.fd
== (LEVEL 2 STOP) t1lmss.fd
= (LEVEL 1 STOP) ltfilehook.dtx
```

Thus if you inspect an issue in the `.log` it is easy to figure out in which file it occurred, simply by searching back for `LEVEL` and if it is a `STOP` then remove 1 from the level value and search further for `LEVEL` with that value which should then be the `START` level of the file you are in.

2 The Implementation

```
⟨*2ekernel⟩
⟨@@=filehook⟩
```

2.1 Document and package-level commands

`\CurrentFile`
`\CurrentFilePath`
`\CurrentFileUsed`
`\CurrentFilePathUsed`

User-level macros that hold the current file name and file path. These are used internally as well because the code takes care to protect against a possible redefinition of these macros in the loaded file (it's necessary anyway to make hooks work with nested `\input`). The versions `\...Used` hold the *actual* file name and path that is loaded by LaTeX, whereas the other two hold the name as requested. They will differ in case there's a file substitution.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\CurrentFile}{Hook management file}%
\ExplSyntaxOn
\tl_new:N \CurrentFile
\tl_new:N \CurrentFilePath
\tl_new:N \CurrentFileUsed
\tl_new:N \CurrentFilePathUsed
```

```
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\CurrentFile}{Hook management file}%
⟨latexrelease⟩
⟨latexrelease⟩\let \CurrentFile         \@undefined
⟨latexrelease⟩\let \CurrentFilePath     \@undefined
⟨latexrelease⟩\let \CurrentFileUsed     \@undefined
⟨latexrelease⟩\let \CurrentFilePathUsed \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\CurrentFile` *and others. These functions are documented on page* 1037.)

2.2 expl3 helpers

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩          {\__filehook_file_parse_full_name:nN}{File helpers}%
\ExplSyntaxOn
```

`\__filehook_file_parse_full_name:nN`
`\__filehook_full_name:nn`

A utility macro to trigger expl3's file-parsing and lookup, and return a normalized representation of the file name. If the queried file doesn't exist, no normalization takes place. The output of `\__filehook_file_parse_full_name:nN` is passed on to the `#2`—a 3-argument macro that takes the ⟨*path*⟩, ⟨*base*⟩, and ⟨*ext*⟩ parts of the file name.

```
\cs_new:Npn \__filehook_file_parse_full_name:nN #1
  {
    \exp_args:Nf \file_parse_full_name_apply:nN
      {
        \exp_args:Nf \__filehook_full_name:nn
          { \file_full_name:n {#1} } {#1}
      }
  }
\cs_new:Npn \__filehook_full_name:nn #1 #2
  {
    \tl_if_empty:nTF {#1}
      { \tl_trim_spaces:n {#2} }
      { \tl_trim_spaces:n {#1} }
  }
```

(*End of definition for* `\__filehook_file_parse_full_name:nN` *and* `\__filehook_full_name:nn`.)

`\__filehook_if_no_extension:nTF`
`\__filehook_drop_extension:N`

Some actions depend on whether the file extension was explicitly given, and sometimes the extension has to be removed. The macros below use `\__filehook_file_parse_full_name:nN` to split up the file name and either check if ⟨*ext*⟩ (`#3`) is empty, or discard it.

```
\cs_new:Npn \__filehook_if_no_extension:nTF #1
  {
    \exp_args:Ne \tl_if_empty:nTF
      { \file_parse_full_name_apply:nN {#1} \use_iii:nnn }
  }
```

```
\cs_new_protected:Npn \__filehook_drop_extension:N #1
  {
    \tl_gset:Nx #1
      {
        \exp_args:NV \__filehook_file_parse_full_name:nN #1
          \__filehook_drop_extension_aux:nnn
      }
  }
\cs_new:Npn \__filehook_drop_extension_aux:nnn #1 #2 #3
   { \tl_if_empty:nF {#1} { #1 / } #2 }
```

(*End of definition for* `\__filehook_if_no_extension:nTF` *and* `\__filehook_drop_extension:N`.)

`\g__filehook_input_file_seq`
`\l__filehook_internal_tl`
`\__filehook_file_push:`
`\__filehook_file_pop:`
`\__filehook_file_pop_assign:nnnn`

Yet another stack, to keep track of `\CurrentFile` and `\CurrentFilePath` with nested `\inputs`. At the beginning of `\InputIfFileExists`, the current value of `\CurrentFilePath` and `\CurrentFile` is pushed to `\g__filehook_input_file_seq`, and at the end, it is popped and the value reassigned. Some other places don't use `\InputIfFileExists` directly (`\include`) or need `\CurrentFile` earlier (`\@onefilewithoptions`), so these are manually used elsewhere as well.

```
\tl_new:N \l__filehook_internal_tl
\seq_if_exist:NF \g__filehook_input_file_seq
  { \seq_new:N \g__filehook_input_file_seq }
\cs_new_protected:Npn \__filehook_file_push:
  {
    \seq_gpush:Nx \g__filehook_input_file_seq
      {
        { \CurrentFilePathUsed } { \CurrentFileUsed }
        { \CurrentFilePath     } { \CurrentFile     }
      }
  }
\cs_new_protected:Npn \__filehook_file_pop:
  {
    \seq_gpop:NNTF \g__filehook_input_file_seq \l__filehook_internal_tl
      { \exp_after:wN \__filehook_file_pop_assign:nnnn \l__filehook_internal_tl }
      {
        \msg_error:nnn { latex2e } { should-not-happen }
          { Tried~to~pop~from~an~empty~file~name~stack. }
      }
  }
\cs_new_protected:Npn \__filehook_file_pop_assign:nnnn #1 #2 #3 #4
  {
    \tl_set:Nn \CurrentFilePathUsed {#1}
    \tl_set:Nn \CurrentFileUsed {#2}
    \tl_set:Nn \CurrentFilePath {#3}
    \tl_set:Nn \CurrentFile {#4}
  }
\ExplSyntaxOff
```

(*End of definition for* `\g__filehook_input_file_seq` *and others.*)

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

When rolling forward the following expl3 functions may not be defined. If we roll back the code does nothing.

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                 {\file_parse_full_name_apply:nN}{Roll forward help}%
⟨latexrelease⟩
⟨latexrelease⟩\ExplSyntaxOn
⟨latexrelease⟩\cs_if_exist:NF\file_parse_full_name_apply:nN
⟨latexrelease⟩{
⟨latexrelease⟩\cs_new:Npn \file_parse_full_name_apply:nN #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \exp_args:Ne \__file_parse_full_name_auxi:nN
⟨latexrelease⟩      { \__kernel_file_name_sanitize:n {#1} }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new:Npn \__file_parse_full_name_auxi:nN #1
⟨latexrelease⟩  {
⟨latexrelease⟩    \__file_parse_full_name_area:nw { } #1
⟨latexrelease⟩      / \s__file_stop
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new:Npn \__file_parse_full_name_area:nw #1 #2 / #3 \s__file_stop
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_if_empty:nTF {#3}
⟨latexrelease⟩      { \__file_parse_full_name_base:nw { } #2 . \s__file_stop {#1} }
⟨latexrelease⟩      { \__file_parse_full_name_area:nw { #1 / #2 }
⟨latexrelease⟩                                      #3 \s__file_stop }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new:Npn \__file_parse_full_name_base:nw #1 #2 . #3 \s__file_stop
⟨latexrelease⟩  {
⟨latexrelease⟩    \tl_if_empty:nTF {#3}
⟨latexrelease⟩      {
⟨latexrelease⟩        \tl_if_empty:nTF {#1}
⟨latexrelease⟩          {
⟨latexrelease⟩            \tl_if_empty:nTF {#2}
⟨latexrelease⟩              { \__file_parse_full_name_tidy:nnnN { } { } }
⟨latexrelease⟩              { \__file_parse_full_name_tidy:nnnN { .#2 } { } }
⟨latexrelease⟩          }
⟨latexrelease⟩          { \__file_parse_full_name_tidy:nnnN {#1} { .#2 } }
⟨latexrelease⟩      }
⟨latexrelease⟩      { \__file_parse_full_name_base:nw { #1 . #2 }
⟨latexrelease⟩                                      #3 \s__file_stop }
⟨latexrelease⟩  }
⟨latexrelease⟩\cs_new:Npn \__file_parse_full_name_tidy:nnnN #1 #2 #3 #4
⟨latexrelease⟩  {
⟨latexrelease⟩    \exp_args:Nee #4
⟨latexrelease⟩      {
⟨latexrelease⟩        \str_if_eq:nnF {#3} { / } { \use_none:n }
⟨latexrelease⟩        #3 \prg_do_nothing:
⟨latexrelease⟩      }
⟨latexrelease⟩      { \use_none:n #1 \prg_do_nothing: }
⟨latexrelease⟩      {#2}
⟨latexrelease⟩  }
⟨latexrelease⟩}
⟨latexrelease⟩\ExplSyntaxOff
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
⟨@@=⟩
```

2.3 Declaring the file-related hooks

These hooks have names with three-parts that start with `file/`, `include/`, `class/` or `package/` and end with `/before` or `/after` (or `/end` in the case of `include/`). They are all generic hooks so will be declared only if code is added to them; this declaration is done for you automatically and, indeed, they should not be declared explicitly.

Those named `.../after` and `include/.../end` are, when code is added, declared as reversed hooks.

2.4 Patching LaTeX's `\InputIfFileExists` command

Most of what we have to do is adding `\UseHook` into several LaTeX 2_ε core commands, because of some circular dependencies in the kernel we do this only now and not in `ltfiles`.

`\InputIfFileExists`
`\@input@file@exists@with@hooks`
`\unqu@tefilef@und`

`\InputIfFileExists` loads any file if it is available so we have to add the hooks `file/before` and `file/after` in the right places. If the file doesn't exist no hooks should be executed.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩          {\InputIfFileExists}{Hook management (files)}%
⟨*2ekernel | latexrelease⟩
```

```
\let\InputIfFileExists\@undefined
\DeclareRobustCommand \InputIfFileExists[2]{%
  \IfFileExists{#1}%
    {%
      \@expl@@@filehook@file@push@@
      \@filehook@set@CurrentFile
```

We pre-expand `\@filef@und` so that in case another file is loaded in the true branch of `\InputIfFileExists`, these don't change their value meanwhile. This isn't a worry with `\CurrentFile...` because they are kept in a stack.

```
      \expandafter\@swaptwoargs\expandafter
        {\expandafter\@input@file@exists@with@hooks
          \expandafter{\@filef@und}}%
        {#2}%
      \@expl@@@filehook@file@pop@@
    }%
}
\def\@input@file@exists@with@hooks#1{%
```

If the file exists then `\CurrentFile` holds its name. But we can't rely on that still being true after the file has been processed. Thus for using the name in the file hooks we need to preserve the name and then restore it for the `file/.../after` hook.

The hook always refers to the file requested by the user. The hook is *always* loaded for `\CurrentFile` which usually is the same as `\CurrentFileUsed`. In the case of a file replacement, the `\CurrentFileUsed` holds the actual file loaded. In any case the file names are normalized so that the hooks work on the real file name, rather than what the user typed in.

expl3's `\file_full_name:n` normalizes the file name (to factor out differences in the `.tex` extension), and then does a file lookup to take into account a possible path from `\l_file_search_path_seq` and `\input@path`. However only the file name and extension

are returned so that file hooks can refer to the file by their name only. The path to the file is returned in `\CurrentFilePath`.

```
  \edef\reserved@a{%
    \@expl@@@filehook@file@pop@assign@@nnnn
      {\CurrentFilePathUsed}%
      {\CurrentFileUsed}%
      {\CurrentFilePath}%
      {\CurrentFile}}%
  \expandafter\@swaptwoargs\expandafter{\reserved@a}%
```

Before adding to the file list we need to make all (letter) characters catcode 11, because several packages use constructions like

```
\filename@parse{<filename>}
\ifx\filename@ext\@clsextension
  ...
\fi
```

and that doesn't work if `\filename@ext` is `\detokenized`. Making `\@clsextension` a string doesn't help much because some packages define their own `\<prefix>@someextension` with normal catcodes. This is not entirely correct because packages loaded (somehow) with catcode 12 alphabetic tokens (say, as the result of a `\string` or `\detokenize` command, or from a TEX string like `\jobname`) will have these character tokens incorrectly turned into letter tokens. This however is rare, so we'll go for the all-letters approach (grepping the packages in TEX Live didn't bring up any obvious candidate for breaking with this catcode change).

```
    {\edef\reserved@a{\unqu@tefilef@und#1\@nil}%
     \@addtofilelist{\string@makeletter\reserved@a}%
     \UseHook{file/before}%
```

The current file name is available in `\CurrentFile` so we use that in the specific hook.

```
     \UseHook{file/\CurrentFile/before}%
     \@@input #1% <- trailing space comes from \@filef@und
    }%
```

And here, `\CurrentFile` is restored (by `\@expl@@@filehook@file@pop@assign@@nnnn`) so we can use it once more.

```
  \UseHook{file/\CurrentFile/after}%
  \UseHook{file/after}}
\def\unqu@tefilef@und"#1" \@nil{#1}
```

Now declare the non-generic file hooks used above:

```
\NewHook{file/before}
\NewReversedHook{file/after}
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
```

Now define `\InputIfFileExists` to input `#1` if it seems to exist. Immediately prior to the input, `#2` is executed. If the file `#1` does not exist, execute '`#3`'.

```
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩          {\InputIfFileExists}{Hook management (files)}%
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand \InputIfFileExists[2]{%
⟨latexrelease⟩  \IfFileExists{#1}%
⟨latexrelease⟩    {%
```

```
⟨latexrelease⟩  \expandafter\@swaptwoargs\expandafter
⟨latexrelease⟩      {\@filef@und}{#2\@addtofilelist{#1}\@@input}}}
⟨latexrelease⟩\let\@input@file@exists@with@hooks\@undefined
⟨latexrelease⟩\let\unqu@tefilef@und\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩          {\InputIfFileExists}{Hook management (files)}%
⟨latexrelease⟩\long\def \InputIfFileExists#1#2{%
⟨latexrelease⟩  \IfFileExists{#1}%
⟨latexrelease⟩    {#2\@addtofilelist{#1}\@@input \@filef@und}}
```

Also undo the internal command as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```
⟨latexrelease⟩\expandafter\let\csname InputIfFileExists \endcsname\@undefined
```

```
⟨latexrelease⟩\let\@input@file@exists@with@hooks\@undefined
⟨latexrelease⟩\let\unqu@tefilef@und\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\InputIfFileExists`, `\@input@file@exists@with@hooks`, *and* `\unqu@tefilef@und`.)

2.5 Declaring a file substitution

```
⟨@@=filehook⟩
```

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩          {\__filehook_subst_add:nn}{Declaring file substitution}%
\ExplSyntaxOn
```

`\__filehook_subst_add:nn`
`\__filehook_subst_remove:n`
`\__filehook_subst_file_normalize:Nn`
`\__filehook_subst_empty_name_chk:NN`

`\__filehook_subst_add:nn` declares a file substitution by doing a (global) definition of the form `\def\@file-subst@`⟨*file*⟩`{`⟨*replacement*⟩`}`. The file names are properly sanitised, and normalized with the same treatment done for the file hooks. That is, a file replacement is declared by using the file name (and extension, if any) only, and the file path should not be given. If a file name is empty it is replaced by `.tex` (the empty csname is used to check that).

```
\cs_new_protected:Npn \__filehook_subst_add:nn #1 #2
  {
    \group_begin:
      \cs_set:cpx { } { \exp_not:o { \cs:w\cs_end: } }
      \int_set:Nn \tex_escapechar:D { -1 }
      \cs_gset:cpx
        {
          @file-subst@
          \__filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
        }
        { \__filehook_subst_file_normalize:Nn \__filehook_file_name_compose:nnn
                                        {#2} }
    \group_end:
  }
\cs_new_protected:Npn \__filehook_subst_remove:n #1
  {
```

```
    \group_begin:
      \cs_set:cpx { } { \exp_not:o { \cs:w\cs_end: } }
      \int_set:Nn \tex_escapechar:D { -1 }
      \cs_undefine:c
        {
          @file-subst@
          \__filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
        }
    \group_end:
  }
\cs_new:Npn \__filehook_subst_file_normalize:Nn #1 #2
  {
    \exp_after:wN \__filehook_subst_empty_name_chk:NN
      \cs:w \exp_after:wN \cs_end:
        \cs:w \__filehook_file_parse_full_name:nN {#2} #1 \cs_end:
  }
\cs_new:Npn \__filehook_subst_empty_name_chk:NN #1 #2
  { \if_meaning:w #1 #2 .tex \else: \token_to_str:N #2 \fi: }
```

(*End of definition for* `\__filehook_subst_add:nn` *and others.*)

`\use_ii_iii:nnn` A variant of `\use_...` to discard the first of three arguments.

Todo: this should move to expl3

```
\cs_gset:Npn \use_ii_iii:nnn #1 #2 #3 {#2 #3}
```

(*End of definition for* `\use_ii_iii:nnn`.)

```
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

`\declare@file@substitution` `\undeclare@file@substitution` For two internals we provide LaTeX 2_ε names so that we can use them elsewhere in the kernel (and so that they can be used in packages if really needed, e.g., scrlfile).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩          {\declare@file@substitution}{File substitution}%
\ExplSyntaxOn
\cs_new_eq:NN \declare@file@substitution   \__filehook_subst_add:nn
\cs_new_eq:NN \undeclare@file@substitution \__filehook_subst_remove:n
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

We are not fully rolling back the file substitutions in case a rollback encounters a package that contains them, but is itself not setup for rollback. So we just bypass them and hope for the best.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩          {\declare@file@substitution}{File substitution}%
⟨latexrelease⟩
⟨latexrelease⟩\let \declare@file@substitution   \@gobbletwo
⟨latexrelease⟩\let \undeclare@file@substitution \@gobble
⟨latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\declare@file@substitution` *and* `\undeclare@file@substitution`*. These functions are documented on page 1040.*)

```
⟨@@=⟩
```

2.6 Selecting a file (`\set@curr@file`)

`\set@curr@file`
`\set@curr@file@nosearch`
`\@curr@file`
`\@curr@file@reqd`

Now we hook into `\set@curr@file` to resolve a possible file substitution, and add `\@expl@@@filehook@set@curr@file@@nNN` at the end, after `\@curr@file` is set.

A file name is built using `\expandafter\string\csname`⟨*filename*⟩`\endcsname` to avoid expanding utf8 active characters. The `\csname` expands the normalization machinery and the routine to resolve a file substitution, returning a control sequence with the same name as the file.

It happens that when ⟨*filename*⟩ is empty, the generated control sequence is `\csname\endcsname`, and doing `\string` on that results in the file `csnameendcsname.tex`. To guard against that we `\ifx`-compare the generated control sequence with the empty csname. To do so, `\csname\endcsname` has to be defined, otherwise it would be equal to `\relax` and we would have false positives. Here we define `\csname\endcsname` to expand to itself to avoid it matching the definition of some other control sequence.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩          {\set@curr@file}{Setting current file name}%
\def\set@curr@file{%
  \begingroup
    \set@curr@file@aux}
\edef\set@curr@file@nosearch{%
  \begingroup
    \let\noexpand\input@path\noexpand\@empty
    \csname seq_clear:N\endcsname
      \expandafter\noexpand\csname l_file_search_path_seq\endcsname
    \noexpand\set@curr@file@aux}
\def\set@curr@file@aux#1{%
    \escapechar\m@ne
    \let\protect\string
    \edef~{\string~}%
    \expandafter\def\csname\expandafter\endcsname
      \expandafter{\csname\endcsname}%
```

Two file names are set here: `\@curr@file@reqd` which is the file requested by the user, and `\@curr@file` which should be the same, except when we have a file substitution, in which case it holds the actual loaded file. `\@curr@file` is resolved first, to check if a substitution happens. If it doesn't, `\@expl@@@filehook@if@file@replaced@@TF` short-cuts and just copies `\@curr@file`, otherwise the full normalization procedure is executed.

At this stage the file name is parsed and normalized, but if the input doesn't have an extension, the default `.tex` is *not* added to `\@curr@file` because for applications other than `\input` (graphics, for example) the default extension may not be `.tex`. First check if the input has an extension, then if the input had no extension, call

\@expl@@@filehook@drop@extension@@N. In case of a file substitution, \@curr@file will have an extension.

```
    \@expl@@@filehook@if@no@extension@@nTF{#1}%
      {\@tempswatrue}{\@tempswafalse}%
    \@kernel@make@file@csname\@curr@file
      \@expl@@@filehook@resolve@file@subst@@w {#1}%
    \@expl@@@filehook@if@file@replaced@@TF
      {\@kernel@make@file@csname\@curr@file@reqd
         \@expl@@@filehook@normalize@file@name@@w{#1}%
       \if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
      {\if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file \fi
       \global\let\@curr@file@reqd\@curr@file}%
    \@expl@@@filehook@clear@replacement@flag@@
  \endgroup}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩          {\set@curr@file}{Setting current file name}%
⟨latexrelease⟩\def\set@curr@file#1{%
⟨latexrelease⟩  \begingroup
⟨latexrelease⟩    \escapechar\m@ne
⟨latexrelease⟩    \let\protect\string
⟨latexrelease⟩    \edef~{\string~}%
⟨latexrelease⟩    \expandafter\def\csname\expandafter\endcsname
⟨latexrelease⟩      \expandafter{\csname\endcsname}%
⟨latexrelease⟩    \@expl@@@filehook@if@no@extension@@nTF{#1}%
⟨latexrelease⟩      {\@tempswatrue}{\@tempswafalse}%
⟨latexrelease⟩    \@kernel@make@file@csname\@curr@file
⟨latexrelease⟩      \@expl@@@filehook@resolve@file@subst@@w {#1}%
⟨latexrelease⟩    \@expl@@@filehook@if@file@replaced@@TF
⟨latexrelease⟩      {\@kernel@make@file@csname\@curr@file@reqd
⟨latexrelease⟩         \@expl@@@filehook@normalize@file@name@@w{#1}%
⟨latexrelease⟩       \if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
⟨latexrelease⟩      {\if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file \fi
⟨latexrelease⟩       \global\let\@curr@file@reqd\@curr@file}%
⟨latexrelease⟩    \@expl@@@filehook@clear@replacement@flag@@
⟨latexrelease⟩  \endgroup}
⟨latexrelease⟩\let\set@curr@file@nosearch\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩          {\set@curr@file}{Setting current file name}%
⟨latexrelease⟩\def\set@curr@file#1{%
⟨latexrelease⟩  \begingroup
⟨latexrelease⟩    \escapechar\m@ne
⟨latexrelease⟩    \expandafter\def\csname\expandafter\endcsname
⟨latexrelease⟩      \expandafter{\csname\endcsname}%
⟨latexrelease⟩    \@expl@@@filehook@if@no@extension@@nTF{#1}%
⟨latexrelease⟩      {\@tempswatrue}{\@tempswafalse}%
⟨latexrelease⟩    \@kernel@make@file@csname\@curr@file
⟨latexrelease⟩      \@expl@@@filehook@resolve@file@subst@@w {#1}%
⟨latexrelease⟩    \@expl@@@filehook@if@file@replaced@@TF
⟨latexrelease⟩      {\@kernel@make@file@csname\@curr@file@reqd
⟨latexrelease⟩         \@expl@@@filehook@normalize@file@name@@w{#1}%
```

```
⟨latexrelease⟩        \if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
⟨latexrelease⟩       {\if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file \fi
⟨latexrelease⟩        \global\let\@curr@file@reqd\@curr@file}%
⟨latexrelease⟩     \@expl@@@filehook@clear@replacement@flag@@
⟨latexrelease⟩   \endgroup}
⟨latexrelease⟩\let\set@curr@file@nosearch\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
⟨latexrelease⟩          {\set@curr@file}{Setting current file name}%
⟨latexrelease⟩\def\set@curr@file#1{%
⟨latexrelease⟩   \begingroup
⟨latexrelease⟩     \escapechar\m@ne
⟨latexrelease⟩     \xdef\@curr@file{%
⟨latexrelease⟩       \expandafter\expandafter\expandafter\unquote@name
⟨latexrelease⟩       \expandafter\expandafter\expandafter{%
⟨latexrelease⟩       \expandafter\string
⟨latexrelease⟩         \csname\@firstofone#1\@empty\endcsname}}%
⟨latexrelease⟩   \endgroup
⟨latexrelease⟩}
⟨latexrelease⟩\let\set@curr@file@nosearch\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩          {\set@curr@file}{Setting current file name}%
⟨latexrelease⟩\let\set@curr@file\@undefined
⟨latexrelease⟩\let\set@curr@file@nosearch\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\set@curr@file` *and others.*)

`\@filehook@set@CurrentFile`
`\@kernel@make@file@csname`
`\@set@curr@file@aux`

Todo: This should get internalized using `@expl@` *names*

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩          {\@kernel@make@file@csname}{Make file csname}%
```

```
\def\@kernel@make@file@csname#1#2#3{%
  \xdef#1{\expandafter\@set@curr@file@aux
    \csname\expandafter#2\@firstofone#3\@nil\endcsname}}
```

This auxiliary compares `\⟨filename⟩` with `\csname\endcsname` to check if the empty `.tex` file was requested.

```
\long\def\@set@curr@file@aux#1{%
  \expandafter\ifx\csname\endcsname#1%
    .tex\else\string#1\fi}
```

Then we call `\@expl@@@filehook@set@curr@file@@nNN` once for `\@curr@file` to set `\CurrentFile(Path)Used` and once for `\@curr@file@reqd` to set `\CurrentFile(Path)`. Here too the slower route is only used if a substitution happened, but here `\@expl@@@filehook@if@file@replaced@@TF` can't be used because the flag is reset at the `\endgroup` above, so we check if `\@curr@file` and `\@curr@file@reqd` differ. This macro is issued separate from `\set@curr@file` because it changes `\CurrentFile`, and side-effects would quickly get out of control.

```
\def\@filehook@set@CurrentFile{%
```

```
  \@expl@@@filehook@set@curr@file@@nNN{\@curr@file}%
    \CurrentFileUsed\CurrentFilePathUsed
  \ifx\@curr@file@reqd\@curr@file
    \let\CurrentFile\CurrentFileUsed
    \let\CurrentFilePath\CurrentFilePathUsed
  \else
    \@expl@@@filehook@set@curr@file@@nNN{\@curr@file@reqd}%
      \CurrentFile\CurrentFilePath
  \fi}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@filehook@set@CurrentFile`, `\@kernel@make@file@csname`, *and* `\@set@curr@file@aux`.)

`\@@_set_curr_file:nNN`
`\@@_set_curr_file_assign:nnnNN`

When inputting a file, `\set@curr@file` does a file lookup (in `\input@path` and `\l_file_search_path_seq`) and returns the actual file name (⟨*base*⟩ plus ⟨*ext*⟩) in `\CurrentFileUsed`, and in case there's a file substitution, the requested file in `\CurrentFile` (otherwise both are the same). Only the base and extension are returned, regardless of the input (both `path/to/file.tex` and `file.tex` end up as `file.tex` in `\CurrentFile`). The path is returned in `\CurrentFilePath`, in case it's needed.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩          {@@_set_curr_file:nNN}{Set curr file}%
\ExplSyntaxOn
⟨@@=filehook⟩
\cs_new_protected:Npn \__filehook_set_curr_file:nNN #1
  {
    \exp_args:Nf \__filehook_file_parse_full_name:nN {#1}
      \__filehook_set_curr_file_assign:nnnNN
  }
\cs_new_protected:Npn \__filehook_set_curr_file_assign:nnnNN #1 #2 #3 #4 #5
  {
    \str_set:Nn #5 {#1}
    \str_set:Nn #4 {#2#3}
  }
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@@_set_curr_file:nNN` *and* `\@@_set_curr_file_assign:nnnNN`.)

2.7 Replacing a file and detecting loops

`\__filehook_resolve_file_subst:w`
`\__filehook_normalize_file_name:w`
`\__filehook_file_name_compose:nnn`

Start by sanitizing the file with `\__filehook_file_parse_full_name:nN` then do `\__filehook_file_subst_begin:nnn{`⟨*path*⟩`}{`⟨*name*⟩`}{`⟨*ext*⟩`}`.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩     {\__filehook_resolve_file_subst:w}{Replace files detect loops}%
\ExplSyntaxOn
```

```
\cs_new:Npn \__filehook_resolve_file_subst:w #1 \@nil
  { \__filehook_file_parse_full_name:nN {#1} \__filehook_file_subst_begin:nnn }
\cs_new:Npn \__filehook_normalize_file_name:w #1 \@nil
  { \__filehook_file_parse_full_name:nN {#1} \__filehook_file_name_compose:nnn }
\cs_new:Npn \__filehook_file_name_compose:nnn #1 #2 #3
  { \tl_if_empty:nF {#1} { #1 / } #2#3 }
```

`__filehook_file_replaced`
`\__filehook_if_file_replaced:TF`
`\__filehook_clear_replacement_flag:`

Since the file replacement is done expandably in a `\csname`, use a flag to remember if a substitution happened. We use this in `\set@curr@file` to short-circuit some of it in case no substitution happened (by far the most common case, so it's worth optimizing). The flag raised during the file substitution algorithm must be explicitly cleared after the `\__filehook_if_file_replaced:TF` conditional is no longer needed, otherwise further uses of `\__filehook_if_file_replaced:TF` will wrongly return true.

```
\flag_new:n { __filehook_file_replaced }
\cs_new:Npn \__filehook_if_file_replaced:TF #1 #2
  { \flag_if_raised:nTF { __filehook_file_replaced } {#1} {#2} }
\cs_new_protected:Npn \__filehook_clear_replacement_flag:
  { \flag_clear:n { __filehook_file_replaced } }
```

`\__filehook_file_subst_begin:nnn`

First off, start by checking if the current file (⟨*name*⟩ + ⟨*ext*⟩) has a declared substitution. If not, then just put that as the name (including a possible ⟨*path*⟩ in this case): this is the default case with no substitutions, so it's the first to be checked. The auxiliary `\__filehook_file_subst_tortoise_hare:nn` sees that there's no replacement for `#2#3` and does nothing else.

```
\cs_new:Npn \__filehook_file_subst_begin:nnn #1 #2 #3
  {
    \__filehook_file_subst_tortoise_hare:nn { #2#3 } { #2#3 }
      { \__filehook_file_name_compose:nnn {#1} {#2} {#3} }
  }
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

2.7.1 The Tortoise and Hare algorithm

`\__filehook_file_subst_tortoise_hare:nn`
`\__filehook_file_subst_loop:NN`
`\__filehook_file_subst_loop:cc`

If there is a substitution (⟨*true*⟩ in the first `\cs_if_exist:cTF` below), then first check if there is no substitution down the line: this should be the second most common case, of one file replaced by another. In that case just leave the substitution there and the job is done. If any substitution happens, then the `\flag __filehook_file_replaced` is raised (conditionally, because checking if a flag is raised is much faster than raising it over and over again).

If, however there are more substitutions, then we need to check for a possible loop in the substitutions, which would otherwise put TEX in an infinite loop if just an exhaustive expansion was used.

To detect a loop, the *Tortoise and Hare* algorithm is used. The name of the algorithm is an analogy to Aesop's fable, in which the Hare outruns a Tortoise. The two pointers here are the csnames which contains each file replacement, both of which start at the position zero, which is the file requested. In the inner part of the macro below, `\__filehook_file_subst_loop:cc` is called with `\@file-subst@`⟨*file*⟩ and `\@file-subst@\@file-subst@`⟨*file*⟩; that is, the substitution of ⟨*file*⟩ and the substitution of that substitution: the Tortoise walks one step while the Hare walks two.

Within `\__filehook_file_subst_loop:NN` the two substitutions are compared, and if they lead to the same file it means that there is a loop in the substitutions. If there's no loop, `\__filehook_file_subst_tortoise_hare:nn` is called again with the Tortoise at position 1 and the hare at 2. Again, the substitutions are checked ahead of the Hare pointer to check that it won't run too far; in case there is no loop in the declarations, eventually one of the `\cs_if_exist:cTF` below will go ⟨*false*⟩ and the algorithm will end; otherwise it will run until the Hare reaches the same spot as the tortoise and a loop is detected.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩  {\__filehook_file_subst_tortoise_hare:nn}{Tortoise and Hare}%
\ExplSyntaxOn
\cs_new:Npn \__filehook_file_subst_tortoise_hare:nn #1 #2 #3
  {
    \cs_if_exist:cTF { @file-subst@ #2 }
      {
        \flag_if_raised:nF { __filehook_file_replaced }
          { \flag_raise:n { __filehook_file_replaced } }
        \cs_if_exist:cTF { @file-subst@ \use:c { @file-subst@ #2 } }
          {
            \__filehook_file_subst_loop:cc
              { @file-subst@ #1 }
              { @file-subst@ \use:c { @file-subst@ #2 } }
          }
          { \use:c { @file-subst@ #2 } }
      }
      { #3 }
  }
```

This is just an auxiliary to check if a loop was found, and continue the algorithm otherwise. If a loop is found, the `.tex` file is used as fallback and `\__filehook_file_subst_cycle_error:cN` is called to report the error.

```
\cs_new:Npn \__filehook_file_subst_loop:NN #1 #2
  {
    \token_if_eq_meaning:NNTF #1 #2
      {
        .tex
        \__filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #1
      }
      { \__filehook_file_subst_tortoise_hare:nn {#1} {#2} {#2} }
  }
\cs_generate_variant:Nn \__filehook_file_subst_loop:NN { cc }
```

Showing this type of error expandably is tricky, as we have a very limited amount of characters to show and a potentially large list. As a work around, several errors are printed, each showing one step of the loop, until all the error messages combined show the loop.

`\__filehook_file_subst_cycle_error:NN`
`\__filehook_file_subst_cycle_error:cN`

```
\cs_new:Npn \__filehook_file_subst_cycle_error:NN #1 #2
  {
    \msg_expandable_error:nnff { latex2e } { file-cycle }
      {#1} { \use:c { @file-subst@ #1 } }
    \token_if_eq_meaning:NNF #1 #2
```

```
      { \__filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #2 }
  }
\cs_generate_variant:Nn \__filehook_file_subst_cycle_error:NN { c }
```

And the error message:

```
\msg_new:nnn { latex2e } { file-cycle }
  { File~loop!~#1~replaced~by~#2... }
```

(*End of definition for* `\__filehook_resolve_file_subst:w` *and others.*)

```
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
⟨@@=⟩
```

2.8 Preventing a package from loading

We support the use case of preventing a package from loading but not any other type of files (e.g., classes).

`\disable@package@load`
`\reenable@package@load`
`\@disable@packageload@do`

`\disable@package@load` defines `\@pkg-disable@`⟨*package*⟩ to expand to some code `#2` instead of loading the package.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩          {\disable@package@load}{Disable packages}%
\def\disable@package@load#1#2{%
  \global\@namedef{@pkg-disable@#1.\@pkgextension}{#2}}
```

Here we check if a control sequence named `\@pkg-disable@`⟨*name*⟩`.sty` is defined, and if so don't use the package loading code `#2`, but use the replacement code stored in that control sequence, write something to the log, and then prevent `\@onefilewithoptions` from sanity-checking the requested package date (the `\expandafter` here triggers one in `\@onefilewithoptions` that ends a conditional there, and the `\@gobbletwo` removes the date checking code from the input stream).

```
\def\@disable@packageload@do#1#2{%
  \@ifundefined{@pkg-disable@#1}%
     {#2}%
     {\@nameuse{@pkg-disable@#1}%
      \@latex@info{Package '#1' has been disabled.%
       \MessageBreak Load request ignored}%
      \expandafter\@gobbletwo}}
```

`\reenable@package@load` undefines `\@pkg-disable@`⟨*package*⟩ to reallow loading a package.

```
\def\reenable@package@load#1{%
  \global\expandafter\let
  \csname @pkg-disable@#1.\@pkgextension \endcsname \@undefined}
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩          {\disable@package@load}{Disable packages}%
⟨latexrelease⟩
```

```
⟨latexrelease⟩\let\disable@package@load   \@undefined
⟨latexrelease⟩\let\@disable@packageload@do\@undefined
⟨latexrelease⟩\let\reenable@package@load  \@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\disable@package@load`, `\reenable@package@load`, *and* `\@disable@packageload@do`. *These functions are documented on page* 1040.)

2.9 High-level interfaces for LaTeX

None so far and the general feeling for now is that the hooks are enough. Packages like filehook, etc., may use them to set up their interfaces (samples are given below) but for the now the kernel will not provide any.

2.10 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) LaTeX 2_ε names to allow for internal commands to be used outside this module (and in parts that still use LaTeX 2_ε syntax. We have to unset the `@@` since we want double "at" sign in place of double underscores.

```
⟨@@=⟩
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩    {\@expl@@@filehook@if@no@extension@@nTF}{2e tmp interfaces}%
\ExplSyntaxOn
\cs_new_eq:NN \@expl@@@filehook@if@no@extension@@nTF
              \__filehook_if_no_extension:nTF
\cs_new_eq:NN \@expl@@@filehook@set@curr@file@@nNN
              \__filehook_set_curr_file:nNN
\cs_new_eq:NN \@expl@@@filehook@resolve@file@subst@@w
              \__filehook_resolve_file_subst:w
\cs_new_eq:NN \@expl@@@filehook@normalize@file@name@@w
              \__filehook_normalize_file_name:w
\cs_new_eq:NN \@expl@@@filehook@if@file@replaced@@TF
              \__filehook_if_file_replaced:TF
\cs_new_eq:NN \@expl@@@filehook@clear@replacement@flag@@
              \__filehook_clear_replacement_flag:
\cs_new_eq:NN \@expl@@@filehook@drop@extension@@N
              \__filehook_drop_extension:N
\cs_new_eq:NN \@expl@@@filehook@file@push@@
              \__filehook_file_push:
\cs_new_eq:NN \@expl@@@filehook@file@pop@@
              \__filehook_file_pop:
\cs_new_eq:NN \@expl@@@filehook@file@pop@assign@@nnnn
              \__filehook_file_pop_assign:nnnn
\ExplSyntaxOff
```

This one specifically has to be undefined because it is left over in the input stream from `\InputIfFileExists` and executed when latexrelease is loaded. It cannot be `\let` to `\@undefined` otherwise it would error as well, so it is `\let` to `\relax` to be silently ignored when loading `\latexrelease`.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩    {\@expl@@@filehook@if@no@extension@@nTF}{2e tmp interfaces}%
⟨latexrelease⟩\let\@expl@@@filehook@file@pop@@\relax
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

This ends the kernel code in this file.

```
⟨/2ekernel⟩
```

3 A sample package for structuring the log output

```
⟨*structuredlog⟩
⟨@@=filehook⟩
```

```
\ProvidesExplPackage
    {structuredlog}{\ltfilehookdate}{\ltfilehookversion}
    {Structuring the TeX transcript file}
```

\g__filehook_nesting_level_int

Stores the current package nesting level.

```
\int_new:N \g__filehook_nesting_level_int
```

Initialise the counter with the number of files in the `\@currnamestack` (the number of items divided by 3) minus one, because this package is skipped when printing to the log.

```
\int_gset:Nn \g__filehook_nesting_level_int
  { ( \tl_count:N \@currnamestack ) / 3 - 1 }
```

(*End of definition for* `\g__filehook_nesting_level_int`.)

__filehook_log_file_record:n

This macro is responsible for increasing and decreasing the file nesting level, as well as printing to the log. The argument is either `STOPTART` or `STOP` and the action it takes on the nesting integer depends on that.

```
\cs_new_protected:Npn \__filehook_log_file_record:n #1
  {
    \str_if_eq:nnT {#1} {START} { \int_gincr:N \g__filehook_nesting_level_int }
    \iow_term:x
      {
        \prg_replicate:nn { \g__filehook_nesting_level_int } { = } ~
        ( LEVEL ~ \int_use:N \g__filehook_nesting_level_int \c_space_tl #1 ) ~
        \CurrentFileUsed
```

If there was a file replacement, show that as well:

```
        \str_if_eq:NNF \CurrentFileUsed \CurrentFile
          { ~ ( \CurrentFile \c_space_tl requested ) }
        \iow_newline:
      }
    \str_if_eq:nnT {#1} {STOP} { \int_gdecr:N \g__filehook_nesting_level_int }
  }
```

Now just hook the macro above in the generic `file/before`...

```
\AddToHook{file/before}{ \__filehook_log_file_record:n { START } }
```

...and `file/after` hooks. We don't want to install the `file/after` hook immediately, because that would mean it is the first time executed when the package finishes. We therefore put the declaration inside `\AddToHookNext` so that it gets only installed when we have left this package.

```
\AddToHookNext{file/after}
  { \AddToHook{file/after}{ \__filehook_log_file_record:n { STOP } } }
```

(*End of definition for* `\__filehook_log_file_record:n`.)

```
⟨@@=⟩
⟨/structuredlog⟩
```

4 Package emulations

4.1 Package **atveryend** emulation

With the new hook management and the hooks in `\enddocument` all of atveryend is taken care of. We can make an emulation only here after the substitution functionality is available:

```
⟨*2ekernel⟩
\declare@file@substitution{atveryend.sty}{atveryend-ltx.sty}
⟨/2ekernel⟩
```

Here is the package file we point to:

```
⟨*atveryend-ltx⟩
\ProvidesPackage{atveryend-ltx}
   [2020/08/19 v1.0a
     Emulation of the original atveryend package^^Jwith kernel methods]
```

Here are new definitions for its interfaces now pointing to the hooks in `\enddocument`

```
\newcommand\AfterLastShipout  {\AddToHook{enddocument/afterlastpage}}
\newcommand\AtVeryEndDocument {\AddToHook{enddocument/afteraux}}
```

Next one is a bit of a fake, but the result should normally be as expected. If not, one needs to add a rule to sort the code chunks in `enddocument/info`.

```
\newcommand\AtEndAfterFileList{\AddToHook{enddocument/info}}
\newcommand\AtVeryVeryEnd     {\AddToHook{enddocument/end}}
```

`\BeforeClearDocument` This one is the only one we don't implement or rather don't have a dedicated hook in the code.

```
\ExplSyntaxOn
\newcommand\BeforeClearDocument[1]
  { \AtEndDocument{#1}
    \atveryend@DEPRECATED{BeforeClearDocument \tl_to_str:n{#1}}
  }
\cs_new:Npn\atveryend@DEPRECATED #1
   {\iow_term:x{======~DEPRECATED~USAGE~#1~==========}}
\ExplSyntaxOff
```

(*End of definition for* `\BeforeClearDocument`.)

```
⟨/atveryend-ltx⟩
```

File 53
ltshipout.dtx

1 Introduction

The code provides an interface to the `\shipout` primitive of TEX which is called when a finished pages is finally "shipped out" to the target output file, e.g., the `.dvi` or `.pdf` file. A good portion of the code is based on ideas by Heiko Oberdiek implemented in his packages atbegshi and atenddvi even though the interfaces are somewhat different.[53]

1.1 Overloading the `\shipout` primitive

`\shipout` With this implementation TEX's shipout primitive is no longer available for direct use. Instead `\shipout` is running some (complicated) code that picks up the box to be shipped out regardless of how that is done, i.e., as a constructed `\vbox` or `\hbox` or as a box register.

It then stores it in a named box register. This box can then be manipulated through a set of hooks after which it is shipped out for real.

Each shipout that actually happens (i.e., where the material is not discarded for one or the other reason) is recorded and the total number is available in a readonly variable and in a LATEX counter.

`\RawShipout` This command implements a simplified shipout that bypasses the foreground and background hooks, e.g., only `shipout/firstpage` and `shipout/lastpage` are executed and the total shipout counters are incremented.

The command doesn't use `\ShipoutBox` but its own private box register so that it can be used inside of shipout hooks to do some additional shipouts while already in the output routine with the current page being stored in `\ShipoutBox`. It does have access to `\ShipoutBox` if it is used in `shipout/before` (or `shipout/after`) and can use its content.

It is safe to use it in `shipout/before` or `shipout/after` but not necessarily in the other `shipout/...` hooks as they are intended for special processing.

[53] Heiko's interfaces are emulated by the kernel code, if a document requests his packages, so older documents will continue to work.

`\ShipoutBox`
`\l_shipout_box`

This box register is called `\ShipoutBox` (alternatively available via the L3 name `\l_shipout_box`).

This box is a "local" box and assignments to it should be done only locally. Global assignments (as done by some packages with older code where this is box is known as 255) may work but they are conceptually wrong and may result in errors under certain circumstances.

During the execution of `shipout/before` this box contains the accumulated material for the page, but not yet any material added by other shipout hooks. During execution of `shipout/after`, i.e., after the shipout has happened, the box also contains any background or foreground material.

Material from the hooks `shipout/firstpage` or `shipout/lastpage` is not included (but only used during the actual shipout) to facilitate reuse of the box data (e.g., `shipout/firstpage` material should never be added to a later page of the output).

`\l_shipout_box_ht_dim`
`\l_shipout_box_dp_dim`
`\l_shipout_box_wd_dim`
`\l_shipout_box_ht_plus_dp_dim`

The shipout box dimensions are available in the L3 registers `\l_shipout_box_ht_dim`, etc. (there are no LaTeX 2_ε names).[54] These variables can be used inside the hook code for `shipout/before`, `shipout/foreground` and `shipout/background` if needed.

1.2 Provided hooks

`shipout/before`
`shipout/after`
`shipout/foreground`
`shipout/background`
`shipout/firstpage`
`shipout/lastpage`

The code for `\shipout` offers a number of hooks into which packages (or the user) can add code to support different use cases. These are:

`shipout/before` This hook is executed after the finished page has been stored in `\ShipoutBox` / `\l_shipout_box`). It can be used to alter that box content or to discard it completely (see `\DiscardShipoutBox` below).

You can use `\RawShipout` inside this hook for special use cases. It can make use of `\ShipoutBox` (which doesn't yet include the background and foreground material).

Note: It is not possible (or say advisable) to try and use this hook to typeset material with the intention to return it to main vertical list, it will go wrong and give unexpected results in many cases—for starters it will appear after the current page not before or it will vanish or the vertical spacing will be wrong!

`shipout/background` This hook adds a picture environment into the background of the page with the `(0,0)` coordinate in the top-left corner using a `\unitlength` of `1pt`.

It should therefore only receive `\put` commands or other commands suitable in a `picture` environment and the vertical coordinate values would normally be negative.

[54] Might need changing, but HO's version as strings is not really helpful I think).

Technically this is implemented by adding a zero-sized `\hbox` as the very first item into the `\ShipoutBox` containing that `picture` environment. Thus the rest of the box content will overprint what ever is typeset by that hook.

`shipout/foreground` This hook adds a picture environment into the foreground of the page with the `(0,0)` coordinate in the top-left corner using a `\unitlength` of `1pt`.

Technically this is implemented by adding a zero-sized `\hbox` as the very last item into the `\ShipoutBox` and raising it up so that it still has its `(0,0)` point in the top-left corner. But being placed after the main box content it will be typeset later and thus overprints it (i.e., is in the foreground).

`shipout` This hook is executed after foreground and/or background material has been added, i.e., just in front of the actual shipout operation. Its purpose is to allow manipulation of the finalized box (stored in `\ShipoutBox`) with the extra material also in place (which is not yet the case in `shipout/before`).

It cannot be used to cancel the shipout operation via `\DiscardShipoutBox` (that has to happen in `shipout/before`, if desired!

`shipout/firstpage` The material from this hook is executed only once at the very beginning of the first output page that is shipped out (i.e., not discarded at the last minute). It should only contain `\special` or similar commands needed to direct post processors handling the `.dvi` or `.pdf` output.[55]

This hook is added to the very first page regardless of how it is shipped out (i.e., with `\shipout` or `\RawShipout`).

`shipout/lastpage` The corresponding hook to add `\specials` at the very end of the output file. It is only executed on the very last page of the output file — or rather on the page that LaTeX believes is the last one. Again it is executed regardless of the shipout method.

It may not be possible for LaTeX to correctly determine which page is the last one without several reruns. If this happens and the hook is non-empty then LaTeX will add an extra page to place the material and also request a rerun to get the correct placement sorted out.

`shipout/after` This hook is executed after a shipout has happened. If the shipout box is discarded this hook is not looked at.

You can use `\RawShipout` inside this hook for special use cases and the main `\ShipoutBox` is still available at this point (but in contrast to `shipout/before` it now includes the background and foreground material).

Note: Just like `shipout/before` this hook is not meant to be used for adding typeset material back to the main vertical list—it might vanish or the vertical spacing will be wrong!

As mentioned above the hook `shipout/before` is executed first and can manipulate the prepared shipout box stored in `\ShipoutBox` or set things up for use in `\write` during the actual shipout. It is even run if there was a `\DiscardShipoutBox` request in the document.

The other hooks (except `shipout` and `shipout/after`) are added inside hboxes to the box being shipped out in the following order:

[55] In LaTeX 2_ε that was already existing, but implemented using a box register with the name `\@begindvibox`.

`shipout/firstpage`	only on the first page
`shipout/background`	
⟨*boxed content of* `\ShipoutBox`⟩	
`shipout/foreground`	
`shipout/lastpage`	only on the last page

If any of the hooks has no code then the corresponding box is added at that point.

Once the (page) box has got the above extra content it can again be manipulated using the `shipout` hook and then is shipped out for real.

Once the (page) box has been shipped out the `shipout/after` hook is called (while you are still inside the output routine). It is not called if the shipout box was discarded.

In a document that doesn't produce pages, e.g., only makes `\typeout`s, none of the hooks are ever executed (as there is no `\shipout`) not even the `shipout/lastpage` hook.

If `\RawShipout` is used instead of `\shipout` then only the hooks `shipout/firstpage` and `shipout/lastpage` are executed (on the first or last page), all others are bypassed.

1.3 Legacy LaTeX commands

`\AtBeginDvi`
`\AtEndDvi`

`\AtBeginDvi {`⟨*code*⟩`}`

`\AtBeginDvi` is the existing LaTeX 2_ε interface to fill the `shipout/firstpage` hook. This is not really a good name as it is not just supporting `.dvi` but also `.pdf` output or `.xdv`.

`\AtEndDvi` is the counterpart that was not available in the kernel but only through the package atenddvi. It fills the `shipout/lastpage` hook.

Neither interface can set a code label but uses the current default label.

As these two wrappers have been available for a long time we continue offering them (but not enhancing them, e.g., by providing support for code labels).

For new code we strongly suggest using the high-level hook management commands directly instead of "randomly-named" wrappers. This will lead to code that is easier to understand and to maintain and it also allows you to set code labels if needed.

For this reason we do not provide any other "new" wrapper commands for the above hooks in the kernel, but only keep the existing ones for backward compatibility.

1.4 Special commands for use inside the hooks

\DiscardShipoutBox
\shipout_discard:

```
\AddToHookNext {shipout/before} {...\DiscardShipoutBox...}
```

The \DiscardShipoutBox declaration (L3 name \shipout_discard:) requests that on the next shipout the page box is thrown away instead of being shipped to the .dvi or .pdf file.

Typical applications wouldn't do this unconditionally, but have some processing logic that decides to use or not to use the page.

Note that if this declaration is used directly in the document it may depend on the placement to which page it applies, given that LaTeX output routine is called in an asynchronous manner! Thus normally one would use this only as part of the shipout/before code.

> *Todo: Once we have a new mark mechanism available we can improve on that and make sure that the declaration applies to the page that contains it — not done (yet)*

\DiscardShipoutBox cannot be used in any of the shipout/... hooks other than shipout/before.

In the atbegshi package there are a number of additional commands for use inside the shipout/before hook. They should normally not be needed any more as one can instead simply add code to the hooks shipout/before, shipout, shipout/background or shipout/foreground.[56] If atbegshi gets loaded then those commands become available as public functions with their original names as given below.

1.5 Provided LuaTeX callbacks

pre_shipout_filter

Under LuaTeX the pre_shipout_filter Lua callback is provided which gets called directly after the shipout hook, immediately before the shipout primitive gets invoked. The signature is

```
function(<node> head)
  return true
end
```

The head is the list node corresponding to the box to be shipped out. The return value should always be true.

[56] If that assumption turns out to be wrong it would be trivial to change them to public functions (right now they are private).

1.6 Information counters

`\ReadonlyShipoutCounter` `\ifnum\ReadonlyShipoutCounter=...`
`\g_shipout_readonly_int` `\int_use:N \g_shipout_readonly_int % expl3 usage`

This integer holds the number of pages shipped out up to now (including the one to be shipped out when inside the output routine). More precisely, it is incremented only after it is clear that a page will be shipped out, i.e., after the `shipout/before` hook (because that might discard the page)! In contrast `shipout/after` sees the incremented value.

Just like with the `page` counter its value is only accurate within the output routine. In the body of the document it may be off by one as the output routine is called asynchronously!

Also important: it *must not* be set, only read. There are no provisions to prevent that restriction, but if you manipulate it, chaos will be the result. To emphasize this fact it is not provided as a LaTeX counter but as a TeX counter (i.e., a command), so `\Alph{\ReadonlyShipoutCounter}` etc, would not work.

`totalpages` `\arabic{totalpages}`
`\g_shipout_totalpages_int` `\int_use:N \g_shipout_totalpage_int % expl3 usage`

In contrast to `\ReadonlyShipoutCounter`, the `totalpages` counter is a LaTeX counter and incremented for each shipout attempt including those pages that are discarded for one or the other reason. Again `shipout/before` sees the counter before it is incremented. In contrast `shipout/after` sees the incremented value.

Furthermore, while it is incremented for each page, its value is never used by LaTeX. It can therefore be freely reset or changed by user code, for example, to additionally count a number of pages that are not build by LaTeX but are added in a later part of the process, e.g., cover pages or picture pages made externally.

Important: as this is a page-related counter its value is only reliable inside the output routine!

`\PreviousTotalPages` `\PreviousTotalPages`

Command that expands to the number of total pages from the previous run. If there was no previous run or if used in the preamble it expands to `0`. Note that this is a command and not a counter, so in order to display the number in, say, Roman numerals you have to assign its value to a counter and then use `\Roman` on that counter.

1.7 Debugging shipout code

`\DebugShipoutsOn` `\DebugShipoutsOn`
`\DebugShipoutsOff`
`\shipout_debug_on:`
`\shipout_debug_off:`

Turn the debugging of shipout code on or off. This displays changes made to the shipout data structures.

Todo: This needs some rationalizing and may not stay this way.

2 Emulating commands from other packages

The packages in this section are no longer necessary, but as they are used by other packages, they are emulated when they are explicitly loaded with `\usepackage` or `\RequirePackage`.

Please note that the emulation only happens if the package is explicitly requested, i.e., the commands documented below are not automatically available in the LaTeX kernel! If you write a new package we suggest to use the appropriate kernel hooks directly instead of loading the emulation.

2.1 Emulating atbegshi

`\AtBeginShipoutUpperLeft`
`\AtBeginShipoutUpperLeftForeground`

`\AddToHook {shipout/before} {...\AtBeginShipoutUpperLeft{`⟨*code*⟩`}...}`

This adds a `picture` environment into the background of the shipout box expecting ⟨*code*⟩ to contain `picture` commands. The same effect can be obtained by simply using kernel features as follows:

`\AddToHook{shipout/background}{`⟨*code*⟩`}`

There is one technical difference: if `\AtBeginShipoutUpperLeft` is used several times each invocation is put into its own box inside the shipout box whereas all ⟨*code*⟩ going into `shipout/background` ends up all in the same box in the order it is added or sorted based on the rules for the hook chunks.

`\AtBeginShipoutUpperLeftForeground` is similar with the difference that the `picture` environment is placed in the foreground. To model it with the kernel functions use the hook `shipout/foreground` instead.

`\AtBeginShipoutAddToBox`
`\AtBeginShipoutAddToBoxForeground`

`\AddToHook {shipout/before} {...\AtBeginShipoutAddToBox{`⟨*code*⟩`}...}`

These work like `\AtBeginShipoutUpperLeft` and `\AtBeginShipoutUpperLeftForeground` with the difference that ⟨*code*⟩ is directly placed into an `\hbox` inside the shipout box and not surrounded by a `picture` environment.

To emulate them using `shipout/background` or `shipout/foreground` you may have to wrap ⟨*code*⟩ into a `\put` statement but if the code is not doing any typesetting just adding it to the hook should be sufficient.

`\AtBeginShipoutBox` This is the name of the shipout box as atbegshi knows it.

`\AtBeginShipoutOriginalShipout`

This is the name of the `\shipout` primitive as atbegshi knows it. This bypasses all the mechanisms set up by the LaTeX kernel and there are various scenarios in which it can therefore fail. It should only be used to run existing legacy atbegshi code but not in newly developed applications.

The kernel alternative is `\RawShipout` which is integrated with the LaTeX mechanisms and updates, for example, the `\ReadonlyShipoutCounter` counter. Please use `\RawShipout` for new code if you want to bypass the before, foreground and background hooks.

\AtBeginShipoutInit By default atbegshi delayed its action until \begin{document}. This command was forcing it in an earlier place. With the new concept it does nothing.

\AtBeginShipout
\AtBeginShipoutNext

\AtBeginShipout{⟨*code*⟩} ≡ \AddToHook{shipout/before}{⟨*code*⟩}
\AtBeginShipoutNext{⟨*code*⟩} ≡ \AddToHookNext{shipout/before}{⟨*code*⟩}

This is equivalent to filling the shipout/before hook by either using \AddToHook or \AddToHookNext, respectively.

\AtBeginShipoutFirst
\AtBeginShipoutDiscard

The atbegshi names for \AtBeginDvi and \DiscardShipoutBox.

2.2 Emulating everyshi

The everyshi package is providing commands to run arbitrary code just before the shipout starts. One point of difference: in the new shipout hooks the page is available as \ShipoutBox for inspection of change, one should not manipulate box 255 directly inside shipout/before, so old code doing this would change to use \ShipoutBox instead of 255 or \@cclv.

\EveryShipout \EveryShipout{⟨*code*⟩} ≡ \AddToHook{shipout/before}{⟨*code*⟩}

\AtNextShipout \AtNextShipout{⟨*code*⟩} ≡ \AddToHookNext{shipout/before}{⟨*code*⟩}

However, most use cases for everyshi are attempts to put some picture or text into the background or foreground of the page and that can be done today simply by using the shipout/background and shipout/foreground hooks without any need to coding.

2.3 Emulating atenddvi

The atenddvi package implemented only a single command: \AtEndDvi and that is now available out of the box so the emulation makes the package a no-op.

2.4 Emulating everypage

This package patched the original \@begindvi hook and replaced it with its own version. Its functionality is now covered by the hooks offered by the kernel so that there is no need for such patching any longer.

\AddEverypageHook \AddEverypageHook{⟨*code*⟩} ≡
\AddToHook{shipout/background}{\put(1in,-1in){⟨*code*⟩}}

\AddEverypageHook is adding something into the background of every page at a position of 1in to the right and 1in down from the top left corner of the page. By using the kernel hook directly you can put your material directly to the right place, i.e., use other coordinates in the \put statement above.

\AddThispageHook \AddThispageHook{⟨*code*⟩} ≡
\AddToHookNext{shipout/background}{\put(1in,-1in){⟨*code*⟩}}

The \AddThispageHook wrapper is similar but uses \AddToHookNext.

3 The Implementation

```
⟨@@=shipout⟩
```

At the moment the whole module rolls back in one go, but if we make any modifications in later releases this will then need splitting.

```
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\shipout}{Hook management (shipout)}%
\ExplSyntaxOn
```

3.1 Debugging

\g__shipout_debug_bool

Holds the current debugging state.

```
\bool_new:N \g__shipout_debug_bool
```

(*End of definition for* `\g__shipout_debug_bool`.)

\shipout_debug_on:
\shipout_debug_off:
__shipout_debug:n
__shipout_debug_gset:

Turns debugging on and off by redefining `\__shipout_debug:n`.

```
\cs_new_eq:NN \__shipout_debug:n  \use_none:n
\cs_new_protected:Npn \shipout_debug_on:
  {
    \bool_gset_true:N \g__shipout_debug_bool
    \__shipout_debug_gset:
  }
\cs_new_protected:Npn \shipout_debug_off:
  {
    \bool_gset_false:N \g__shipout_debug_bool
    \__shipout_debug_gset:
  }
\cs_new_protected:Npn \__shipout_debug_gset:
  {
    \cs_gset_protected:Npx \__shipout_debug:n ##1
      { \bool_if:NT \g__shipout_debug_bool {##1} }
  }
```

(*End of definition for* `\shipout_debug_on:` *and others. These functions are documented on page* 1064.)

\ShipoutBox
\l_shipout_box

The box filled with the page to be shipped out (both L3 and LaTeX 2_ε name).

```
\box_new:N  \l_shipout_box
\cs_set_eq:NN \ShipoutBox \l_shipout_box
```

(*End of definition for* `\ShipoutBox` *and* `\l_shipout_box`. *These functions are documented on page* 1060.)

\l__shipout_raw_box

The `\RawShipout` gets its own box but it is internal as there is no hook manipulation for it.

```
\box_new:N  \l__shipout_raw_box
```

(*End of definition for* `\l__shipout_raw_box`.)

__shipout_finalize_box: For LuaTEX invoke the pre_shipout_filter callback.

```
\sys_if_engine_luatex:TF
  {
    \newprotectedluacmd \__shipout_finalize_box:
    \exp_args:Nx \everyjob {
      \exp_not:V \everyjob
      \exp_not:N \lua_now:n {
        luatexbase.create_callback('pre_shipout_filter', 'list')
        local~call, getbox, setbox = luatexbase.call_callback, tex.getbox, tex.setbox~
        lua.get_functions_table()[\the \allocationnumber] = function()
          local~head = getbox(\the \l_shipout_box)
          local~result = call('pre_shipout_filter', head)
          if~not (result == head) then~
            setbox(\the \l_shipout_box, result~or~nil)
          end~
        end
      }
    }
  } {
    \cs_set_eq:NN \__shipout_finalize_box: \scan_stop:
  }
```

(*End of definition for* __shipout_finalize_box:.)

__shipout_execute: This is going to the be the code run by \shipout. The code follows closely the ideas from atbegshi, so not documenting that here for now.

```
\cs_set_protected:Npn \__shipout_execute: {
  \tl_set:Nx \l__shipout_group_level_tl
     { \int_value:w \tex_currentgrouplevel:D }
  \tex_afterassignment:D \__shipout_execute_test_level:
  \tex_setbox:D \l_shipout_box
}
```

(*End of definition for* __shipout_execute:.)

\shipout Overloading the \shipout primitive:

```
\cs_gset_eq:NN \shipout \__shipout_execute:
```

(*End of definition for* \shipout. *This function is documented on page* 1059.)

\l__shipout_group_level_tl Helper token list to record the group level at which __shipout_execute: is encountered.

```
\tl_new:N \l__shipout_group_level_tl
```

(*End of definition for* \l__shipout_group_level_tl.)

__shipout_execute_test_level: If the group level has changed then we are still constructing \l_shipout_box and to continue we need to wait until the current group has finished, hence the \tex_aftergroup:D.

```
\cs_new:Npn \__shipout_execute_test_level: {
  \int_compare:nNnT
     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
     \tex_aftergroup:D \__shipout_execute_cont:
}
```

(*End of definition for* __shipout_execute_test_level:.)

__shipout_execute_cont:

This does the actual shipout running several hooks as part of it. The code for them is passed as argument #2 to #4 to __shipout_execute_main_cont:Nnnn; the first argument is the box to be shipped out.

```
\cs_new:Npn \__shipout_execute_cont: {
  \__shipout_execute_main_cont:Nnnn
     \l_shipout_box
     { \hook_use:n {shipout/before} }
     { \hook_if_empty:nF {shipout/foreground}
         { \__shipout_add_foreground_picture:n
             { \hook_use:n {shipout/foreground} } }
```

If the user hook for the background (shipout/background) has no code, there might still code in the kernel hook so we need to test for this too. We only test for the \@kernel@before@shipout@background though. If the \@kernel@after@shipout@background needs executing even if the user hook is empty then we can add another test (or the kernel could put something into the before hook).

```
        \bool_lazy_and:nnF
          { \hook_if_empty_p:n {shipout/background} }
          { \tl_if_empty_p:N \@kernel@before@shipout@background }
          { \__shipout_add_background_picture:n
            { \@kernel@before@shipout@background
              \hook_use:n {shipout/background}
              \@kernel@after@shipout@background }
          }
     }
     { \hook_use:n {shipout/after} }
}
```

(*End of definition for* __shipout_execute_cont:.)

__shipout_execute_main_cont:Nnnn

When we have reached this point the shipout box has been processed and is available in \l_shipout_box and ready for real ship out (unless it gets discarded during the process).

The three arguments hold hook code that is executed just before the actual shipout (#1), within the shipout adding background and foreground material (#2) and after the shipout has happened (#3). These are passed as arguments because the same code without those hooks is also used when doing a "raw" shipout implemented by \RawShipout. The only hook that is always executed is that for the very last page, i.e., shipout/lastpage.

First we quickly check if it is void (can't happen in the standard LaTeX output routine but \shipout might be called from a package that has some special processing logic). If it is void we aren't shipping anything out and processing ends.[57]

```
\cs_new:Npn \__shipout_execute_main_cont:Nnnn #1#2#3#4 {
  \box_if_empty:NTF #1
    { \@latex@warning@no@line{ Ignoring~ void~ shipout~ box } }
    {
```

Otherwise we assume that we will ship something and prepare for final adjustments (in particular setting the state of \protect while we are running the hook code). We also save the current \protect state to restore it later.

```
%      \bool_gset_false:N \g__shipout_discard_bool  % setting this would disable
                                                    % \DiscardShipoutBox on doc-level
```

[57] In that case we don't reset the deadcycles, that would be up to the OR processing logic to do.

```
      \cs_set_eq:NN \__shipout_saved_protect: \protect
      \set@typeset@protect
```

We also store the current shipout box dimension in registers, so that they can be used in the hook code.[58]

```
      \__shipout_get_box_size:N #1
```

Then we execute the `shipout/before` hook (or nothing in case of `\RawShipout`).

```
      #2
```

In `\g_shipout_totalpages_int` we count all shipout attempts so we increment that counter already here (the other one is incremented later when we know for sure that we do a `\shipout`.

We increment it after running the above hook so that the values for `\g_shipout_totalpages_int` and `\g_shipout_readonly_int` are in sync while the hook is executed (in the case that totalpages isn't manually altered or through discarding pages that is).

```
      \int_gincr:N \g_shipout_totalpages_int
```

The above hook might contain code that requests the page to be discarded so we now test for it.

```
      \bool_if:NTF \g__shipout_discard_bool
        { \@latex@info@no@line{Completed~ page~ discarded}
          \bool_gset_false:N \g__shipout_discard_bool
```

As we are discarding the page box and not shipping anything out, we need to do some house cleaning and reset TEX's deadcycles so that it doesn't complain about too many calls to the OR without any shipout.

```
          \tex_deadcycles:D \c_zero_int
```

Todo: In atbegshi *the box was dropped but is that actually needed? Or the resetting of* `\protect` *to its kernel value?*

```
%           \group_begin:
%             \box_set_eq_drop:NN #1 #1
%           \group_end:
%           \cs_set_eq:NN \protect \exp_not:N
        }
```

Even if there was no explicit request to discard the box it is possible that the code for the hook `shipout/before` has voided the box (by mistake or deliberately). We therefore test once more but this time make it a warning, because the best practice way is to use the request mechanism.

```
        { \box_if_empty:NTF #1
            { \@latex@warning@no@line { Ignoring~ void~ shipout~ box.
                 \MessageBreak The~ shipout~ box~ was~ voided~ by~ hook~ code }
            }
```

Finally, if the box is still non-empty we are nearly ready to ship it out. First we increment the total page counter so that we can later test if we have reached the final page according to our available information.[59]

```
            {
```

[58] This is not really necessary as the code could access them via `\box_ht:N`, etc., but it is perhaps convenient.

[59] Doing that earlier would be wrong because we might end up with the last page counted but discard and then we have no place to add the final objects into the output file.

```
\int_gincr:N \g_shipout_readonly_int
\__shipout_debug:n {
  \typeout{Absolute~ page~ =~ \int_use:N \g_shipout_readonly_int
           \space (target:~ \@abspage@last)}
}
```

Then we store the box sizes again (as they may have changed) and then look at the hooks `shipout/foreground` and `shipout/background`. If either or both are non-empty we add a `picture` environment to the box (in the foreground and/or in the background) and execute the hook code inside that environment.

```
\__shipout_get_box_size:N #1
```

The first hook we run is the `shipout/firstpage` hook. This is only done once, then the `\__shipout_run_firstpage_hook:` command redefines itself to do nothing. If the hook contains `\special`s for integration at the top of the page they will be temporarily stored in a safe place and added later with `\__shipout_add_firstpage_specials:`.

```
\__shipout_run_firstpage_hook:
```

Run the hooks for background and foreground or, if this is called by `\RawShipout`, copy the box `\l__shipout_raw_box` to `\l_shipout_box` so that firstpage and lastpage material gets added if necessary (that is always done to `\l_shipout_box`.

```
#3
```

We then run `\__shipout_add_firstpage_specials:` that adds the content of the hook `shipout/firstpage` to the start of the first page (if non-empty). It is then redefined to do nothing on later pages.

```
\__shipout_add_firstpage_specials:
```

Then we check if we have to add the `shipout/lastpage` hook or the corresponding kernel hook because we have reached the last page. This test will be false for all but one (and hopefully the correct) page.

```
\int_compare:nNnT \@abspage@last = \g_shipout_readonly_int
  { \bool_lazy_and:nnF
      { \hook_if_empty_p:n {shipout/lastpage} }
      { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
      { \__shipout_debug:n { \typeout{Executing~ lastpage~ hook~
            on~ page~ \int_use:N \g_shipout_readonly_int } }
        \__shipout_add_foreground_box:n
            { \UseHook{shipout/lastpage}
              \@kernel@after@shipout@lastpage }
```

We record that we have handled the `shipout/lastpage` hook but only if we really did.

```
        \bool_gset_true:N \g__shipout_lastpage_handled_bool
      }
  }

\hook_use:n {shipout}
\__shipout_finalize_box:
```

Finally we run the actual TEX primitive for shipout. As that will expand delayed `\write` statements inside the page in which protected commands should not expand we first change `\protect` to the appropriate definition for that case.

```
\cs_set_eq:NN \protect \exp_not:N
\tex_shipout:D \box_use:N \l_shipout_box
```

The \l_shipout_box may contain the firstpage material if this was the very first shipout. That makes it unsuitable for reuse in another shipout, so as a safety measure the next command resets \l_shipout_box to its earlier state if that is necessary. On later pages this is then a no-op.

```
                \__shipout_drop_firstpage_specials:
```

The shipout/after hook (if in #4) needs to run with \protected commands again being executed, because that hook will "typeset" material added at the top of the next page.

```
                \set@typeset@protect
                #4
              }
          }
```

Restore the value of \protect in case \shipout is called outside of the output routine (where it is automatically restored because of the implicit group).

```
        \cs_set_eq:NN \protect \__shipout_saved_protect:
      }
}
```

(*End of definition for* __shipout_execute_main_cont:Nnnn.)

__shipout_execute_raw:
__shipout_execute_test_level_raw:

This implements the "raw" shipout which bypasses the before, foreground, background and after hooks. It follows the same pattern than __shipout_execute_raw: except that it finally calls __shipout_execute_main_cont:Nnnn with three empty arguments. instead of the hook code.

```
\cs_set_protected:Npn \__shipout_execute_raw: {
  \tl_set:Nx \l__shipout_group_level_tl
     { \int_value:w \tex_currentgrouplevel:D }
  \tex_afterassignment:D \__shipout_execute_test_level_raw:
  \tex_setbox:D \l__shipout_raw_box
}
```

```
\cs_new:Npn \__shipout_execute_test_level_raw: {
  \int_compare:nNnT
     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
     \tex_aftergroup:D \__shipout_execute_nohooks_cont:
}
```

Well, not totally empty arguments, we add some debugging if we are actually doing a shipout.

```
\cs_new:Npn \__shipout_execute_nohooks_cont: {
  \__shipout_execute_main_cont:Nnnn \l__shipout_raw_box
     {} { \__shipout_debug:n{ \typeout{Doing~ raw~ shipout~ ...} }
          \box_set_eq:NN \l_shipout_box \l__shipout_raw_box } {}
}
```

(*End of definition for* __shipout_execute_raw: *and* __shipout_execute_test_level_raw:.)

\RawShipout

The interface name for raw shipout.

```
\cs_gset_eq:NN \RawShipout \__shipout_execute_raw:
```

(*End of definition for* \RawShipout. *This function is documented on page* 1059.)

__shipout_saved_protect:

Remember the current \protect state.

```
\cs_new_eq:NN  \__shipout_saved_protect: \protect
```

(*End of definition for* `\__shipout_saved_protect:`.)

shipout/before
shipout
shipout/after
shipout/foreground
shipout/background
shipout/firstpage
shipout/lastpage

Declaring all hooks for the shipout code.

```
\hook_new:n{shipout/before}
\hook_new:n{shipout}
\hook_new:n{shipout/after}
\hook_new:n{shipout/foreground}
\hook_new:n{shipout/background}
\hook_new:n{shipout/firstpage}
\hook_new:n{shipout/lastpage}
```

(*End of definition for* `shipout/before` *and others. These functions are documented on page* 1060.)

\@kernel@after@shipout@lastpage
\@kernel@before@shipout@background
\@kernel@after@shipout@background

And here are the internal kernel hooks going before or after the public ones where needed.

```
\let\@kernel@after@shipout@lastpage\@empty
\let\@kernel@before@shipout@background\@empty
\let\@kernel@after@shipout@background\@empty
```

(*End of definition for* `\@kernel@after@shipout@lastpage`, `\@kernel@before@shipout@background`, *and* `\@kernel@after@shipout@background`.)

__shipout_run_firstpage_hook:

There are three commands to handle the `shipout/firstpage` hook: `\__shipout_run_firstpage_hook:`, `\__shipout_add_firstpage_specials:` and `\__shipout_drop_firstpage_specials:`.

That hook is supposed to contain `\special`s and similar material to be placed at the very beginning of the output page and so it needs careful placing to avoid that anything else gets in front of it. And this means we have to wait with this until other hooks such as `shipout/background` have added their bits. It is also important that such `\special`s show up only on the very first page, so if this page gets saved before `\shipout` for later reuse, we have to make sure that they aren't in the saved version.

In addition the hook may also contain code to be executed "first", e.g., visible from code in `shipout/background` and this conflicts with adding the `\special`s late.

Therefore the processing is split into different parts: `\__shipout_run_firstpage_hook:` is done early and checks if there is any material in the hook.

```
\cs_new:Npn \__shipout_run_firstpage_hook: {
  \hook_if_empty:nTF {shipout/firstpage}
```

If not then we define the other two commands to do nothing.

```
        {
          \cs_gset_eq:NN \__shipout_add_firstpage_specials:  \prg_do_nothing:
          \cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:
        }
```

If there is material we execute inside a box, which means any `\special` will end up in that box and any other code is executed and can have side effects (as long as they are global).

```
        {
          \hbox_set:Nn \l__shipout_firstpage_box { \UseHook{shipout/firstpage} }
        }
```

Once we are here we change the definition to do nothing next time and we also change the command used to implement \AtBeginDvi to become a warning and not add further material to a hook that is never used again.

```
  \cs_gset_eq:NN \__shipout_run_firstpage_hook: \prg_do_nothing:
  \cs_gset:Npn \__shipout_add_firstpage_material:Nn ##1 ##2 {
    \@latex@warning{ First~ page~ is~ already~ shipped~ out,~ ignoring
                     \MessageBreak \string##1 }
  }
}
```

(*End of definition for* __shipout_run_firstpage_hook:.)

__shipout_add_firstpage_specials:
__shipout_drop_firstpage_specials:

The __shipout_add_firstpage_specials: then adds the \specials stored in \l__shipout_firstpage_box to the page to be shipped out when the time is ready. Note that if there was no material in the shipout/firstpage hook then this command gets redefined to do nothing. But for most documents there is something, e.g., some PostScript header, or some meta data declaration, etc. so by default we assume there is something to do.

```
\cs_new:Npn \__shipout_add_firstpage_specials: {
```

First we make a copy of the \l_shipout_box that we can restore it later on.

```
  \box_set_eq:NN \l__shipout_raw_box \l_shipout_box
```

Adding something to the beginning means adding it to the background as that layer is done first in the output.

```
  \__shipout_add_background_box:n { \hbox_unpack_drop:N \l__shipout_firstpage_box }
```

After the actual shipout __shipout_drop_firstpage_specials: is run to restore the earlier content of \l_shipout_box and then redefines itself again to do nothing.

As a final act we change the definition to do nothing next time.

```
  \cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:
}
```

The __shipout_drop_firstpage_specials: is run after the shipout has occurred but before the shipout/afterpage hook is executed. That is the point where we have to restore the \ShipoutBox to its state without the shipout/firstpage material.

```
\cs_new:Npn \__shipout_drop_firstpage_specials: {
    \box_set_eq:NN \l_shipout_box \l__shipout_raw_box
```

If there was no such material then __shipout_run_firstpage_hook: will have changed the definition to a no-op already. Otherwise this is what we do here.

```
    \cs_gset_eq:NN \__shipout_drop_firstpage_specials:  \prg_do_nothing:
  }
```

(*End of definition for* __shipout_add_firstpage_specials: *and* __shipout_drop_firstpage_specials:.)

\l__shipout_firstpage_box

The box to hold any firstpage \specials.

```
\box_new:N \l__shipout_firstpage_box
```

(*End of definition for* \l__shipout_firstpage_box.)

\g__shipout_lastpage_handled_bool

A boolean to signal if we have already handled the shipout/lastpage hook.

```
\bool_new:N \g__shipout_lastpage_handled_bool
```

(*End of definition for* \g__shipout_lastpage_handled_bool.)

__shipout_add_firstpage_material:Nn

This command adds material to the `shipout/firstpage` hook. It is used in `\AtBeginDvi`, etc. The first argument is the command through which is it called. Initially this is ignored but once we are passed the first page it can be used to generate a warning message mentioning the right user command.

```
\cs_new:Npn \__shipout_add_firstpage_material:Nn #1#2 {
   \AddToHook{shipout/firstpage}{#2}
}
```

(*End of definition for* `\__shipout_add_firstpage_material:Nn`.)

__shipout_get_box_size:N

Store the box dimensions in dimen registers.

> *Todo: This could/should perhaps be generalized to set height depth and width given an arbitrary box.*

```
\cs_new:Npn \__shipout_get_box_size:N #1 {
  \dim_set:Nn \l_shipout_box_ht_dim { \box_ht:N #1 }
  \dim_set:Nn \l_shipout_box_dp_dim { \box_dp:N #1 }
  \dim_set:Nn \l_shipout_box_wd_dim { \box_wd:N #1 }
  \dim_set:Nn \l_shipout_box_ht_plus_dp_dim
      { \l_shipout_box_ht_dim + \l_shipout_box_dp_dim }
}
```

(*End of definition for* `\__shipout_get_box_size:N`.)

\l_shipout_box_ht_dim
\l_shipout_box_dp_dim
\l_shipout_box_wd_dim
\l_shipout_box_ht_plus_dp_dim

And here are the variables set by `\__shipout_get_box_size:N`.

```
\dim_new:N \l_shipout_box_ht_dim
\dim_new:N \l_shipout_box_dp_dim
\dim_new:N \l_shipout_box_wd_dim
\dim_new:N \l_shipout_box_ht_plus_dp_dim
```

(*End of definition for* `\l_shipout_box_ht_dim` *and others. These functions are documented on page* *1060*.)

\g__shipout_discard_bool

Indicate whether or not the current page box should be discarded

```
\bool_new:N \g__shipout_discard_bool
```

(*End of definition for* `\g__shipout_discard_bool`.)

\l__shipout_tmp_box
\l__shipout_saved_badness_tl

We need a box for the background and foreground material and a token register to remember badness settings as we disable them during the buildup below.

```
\box_new:N \l__shipout_tmp_box
\tl_new:N  \l__shipout_saved_badness_tl
```

(*End of definition for* `\l__shipout_tmp_box` *and* `\l__shipout_saved_badness_tl`.)

__shipout_add_background_box:n

In standard LaTeX the shipout box is always a `\vbox` but here we are allow for other usage as well, in case some package has its own output routine.

```
\cs_new:Npn \__shipout_add_background_box:n #1
{ \__shipout_get_box_size:N \l_shipout_box
```

But we start testing for a vertical box as that should be the normal case.

```
  \box_if_vertical:NTF \l_shipout_box
      {
```

Save current values of \vfuzz and \vbadness then change them to allow box manipulations without warnings.

```
\tl_set:Nx \l__shipout_saved_badness_tl
   { \vfuzz=\the\vfuzz\relax
     \vbadness=\the\vbadness\relax }
\vfuzz=\c_max_dim
\vbadness=\c_max_int
```

Then we reconstruct \l_shipout_box ...

```
\vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
     {
```

... the material in #1 is placed into a horizontal box with zero dimensions.

```
          \hbox_set:Nn \l__shipout_tmp_box
               { \l__shipout_saved_badness_tl #1 }
          \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
          \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
          \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
```

The we typeset that box followed by whatever was in \l_shipout_box before (unpacked).

```
          \skip_zero:N \baselineskip
          \skip_zero:N \lineskip
          \skip_zero:N \lineskiplimit
          \box_use:N \l__shipout_tmp_box
          \vbox_unpack:N \l_shipout_box
```

The \kern ensures that the box has no depth which is afterwards explicitly corrected.

```
          \kern \c_zero_dim
     }
\box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
\box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
```

Todo: The whole boxing maneuver looks a bit like overkill to me, but for the moment I leave.

```
\l__shipout_saved_badness_tl
}
{
```

A horizontal box is handled in a similar way. The last case would be a void box in which case we do nothing hence the missing F branch.

```
\box_if_horizontal:NT \l_shipout_box
    {
      \tl_set:Nx \l__shipout_saved_badness_tl
         { \hfuzz=\the\hfuzz\relax
           \hbadness=\the\hbadness\relax }
      \hfuzz=\c_max_dim
      \hbadness=\c_max_int
      \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
           {
             \hbox_set:Nn \l__shipout_tmp_box
                  { \l__shipout_saved_badness_tl #1 }
             \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
             \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
             \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
```

```
                    \box_move_up:nn
                        \l_shipout_box_ht_dim
                        { \box_use:N \l__shipout_tmp_box }
                    \hbox_unpack:N \l_shipout_box
                  }
              \l__shipout_saved_badness_tl
            }
      }
}
```

(*End of definition for* `\__shipout_add_background_box:n`.)

`\__shipout_add_foreground_box:n` Foreground boxes are done in the same way, only the order and placement of boxes has to be done differently.

```
\cs_new:Npn \__shipout_add_foreground_box:n #1
{
  \box_if_vertical:NTF \l_shipout_box
    {
      \tl_set:Nx \l__shipout_saved_badness_tl
         { \vfuzz=\the\vfuzz\relax
           \vbadness=\the\vbadness\relax }
      \vfuzz=\c_max_dim
      \vbadness=\c_max_int
      \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
           {
             \hbox_set:Nn \l__shipout_tmp_box
                  { \l__shipout_saved_badness_tl #1 }
             \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
             \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
             \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
             \skip_zero:N \baselineskip
             \skip_zero:N \lineskip
             \skip_zero:N \lineskiplimit
             \vbox_unpack:N \l_shipout_box
             \kern -\l_shipout_box_ht_plus_dp_dim
             \box_use:N \l__shipout_tmp_box
             \kern  \l_shipout_box_ht_plus_dp_dim
           }
      \l__shipout_saved_badness_tl
      \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
      \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
    }
    {
      \box_if_horizontal:NT \l_shipout_box
        {
          \tl_set:Nx \l__shipout_saved_badness_tl
            { \hfuzz=\the\hfuzz\relax
              \hbadness=\the\hbadness\relax }
          \hfuzz=\c_max_dim
          \hbadness=\c_max_int
          \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
               {
                 \hbox_unpack:N \l_shipout_box
                 \kern -\box_wd:N \l_shipout_box
```

```
                  \hbox_set:Nn \l__shipout_tmp_box
                      { \l__shipout_saved_badness_tl #1 }
                  \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
                  \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
                  \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
                  \box_move_up:nn { \box_ht:N \l_shipout_box }
                                { \box_use:N \l__shipout_tmp_box }
                  \kern \box_wd:N \l_shipout_box
                }%
                \l__shipout_saved_badness_tl
        }
    }
}
```

(*End of definition for* `\__shipout_add_foreground_box:n`.)

`\__shipout_init_page_origins:`
`\c__shipout_horigin_tl`
`\c__shipout_vorigin_tl`

Two constants holding the offset of the top-left with respect to the media box.

Setting the constants this way is courtesy of Bruno.

We delay setting the constants to the last possible place as there might be updates in the preamble or even in the `begindocument` hook that affects their setup.

```
\cs_new:Npn \__shipout_init_page_origins: {
  \tl_const:Nx \c__shipout_horigin_tl
     {
       \cs_if_exist_use:NTF \pdfvariable { horigin }
          { \cs_if_exist_use:NF \pdfhorigin { 1in } }
     }
  \tl_const:Nx \c__shipout_vorigin_tl
     {
       \cs_if_exist_use:NTF \pdfvariable { vorigin }
          { \cs_if_exist_use:NF \pdfvorigin { 1in } }
     }
```

After the constants have been set there is no need to execute this command again, in fact it would raise an error, so we redefine it to do nothing.

```
  \cs_gset_eq:NN \__shipout_init_page_origins: \prg_do_nothing:
}
```

(*End of definition for* `\__shipout_init_page_origins:`, `\c__shipout_horigin_tl`, *and* `\c__shipout_vorigin_tl`.)

`\__shipout_picture_overlay:n`

Put the argument into a `picture` environment that doesn't take up any size and uses `1pt` for `\unitlength`.

> *Todo: Could perhaps be generalized as it might be useful elsewhere. For now it is not.*

```
\cs_new:Npn \__shipout_picture_overlay:n #1 {
```

The very first time this is executed we have to initialize (and freeze) the origins.

```
    \__shipout_init_page_origins:
    \kern -\c__shipout_horigin_tl \scan_stop:
    \vbox_to_zero:n {
      \kern -\c__shipout_vorigin_tl \scan_stop:
      \unitlength 1pt \scan_stop:
```

This mimics a simple zero-sized picture environment. The \hss is need in case there is horizontal material (without using \put with a positive width.

```
      \hbox_set_to_wd:Nnn \l__shipout_tmp_box \c_zero_dim
                          { \ignorespaces #1 \hss }
      \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
      \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
      \box_use:N \l__shipout_tmp_box
      \tex_vss:D
    }
}
```

(*End of definition for* __shipout_picture_overlay:n.)

__shipout_add_background_picture:n

Put a picture env in the background of the shipout box with its reference point in the top-left corner.

```
\cs_new:Npn \__shipout_add_background_picture:n #1 {
   \__shipout_add_background_box:n { \__shipout_picture_overlay:n {#1} }
}
```

(*End of definition for* __shipout_add_background_picture:n.)

__shipout_add_foreground_picture:n

Put a picture env in the foreground of the shipout box with its reference point in the top-left corner.

```
\cs_new:Npn \__shipout_add_foreground_picture:n #1 {
   \__shipout_add_foreground_box:n { \__shipout_picture_overlay:n {#1} }
}
```

(*End of definition for* __shipout_add_foreground_picture:n.)

\shipout_discard:

Request that the next shipout box should be discarded. At the moment this is just setting a boolean, but we may want to augment this behavior that the position of the call is taken into account (in case LaTeX looks ahead and is not using the position for on the next page).

```
\cs_new_protected:Npn \shipout_discard: {
  \bool_gset_true:N \g__shipout_discard_bool
}
```

(*End of definition for* \shipout_discard:. *This function is documented on page* 1063.)

3.2 Handling the end of job hook

At the moment this is partly solved by using the existing hooks. But rather than putting the code into these hooks it should be moved to the right place directly as we shouldn't prefill hooks with material unless it needs to interact with other code.

\g_shipout_readonly_int
\ReadonlyShipoutCounter

We count every shipout activity that makes a page (but not those that are discarded) in order to know how many pages got produced.

```
\int_new:N \g_shipout_readonly_int
```

For LaTeX 2_ε it is available as a command (i.e., a TeX counter only.

```
\cs_new_eq:NN \ReadonlyShipoutCounter  \g_shipout_readonly_int
```

(*End of definition for* \g_shipout_readonly_int *and* \ReadonlyShipoutCounter. *These functions are documented on page* 1064.)

\g_shipout_totalpages_int
\c@totalpages

We count every shipout attempt (even those that are discarded) in this counter. It is not used in the code but may get used in user code.

```
\int_new:N \g_shipout_totalpages_int
```

For LaTeX 2ε this is offered as a LaTeX counter so can be easily typeset inside the output routine to display things like "\thepage/\thetotalpages", etc.

```
\cs_new_eq:NN \c@totalpages \g_shipout_totalpages_int
\cs_new:Npn \thetotalpages { \arabic{totalpages} }
```

(*End of definition for* \g_shipout_totalpages_int *and* \c@totalpages*. These functions are documented on page* 1064.)

\@abspage@last

In \@abspage@last record the number of pages from the last run. This is written to the .aux and this way made available to the next run. In case there is no .aux file or the statement is missing from it we initialize it with the largest possible number in TeX. We use this as the default because then we are inserting the shipout/lastpage on the last page (or after the last page) but not on page 1 for a multipage document.

```
\xdef\@abspage@last{\number\maxdimen}
```

(*End of definition for* \@abspage@last.)

\enddocument

Instead of using the hooks enddocument and enddocument/afterlastpage we add this code to private kernel hooks to be 100% sure when it is executed and to avoid cluttering the hooks with data that is always there.

Inside \enddocument there is a \clearpage. Just before that we execute this code here. There is a good chance that we are on the last page. Therefore, if we don't know the value from the last run, we assume that the current page is the right one. So we set \@abspage@last and as a result the next shipout will run the shipout/lastpage code. Of course, if there are floats that still need a placement this guess will be wrong but then rerunning the document will give us the correct value next time around.

\@kernel@after@enddocument

```
\g@addto@macro \@kernel@after@enddocument {
  \int_compare:nNnT \@abspage@last = \maxdimen
    {
```

We use LaTeX 2ε coding as \@abspage@last is not an L3 name.

```
      \xdef\@abspage@last{ \int_eval:n {\g_shipout_readonly_int + 1} }
    }
}
```

\@kernel@after@enddocument@afterlastpage

Once the \clearpage has done its work inside \enddocument we know for sure how many pages this document has, so we record that in the .aux file for the next run.

```
\g@addto@macro \@kernel@after@enddocument@afterlastpage {
```

There is one special case: If no output is produced then there is no point in a) recording the number as 0 will never match the page number of a real page and b) adding an extra page to ran the shipout/lastpage is pointless as well (as it would remain forever). So we test for this and run the code only if there have been pages.

```
  \int_compare:nNnF \g_shipout_readonly_int = 0
    {
```

This ends up in the .aux so we use LaTeX 2ε names here.

Todo: This needs an interface for `\nofiles` *in expl3, doesn't at the moment!*

```
    \if@filesw
       \iow_now:Nx \@auxout {
         \gdef\string\@abspage@last {\int_use:N \g_shipout_readonly_int}}
    \fi
```

But we may have guessed wrongly earlier and have run it too early or we still have to run the `shipout/lastpage` even though there is no page to place it into. If that is the case we make a trivial extra page and put it there. This temporary page will then vanish again on the next run but helps to keep pdf viewers happy. In either case we should put out an appropriate "rerun" warning.

```
     \bool_if:NTF \g__shipout_lastpage_handled_bool
        {
```

If the hook was already executed, we have to test if that total shipouts match the shipouts from last run (because that corresponds to the page it was executed). If not we output a warning.

```
          \int_compare:nNnF \@abspage@last = \g_shipout_readonly_int
           {
             \@latex@warning@no@line{Hook~ 'shipout/lastpage'~ executed~
               on~ wrong~ page~ (\@abspage@last\space not~
               \int_use:N\g_shipout_readonly_int).\MessageBreak
               Rerun~ to~ correct~ this}%
           }
        }
        {
```

If the hook was not run, we need to add an extra page and place it there. However, making this extra page in case the hook is actually empty would be forcing a rerun without any reason, so we check that condition and also check if `\@kernel@after@shipout@lastpage` contains any code. If both are empty we omit the page generation.

```
         \bool_lazy_and:nnF
           { \hook_if_empty_p:n {shipout/lastpage} }
           { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
           {
             \tex_shipout:D\vbox to\textheight
               {
                 \hbox:n { \UseHook{shipout/lastpage}
                           \@kernel@after@shipout@lastpage }
```

This extra page could be totally empty except for the hook content, but to help the user understanding why it is there we put some text into it.

```
                 \__shipout_excuse_extra_page:
                 \null
               }
```

At this point we also signal to LaTeX's endgame that a rerun is necessary so that an appropriate message can be shown on the terminal. We do this by simply defining a command used as a flag and tested in `\enddocument`.

```
             \cs_gset_eq:NN \@extra@page@added \relax
           }
        }
    }
}
```

(*End of definition for* `\enddocument`, `\@kernel@after@enddocument`, *and* `\@kernel@after@enddocument@afterlastpage`.)

`\__shipout_excuse_extra_page:` Say mea culpa ...

```
\cs_new:Npn \__shipout_excuse_extra_page: {
  \vfil
  \begin{center}
    \bfseries Temporary~ page!
  \end{center}
    \LaTeX{}~ was~ unable~ to~ guess~ the~ total~ number~ of~ pages~
    correctly.~ ~ As~ there~ was~ some~ unprocessed~ data~ that~
    should~ have~ been~ added~ to~ the~ final~ page~ this~ extra~
    page~ has~ been~ added~ to~ receive~ it.
    \par
    If~ you~ rerun~ the~ document~ (without~ altering~ it)~ this~
    surplus~ page~ will~ go~ away,~ because~ \LaTeX{}~ now~ knows~
    how~ many~ pages~ to~ expect~ for~ this~ document.
  \vfil
}
```

(*End of definition for* `\__shipout_excuse_extra_page:`.)

`\PreviousTotalPages` `\@kernel@before@begindocument` In the preamble before the `aux` file was read `\PreviousTotalPages` is always zero.

```
\def\PreviousTotalPages{0}
```

In the `aux` file there should be an update for `\@abspage@last` recording the number of pages from the previous run. If not that macro holds the value of `\maxdimen`. So we test for it and update `\PreviousTotalPages` if there was a real value. This should happen just before the `begindocument` hook is executed so that the value can be used inside that hook.

```
\g@addto@macro\@kernel@before@begindocument
  {\ifnum\@abspage@last<\maxdimen
     \xdef\PreviousTotalPages{\@abspage@last}\fi}
```

(*End of definition for* `\PreviousTotalPages` *and* `\@kernel@before@begindocument`. *These functions are documented on page* 1064.)

4 Legacy LaTeX 2ε interfaces

`\DiscardShipoutBox` Request that the next shipout box is to be discarded.

```
\cs_new_eq:NN \DiscardShipoutBox \shipout_discard:
```

(*End of definition for* `\DiscardShipoutBox`. *This function is documented on page* 1063.)

`\AtBeginDvi` If we roll forward from an earlier kernel `\AtBeginDvi` is defined so we better not use `\cs_new_protected:Npn` here.

```
\cs_set_protected:Npn \AtBeginDvi
                      {\__shipout_add_firstpage_material:Nn \AtBeginDvi}
```

(*End of definition for* `\AtBeginDvi`. *This function is documented on page* 1062.)

`\DebugShipoutsOn` `\DebugShipoutsOff`

```
\cs_new_eq:NN \DebugShipoutsOn  \shipout_debug_on:
\cs_new_eq:NN \DebugShipoutsOff \shipout_debug_off:
```

(*End of definition for* `\DebugShipoutsOn` *and* `\DebugShipoutsOff`. *These functions are documented on page* 1064.)

5 Internal commands needed elsewhere

These internal commands use double and triple `@` signs so we need to stop getting them translated to the module name.

```
⟨@@=⟩
```

\@expl@@@shipout@add@firstpage@material@@Nn
\@expl@@@shipout@add@background@box@@n
\@expl@@@shipout@add@foreground@box@@n
\@expl@@@shipout@add@background@picture@@n
\@expl@@@shipout@add@foreground@picture@@n

Some internals needed elsewhere.

```
\cs_set_eq:NN \@expl@@@shipout@add@firstpage@material@@Nn
              \__shipout_add_firstpage_material:Nn
\cs_set_eq:NN \@expl@@@shipout@add@background@box@@n
              \__shipout_add_background_box:n
\cs_set_eq:NN \@expl@@@shipout@add@foreground@box@@n
              \__shipout_add_foreground_box:n
\cs_set_eq:NN \@expl@@@shipout@add@background@picture@@n
              \__shipout_add_background_picture:n
\cs_set_eq:NN \@expl@@@shipout@add@foreground@picture@@n
              \__shipout_add_foreground_picture:n
```

(*End of definition for* `\@expl@@@shipout@add@firstpage@material@@Nn` *and others.*)

```
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

Rolling back here doesn't undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\shipout}{Hook management (shipout)}%
⟨latexrelease⟩
```

If we roll forward then `\tex_shipout:D` may not be defined in which case `\shipout` does have it original definition and so we must not `\let` it to something else which is `\relax`!

```
⟨latexrelease⟩\ifcsname tex_shipout:D\endcsname
⟨latexrelease⟩\expandafter\let\expandafter\shipout
⟨latexrelease⟩                \csname tex_shipout:D\endcsname
⟨latexrelease⟩\fi
⟨latexrelease⟩
⟨latexrelease⟩\let \RawShipout\@undefined
⟨latexrelease⟩\let \ShipoutBox\@undefined
⟨latexrelease⟩\let \ReadonlyShipoutCounter \@undefined
⟨latexrelease⟩\let \c@totalpages \@undefined
⟨latexrelease⟩\let \thetotalpages \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let \DiscardShipoutBox \@undefined
⟨latexrelease⟩\let \DebugShipoutsOn \@undefined
⟨latexrelease⟩\let \DebugShipoutsOff \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand \AtBeginDvi [1]{%
⟨latexrelease⟩  \global \setbox \@begindvibox
⟨latexrelease⟩    \vbox{\unvbox \@begindvibox #1}%
⟨latexrelease⟩}
```

```
⟨latexrelease⟩
⟨latexrelease⟩\let \AtBeginShipout \@undefined
⟨latexrelease⟩\let \AtBeginShipoutNext \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let \AtBeginShipoutFirst \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let \ShipoutBoxHeight \@undefined
⟨latexrelease⟩\let \ShipoutBoxDepth \@undefined
⟨latexrelease⟩\let \ShipoutBoxWidth \@undefined
⟨latexrelease⟩
```

We do not undo a substitution when rolling back. As the file support gets undone the underlying data is no longer used (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\undeclare@...` and its support macros available in all earlier kernel releases which is pointless (and actually worse).

```
⟨latexrelease⟩
⟨latexrelease⟩\let  \AtEndDvi \@undefined
```

We do not reenable a disabled package load when rolling back. As the file support gets undone the underlying data is no longer checked (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\reenable@package@load` command available in all earlier kernel releases which is pointless (and actually worse).

```
%\reenable@package@load{atenddvi}
```

```
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

6 Package emulation for compatibility

6.1 Package **atenddvi** emulation

`\AtEndDvi` This package has only one public command, so simulating it is easy and actually sensible to provide as part of the kernel.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\AtEndDvi}{atenddvi emulation}%
\ExplSyntaxOn
\cs_new_protected:Npn \AtEndDvi #1 {\AddToHook{shipout/lastpage}{#1}}
\ExplSyntaxOff
```

As the package is integrate we prevent loading (no need to roll that back):

```
\disable@package@load{atenddvi}
   {\PackageWarning{atenddvi}
     {Functionality of this package is already\MessageBreak
      provided by LaTeX.\MessageBreak\MessageBreak
      It is there no longer necessary to load it\MessageBreak
      and you can safely remove it.\MessageBreak
      Found on}}
```

```
⟨/2ekernel | latexrelease⟩
```

```
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\AtEndDvi}{atenddvi emulation}%
⟨latexrelease⟩\let \AtEndDvi \@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\AtEndDvi`*. This function is documented on page* 1062.)

```
⟨/2ekernel⟩
```

6.2 Package **atbegshi** emulation

```
⟨*atbegshi-ltx⟩
\ProvidesPackage{atbegshi-ltx}
   [2021/01/10 v1.0c
     Emulation of the original atbegshi^^Jpackage with kernel methods]
```

`\AtBeginShipoutBox`

```
\let \AtBeginShipoutBox \ShipoutBox
```

(*End of definition for* `\AtBeginShipoutBox`*. This function is documented on page* 1065.)

`\AtBeginShipoutInit` Compatibility only, we aren't delaying ...

```
\let \AtBeginShipoutInit \@empty
```

(*End of definition for* `\AtBeginShipoutInit`*. This function is documented on page* 1066.)

`\AtBeginShipout` `\AtBeginShipoutNext` Filling hooks

```
\protected\long\def\AtBeginShipout     #1{\AddToHook{shipout/before}{#1}}
\protected\long\def\AtBeginShipoutNext #1{\AddToHookNext{shipout/before}{#1}}
```

(*End of definition for* `\AtBeginShipout` *and* `\AtBeginShipoutNext`*. These functions are documented on page* 1066.)

`\AtBeginShipoutFirst` Slightly more complex as we need to know the name of the command under which the `shipout/firstpage` hook is filled.

```
\protected \def \AtBeginShipoutFirst
   {\@expl@@@shipout@add@firstpage@material@@Nn \AtBeginShipoutFirst}
```

(*End of definition for* `\AtBeginShipoutFirst`*. This function is documented on page* 1066.)

`\AtBeginShipoutDiscard` Just a different name.

```
\let \AtBeginShipoutDiscard \DiscardShipoutBox
```

(*End of definition for* `\AtBeginShipoutDiscard`*. This function is documented on page* 1066.)

`\AtBeginShipoutAddToBox` `\AtBeginShipoutAddToBoxForeground` `\AtBeginShipoutUpperLeft` `\AtBeginShipoutUpperLeftForeground` We don't expose them.

```
\let \AtBeginShipoutAddToBox
              \@expl@@@shipout@add@background@box@@n
\let \AtBeginShipoutAddToBoxForeground
              \@expl@@@shipout@add@foreground@box@@n
\let \AtBeginShipoutUpperLeft
              \@expl@@@shipout@add@background@picture@@n
\let \AtBeginShipoutUpperLeftForeground
              \@expl@@@shipout@add@foreground@picture@@n
```

(*End of definition for* `\AtBeginShipoutAddToBox` *and others. These functions are documented on page* 1065.)

`\AtBeginShipoutOriginalShipout`

This offers the raw `\shipout` primitive of the engine. A page shipped out with this is not counted by `\ReadonlyShipoutCounter` counter and thus the mechanism to place `\specials` at the very end of the output might fail, etc. It should therefore not be used in new applications but is only provided to allow running legacy code. For new code use the commands provided by the kernel instead.

```
\ExplSyntaxOn
\cs_new_eq:NN \AtBeginShipoutOriginalShipout \tex_shipout:D
```

(*End of definition for* `\AtBeginShipoutOriginalShipout`*. This function is documented on page* 1065.)

`\ShipoutBoxHeight`
`\ShipoutBoxWidth`
`\ShipoutBoxDepth`

This is somewhat different from the original in atbegshi where `\ShipoutBoxHeight` etc. only holds the `\the\ht<box>` value. This may has some implications in some use cases and if that is a problem then it might need changing.

```
\cs_new:Npn \ShipoutBoxHeight { \dim_use:N \l_shipout_box_ht_dim }
\cs_new:Npn \ShipoutBoxDepth  { \dim_use:N \l_shipout_box_dp_dim }
\cs_new:Npn \ShipoutBoxWidth  { \dim_use:N \l_shipout_box_wd_dim }
\ExplSyntaxOff
```

(*End of definition for* `\ShipoutBoxHeight`*,* `\ShipoutBoxWidth`*, and* `\ShipoutBoxDepth`*.*)

```
⟨/atbegshi-ltx⟩
```

If the package is requested we substitute the one above:

```
⟨*2ekernel⟩
\declare@file@substitution{atbegshi.sty}{atbegshi-ltx.sty}
⟨/2ekernel⟩
```

6.3 Package **everyshi** emulation

This is now directly handled in that package so emulation is not necessary any more. Rather important :-)

```
⟨@@=⟩
```

File 54
ltoutput.dtx

1 Output Routine

1.1 Floats

The '2ekernel' code ensures that a `\usepackage{autoout1}` is essentially ignored if a 'full' format is being used that has the autoload file mode already in the format.

```
⟨defx⟩\begingroup
⟨defx⟩\makeatletter
⟨defx⟩\nfss@catcodes
⟨2ekernel⟩\expandafter\let\csname ver@autoout1.sty\endcsname\fmtversion
```

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
⟨*2ekernel⟩
\message{output,}
```

```
****************************************
*                OUTPUT                *
****************************************


PAGE LAYOUT PARAMETERS

\topmargin       : Extra space added to top of page.
@twoside         : boolean.  T if two-sided printing
\oddsidemargin   : IF @twoside = T
                     THEN extra space added to left of odd-numbered
                          pages.
                     ELSE extra space added to left of all pages.
\evensidemargin  : IF @twoside = T
                     THEN extra space added to left of even-numbered
                          pages.
\headheight      : height of head
\headsep         : separation between head and text
\footskip        : distance separation between baseline of last
                   line of text and baseline of foot.
                   Note difference between \footSKIP and \headSEP.
\textheight      : height of text on page, excluding head and foot
\textwidth       : width of printing on page
\columnsep       : IF @twocolumn = T
                     THEN width of space between columns
\columnseprule   : IF @twocolumn = T
                     THEN width of rule between columns (0 if none).
\columnwidth     : IF @twocolumn = T
                     THEN (\textwidth - \columnsep)/2
                     ELSE \textwidth
                   It is set by the \twocolumn and
                   \onecolumn commands.
```

`\@textbottom` : Command executed at bottom of vbox holding text of page (including figures). The `\raggedbottom` command almost `\let`'s this to `\vfil` (actually sets it to `\vskip \z@` plus.0001fil).
Should have depth 0pt.

`\@texttop` : Command executed at top of vbox holding text of page (including figures). Used by letter style; can also be used to produce centered pages.
Let to `\relax` by `\raggedbottom` and `\flushbottom`.

Page layout must initialize `\@colht` and `\@colroom` to `\textheight`.

PAGE STYLE PARAMETERS:

`\floatsep` : Space left between floats.
`\textfloatsep` : Space between last top float or first bottom float and the text.
`\topfigrule` : Command to place rule (or whatever) between floats at top of page and text. Executed in inner vertical mode right before the `\textfloatsep` skip separating the floats from the text. Must occupy zero vertical space. (See `\footnoterule`.)
`\botfigrule` : Same as `\topfigrule`, but put after the `\textfloatsep` skip separating text from the floats at bottom of page.
`\intextsep` : Space left on top and bottom of an in-text float.
`\dblfloatsep` : Space between double-column floats.
`\dbltextfloatsep` : Space between top double-column floats and text.
`\dblfigrule` : Similar to `\topfigrule`, but for double-column floats.
`\@fptop` : Glue to go at top of float column − must be 0pt + stretch
`\@fpsep` : Glue to go between floats in a float column.
`\@fpbot` : Glue to go at bottom of float column − must be 0pt + stretch
`\@dblfptop, \@dblfpsep, \@dblfpbot` : Analogous for double-column float page in two-column format.

FOOTNOTES: As in PLAIN, footnotes use `\insert\footins`.

PAGE LAYOUT SWITCHES AND MACROS

@twocolumn : Boolean. T if two columns per page globally.

PAGE STYLE MACROS AND SWITCHES

```
\@oddhead         : IF @twoside = T
                        THEN macro to generate head of odd-numbered
                             pages.
                        ELSE macro to generate head of all pages.
\@evenhead        : IF @twoside = T
                        THEN macro to generate head of even-numbered
                             pages.
\@oddfoot         : IF @twoside = T
                        THEN macro to generate foot of odd-numbered
                             pages.
                        ELSE macro to generate foot of all pages.
\@evenfoot        : IF @twoside = T
                        THEN macro to generate foot of even-numbered
                             pages.
@specialpage     : boolean.  T if current page is to have a special
                             format.
\@specialstyle  : If its value is  foo then
                   IF @specialpage = T
                     THEN the command \ps@foo is executed to
                          temporarily reset the page style parameters
                          before composing the current page.
                          This command should execute only \def's and
                          \edef's, making only local definitions.
```

FLOAT PLACEMENT PARAMETERS

```
The following parameters are set by the macro \@floatplacement.
When \@floatplacement is called,
\@colht is the height of the page or column being built.  I.e.:
        * For single-column page it equals \textheight.
        * For double-column page it equals \textheight - height
          of double-column floats on page.
Note that some are set globally and some locally:
   \@topnum  :=G Maximum number of floats allowed on the top of a
                 column.
   \@toproom :=G Maximum amount of top of column devoted to floats–
                 excluding \textfloatsep separation below the floats
                 and \floatsep separation between them.  For
                 two-column output, should be computed as a function
                 of \@colht.
   \@botnum, \@botroom
                : Analogous to above.
   \@colnum  :=G Maximum number of floats allowed in a column,
                 including in-text floats.
   \@textmin :=L Minimum amount of text (excluding footnotes) that
                 must appear on a text page.
                    %% 27 Sep 85 : made local to
                    %% \@addtocurcol and \@addtonextcol
                 It is now also used locally in processing double
                 floats.
```

`\@fpmin` :=L Minimum height of floats in a float column.

The macro `\@dblfloatplacement` sets the following parameters.

`\@dbltopnum` :=G Maximum number of double-column floats allowed at the top of a two-column page.

`\@dbltoproom` :=G Maximum height of double-column floats allowed at top of two-column page.

`\@fpmin` :=L Minimum height of floats in a float column.

It should also perform the following local assignments where necessary – i.e., where the new value differs from the old one:

```
\@fptop      :=L \@dblfptop
\@fpsep      :=L \@dblfpsep
\@fpbot      :=L \@dblfpbot
```

OUTPUT ROUTINE VARIABLES

`\@colht` : The total height of the current column. In single column style, it equals `\textheight`. In two-column style, it is `\textheight` minus the height of the double-column floats on the current page. MUST BE INITIALIZED TO `\textheight`.

`\@colroom` : The height available in the current column for text and footnotes. It equals `\@colht` minus the height of all floats committed to the top and bottom of the current column.

`\@textfloatsheight` : The total height of in-text floats on the current page.

`\footins` : Footnote insertion number.

`\@maxdepth` : Saved value of TeX's `\maxdepth`. Must be set when any routine sets `\maxdepth`.

CALLING THE OUTPUT ROUTINE

The output routine is called either by TeX's normal page-breaking mechanism, or by a macro putting a penalty `<` or = -10000 in the output list. In the latter case, the penalty indicates why the output routine was called, using the following code.

penalty	reason
-10000	`\pagebreak` `\newpage`
-10001	`\clearpage` (`\penalty` -10000 `\vbox{}` `\penalty` -10001)
-10002	float insertion, called from horizontal mode
-10003	float insertion, called from vertical mode.
-10004	float insertion.

Note: A float or marginpar puts the following sequence in the output list: (i) a penalty of -10004,
(ii) a null `\vbox`
(iii) a penalty of -10002 or -10003.
This solves two special problems:

1. If the float comes right after a `\newpage` or `\clearpage`, then the first penalty is ignored, but the second one invokes the output routine.
2. If there is a split footnote on the page, the second 'page' puts out the rest of the footnote.

THE OUTPUT ROUTINE

FUNCTIONS USED IN THE OUTPUT ROUTINE:

`\@outputpage` : Produces an output page with the contents of box `\@outputbox` as the text part.
Also sets `\@colht` :=G `\textheight`.
The page style is determined as follows.
IF @thispagestyle = true
THEN use `\thispagestyle` style
ELSE use ordinary page style.

`\@tryfcolumn\FLIST` : Tries to form a float column composed of floats from `\FLIST` (if nonempty) with the following parameters:
`\@colht` : height of box
`\@fpmin` : minimum height of floats in the box
`\@fpsep` : interfloat space
`\@fptop` : glue at top of box
`\@fpbot` : glue at bottom of box.
If it succeeds, then it does the following:
* `\@outputbox` :=L the composed float box.
* @fcolmade :=G true
* `\FLIST` :=G `\FLIST` - floats put in box
* `\@freelist` :=G `\@freelist` + floats put in box

If it fails, then:
* @fcolmade :=G false

NOTE: BIT MUST BE A SINGLE TOKEN!

`\@makefcolumn \FLIST` : Same as `\@tryfcolumn` except that it fails to make a float column only if `\FLIST` is empty. Otherwise, it makes a float column containing at least the first box in `\FLIST`, disregarding `\@fpmin`.

`\@startcolumn` :
Calls `\@tryfcolumn\@deferlist`. If `\@tryfcolumn` returns with (globally set) @fcolmade = false, then:
* Globally sets `\@toplist` and `\@botlist` to floats

from \@deferlist to go at top and bottom of column, deleting them from \@deferlist. It does this using \@colht as the total height, the page style parameters \@floatsep and \@textfloatsep, and the float placement parameters \@topnum, \@toproom, \@botnum, \@botroom, \@colnum and \textfraction.

* Globally sets \@colroom to \@colht minus the height of the added floats.

\@startdblcolumn :
Calls \@tryfcolumn\@dbldeferlist{8}. If \@tryfcolumn returns with (globally set) @fcolmade = false, then:

* Globally sets \@dbltoplist to floats from \@dbldeferlist to go at top and bottom of column, deleting them from \@dbldeferlist. It does this using \textheight as the total height, and the parameters \@dblfloatsep, etc.
* Globally sets \@colht to \textheight minus the height of the added floats.

\@combinefloats : Combines the text from box \@outputbox with the floats from \@toplist and \@botlist, putting the new box in \@outputbox. It uses \floatsep and \textfloatsep for the appropriate separations. It puts the elements of \TOPLIST and \BOTLIST onto \@freelist, and makes those lists null.

\@makecol : Makes the contents of \box255 plus the accumulated footnotes, plus the floats in \@toplist and \@botlist, into a single column of height \@colht (unless the page height has been locally changed), which it puts into box \@outputbox. It puts boxes in \@midlist back onto \@freelist and restores \maxdepth.

\@opcol : Outputs a column whose text is in box \@outputbox
If @twocolumn = false, then it calls \@outputpage, sets \@colht :=G \textheight, and calls \@floatplacement.

If @twocolumn = true, then:
If @firstcolumn = true, then it puts box \@outputbox into \@leftcolumn and sets @firstcolumn :=G false.

If @firstcolumn = false, then it puts out the current two-column page, any possible two-column float pages, and determines \@dbltoplist for the next page.

USER COMMANDS THAT CALL OR AFFECT THE OUTPUT ROUTINE

```
\newpage == BEGIN \par\vfil\penalty -10000 END

\clearpage == BEGIN \newpage
                    \write -1{}     % Part of hack to make sure no
                    \vbox{}         % \write's get lost.
                    \penalty -10001
              END

\cleardoublepage == BEGIN \clearpage
                          if @twoside = true and c@page is even
                            then \hbox{} \newpage fi
                    END
```

\twocolumn[BOX] : starts a new page, changing to twocolumn setting and puts BOX in a parbox of width \textwidth across the top. Useful for full-width titles for double-column pages.
SURPRISE: The stretch from \@dbltextfloatsep will be inserted between the BOX and the top of the two columns.

FLOAT-HANDLING MECHANISMS

The float environment obtains an insertion number B from the \@freelist (see below for a description of list manipulation), puts the float into box B and sets \count B to a FLOAT SPECIFIER. For a normal (not double-column) float, it then causes a page break in one of the following two ways:

- In outer hmode: \vadjust{\penalty -10002}
- In vmode : \penalty -10003.

For a double-column float, it puts B onto the \@dbldeferlist.
The float specifier has two components:

* A PLACEMENT SPECIFICATION, describing where the float may be placed.
* A TYPE, which is a power of two–e.g., figures might be type 1 floats, tables type 2 floats, programs type 4 floats, etc.

The float specifier is encoded as follows, where bit 0 is the least significant bit.

Bit	Meaning
0	1 iff the float may go where it appears in the text.
1	1 iff the float may go on the top of a page.
2	1 iff the float may go on the bottom of a page.
3	1 iff the float may go on a float page.
4	1 unless the PLACEMENT includes a !
5	1 iff a type 1 float

6 1 iff a type 2 float
etc.

A negative float specifier is used to indicate a marginal note.

MACROS AND DATA STRUCTURES FOR PROCESSING FLOATS

A FLOAT LIST consisting of the floats in boxes `\boxa` ... `\boxN` has the form:

```
        \@elt \boxa ... \@elt \boxN
```

where `\boxI` is defined by

```
        \newinsert\boxI
```

Normally, `\@elt` is `\let` to `\relax`. A test can be performed on the entire float list by locally `\def`'ing `\@elt` appropriately and executing the list.
This is a lot more efficient than looping through the list.

The following macros are used for manipulating float lists.

```
\@next \CS \LIST {NONEMPTY}{EMPTY} ==  %% NOTE: ASSUME \@elt
= \relax
  BEGIN  assume that \LIST == \@elt \B1 ... \@elt \Bn
         if n = 0
           then  EMPTY
           else  \CS     :=L \B1
                 \LIST  :=G \@elt \B2 ... \@elt \Bn
                 NONEMPTY
         fi
  END
```

`\@bitor\NUM\LIST` : Globally sets switch @test to the disjunction for all I of bit log2 `\NUM` of the float specifiers of all the floats in `\LIST`.
I.e., @test is set to true iff there is at least one float in `\LIST` having bit log2 `\NUM` of its float specifier equal to 1.

Note: log2 [(`\count` I)/32] is the bit number corresponding to the type of float I. To see if there is any float in `\LIST` having the same type as float I, you run `\@bitor` with
`\NUM` = [(`\count` I)/32] * 32.

```
\@bitor\NUM\LIST ==
  BEGIN
     @test :=G false
     { \@elt \CTR ==  if \NUM <> 0 then
                        if \count\CTR / \NUM is odd
                           then  @test := true        fi fi
```

```
      \LIST
    }
  END
```

`\@cons\LIST\NUM` : Globally sets `\LIST := \LIST * \@elt \NUM`

```
\@cons\LIST\NUM ==
  BEGIN {  \@elt == \relax
           \LIST :=G \LIST \@elt \NUM
        }
```

BOX LISTS FOR FLOAT-PLACEMENT ALGORITHMS

`\@freelist`	: List of empty boxes for placing new floats.
`\@toplist`	: List of floats to go at top of current column.
`\@midlist`	: List of floats in middle of current column.
`\@botlist`	: List of floats to go at bottom of current column.
`\@deferlist`	: List of floats to go after current column.
`\@dbltoplist`	: List of double-col. floats to go at top of current page.
`\@dbldeferlist`	: List of double-column floats to go on subsequent pages.

FLOAT-PLACEMENT ALGORITHMS

`\@addtobot` : Tries to put insert `\@currbox` on `\@botlist.`
Called only when:
* `\ht` BOX `<` `\@colroom`
* type of `\@currbox` not on `\@deferlist`
* `\@colnum` `>` 0
* @insert = false

If it succeeds, then:
* sets @insert true
* decrements `\@botroom` by `\ht` BOX
* decrements `\@botnum` and `\@colnum` by 1
* decrements `\@colroom` by `\ht` BOX + either `\floatsep` or `\textfloatsep,` as appropriate.
* sets `\maxdepth` to 0pt

`\@addtotoporbot` : Tries to put insert `\@currbox` on `\@toplist` or `\@botlist.`
Called only under same conditions as `\@addtobot.`
If it succeeds, then:
* sets @insert true
* decrements `\@toproom` or `\@botroom` by `\ht` BOX
* decrements `\@colnum` and either `\@topnum` or `\@botnum` by 1
* decrements `\@colroom` by `\ht` BOX + `\floatsep`

or \textfloatsep, as appropriate.

\@addtocurcol : Tries to add \@currbox to current column, setting @insert true if it succeeds, false otherwise.
It will add \@currbox to top only if bit 0 of \count \@currbox is 0, and to the bottom only if bit 0 = 0 or an earlier float of the same type is put on the bottom.
If the float is put in the text, then \penalty\interlinepenalty is put right after the float, before the following \vskip, and \outputpenalty :=L 0.

\@addtonextcol : Tries to add \@currbox to the next column, setting @insert true if it succeeds, false otherwise.

\@addtodblcol : Tries to add \@currbox to the next double-column page, adding it to \@dbltoplist if it succeeds and \@dbldeferlist if it fails.

```
\@addmarginpar ==
 BEGIN
   if \@currlist nonempty
     then remove \@marbox from \@currlist
          add \@marbox and \@currbox to \@freelist
               %% NOTE: \@currbox = left box
     else LaTeX error: ?  %% shouldn't happen
   fi
   \@tempcnta := 1      %% 1 = right, -1 = left
   if @twocolumn = true
     then if @firstcolumn = true
             then \@tempcnta := -1
          fi
     else if @mparswitch = true
             then if count0 odd
                     else \@tempcnta := -1
                  fi
          fi
          if @reversemargin = true
             then \@tempcnta := -\@tempcnta
          fi
   fi
   if \@tempcnta < 0 then \box\@marbox :=G \box\@currbox
   fi
   \@tempdima   :=L maximum(\@mparbottom - \@pageht
                                          + ht of \@marbox, 0)
   if \@tempdima > 0 then LaTeX warning: 'marginpar moved' fi
   \@mparbottom :=G \@pageht + \@tempdima + depth of \@marbox
                        + \marginparpush
```

```
  \@tempdima   :=L \@tempdima - ht of \@marbox
  \box\@marbox :=G \box\@currbox
                            \vbox { \vskip \@tempdima
                                    \box\@marbox
                                  }
  height of \@marbox :=G depth of \@marbox :=G 0
  \kern -\@pagedp
  \nointerlineskip
  \hbox{ if @tempcnta > 0 then \hskip \columnwidth
                               \hskip \marginparsep
                          else \hskip -\marginparsep
                               \hskip -\marginparwidth
         fi
         \box\@marbox \hss
       }
  \nobreak
  \nointerlineskip
  \hbox{\vrule height 0 width 0 depth \@pagedp}
END
```

Floats and marginpars add a lot of dead cycles.
End of historical LaTeX 2.09 comments.

```
\maxdeadcycles = 100
\let\@elt\relax
\def\@next#1#2#3#4{\ifx#2\@empty #4\else
   \expandafter\@xnext #2\@@#1#2#3\fi}
\def\@xnext \@elt #1#2\@@#3#4{\def#3{#1}\gdef#4{#2}}

\def\@testfalse{\global\let\if@test\iffalse}
\def\@testtrue {\global\let\if@test\iftrue}
\@testfalse

\def\@bitor#1#2{\@testfalse {\let\@elt\@xbitor
   \@tempcnta #1\relax #2}}
```

RmS 91/11/22: Added test for `\count#1 = 0`. Suggested by Chris Rowley.

```
\def\@xbitor #1{\@tempcntb \count#1
   \ifnum \@tempcnta =\z@
   \else
     \divide\@tempcntb\@tempcnta
     \ifodd\@tempcntb \@testtrue\fi
   \fi}
```

DEFINITION OF FLOAT BOXES:

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/10/01}%
⟨latexrelease⟩                  {\bx@ZZ}{Extended float list}%
⟨*2ekernel | latexrelease⟩
\let\@elt\newinsert
⟨*2ekernel⟩
\def\@freelist{%
```

```
  \@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
  \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
  \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
  \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}
\@freelist
⟨/2ekernel⟩
\ifx\numexpr\@undefined\else
\def\reserved@a{%
  \@elt\bx@S\@elt\bx@T\@elt\bx@U\@elt\bx@V
  \@elt\bx@W\@elt\bx@X\@elt\bx@Y\@elt\bx@Z
  \@elt\bx@AA\@elt\bx@BB\@elt\bx@CC\@elt\bx@DD\@elt\bx@EE
  \@elt\bx@FF\@elt\bx@GG\@elt\bx@HH\@elt\bx@II\@elt\bx@JJ
  \@elt\bx@KK\@elt\bx@LL\@elt\bx@MM\@elt\bx@NN
  \@elt\bx@OO\@elt\bx@PP\@elt\bx@QQ\@elt\bx@RR
  \@elt\bx@SS\@elt\bx@TT\@elt\bx@UU\@elt\bx@VV
  \@elt\bx@WW\@elt\bx@XX\@elt\bx@YY\@elt\bx@ZZ}
\reserved@a
\def\@elt{\noexpand\@elt\noexpand}
\edef\@freelist{\@freelist\reserved@a}
\fi
\let\reserved@a\relax
\let\@elt\relax
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\bx@ZZ}{Extended float list}%
⟨latexrelease⟩\def\@freelist{%
⟨latexrelease⟩  \@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
⟨latexrelease⟩  \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
⟨latexrelease⟩  \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
⟨latexrelease⟩  \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}
⟨latexrelease⟩  \insc@unt=234
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

```
\gdef\@toplist{}
\gdef\@botlist{}
\gdef\@midlist{}
\gdef\@currlist{}
\gdef\@deferlist{}
\gdef\@dbltoplist{}
```

The new algorithm stores page wide floats together with column floats in a single `\@deferlist` list. We keep `\@dbldeferlist` initialised as empty so that packages that are testing for deferred floats can use the same code for old or new float handling.

```
\gdef\@dbldeferlist{}
```

PAGE LAYOUT PARAMETERS

```
\newdimen\topmargin
\newdimen\oddsidemargin
\newdimen\evensidemargin
\let\@themargin=\oddsidemargin
\newdimen\headheight
\newdimen\headsep
\newdimen\footskip
```

```
\newdimen\textheight
\newdimen\textwidth
\newdimen\columnwidth
\newdimen\columnsep
\newdimen\columnseprule
\newdimen\marginparwidth
\newdimen\marginparsep
\newdimen\marginparpush
```

\AtBeginDvi
\@begindvibox

We use a box register in which to put stuff that must appear before anything else in the .dvi file.

The stuff in the box should not add any typeset material to the page when it is unboxed.

This interface is no longer used. Instead a new one is inside ltshipout.dtx. We only keep the box in case some old code refers to it directly (or we do some rollback).

```
\newbox\@begindvibox
%\DeclareRobustCommand \AtBeginDvi [1]{%
%  \global \setbox \@begindvibox
%    \vbox{\unvbox \@begindvibox #1}%
%}
```

(*End of definition for* \AtBeginDvi *and* \@begindvibox*. These functions are documented on page* *1062*.)

\@maxdepth

This is not the right place to set this; it needs to be set in a class/style file when \maxdepth is set.

Also, many settings to \maxdepth should be to \@maxdepth, probably?

```
\newdimen\@maxdepth
\@maxdepth = \maxdepth
```

(*End of definition for* \@maxdepth.)

\paperheight
\paperwidth

New \paper... registers.

```
\newdimen\paperheight
\newdimen\paperwidth
```

(*End of definition for* \paperheight *and* \paperwidth.)

\stockheight
\stockwidth

New \stock... registers.

```
\newdimen\stockheight
\newdimen\stockwidth
```

(*End of definition for* \stockheight *and* \stockwidth.)

\if@insert
\if@fcolmade
\if@specialpage
\if@firstcolumn
\if@twocolumn
\if@twoside
\if@reversemargin
\if@mparswitch
\col@number

Local switches first:

```
\newif \if@insert
```

These should definitely be global:

```
\newif \if@fcolmade
\newif \if@specialpage \@specialpagefalse
```

These should be global but are not always set globally in other files.

```
\newif \if@firstcolumn \@firstcolumntrue
\newif \if@twocolumn   \@twocolumnfalse
```

Not sure about these: two questions. Should things which must apply to a whole document be local or global (they probably should be 'preamble only' commands)? Are these three such things?

```
\newif \if@twoside     \@twosidefalse
\newif \if@reversemargin \@reversemarginfalse
\newif \if@mparswitch  \@mparswitchfalse
```

This counter has been imported from 'multicol'.

```
\newcount \col@number
\col@number \@ne
```

(*End of definition for* `\if@insert` *and others.*)

Historical LaTeX 2.09 comments (not necessarily accurate any more):

INTERNAL REGISTERS

```
\newcount\@topnum
\newdimen\@toproom
\newcount\@dbltopnum
\newdimen\@dbltoproom
\newcount\@botnum
\newdimen\@botroom
\newcount\@colnum
\newdimen\@textmin
\newdimen\@fpmin
\newdimen\@colht
\newdimen\@colroom
\newdimen\@pageht
\newdimen\@pagedp
\newdimen\@mparbottom \@mparbottom\z@
\newcount\@currtype
\newbox\@outputbox
\newbox\@leftcolumn
\newbox\@holdpg
```

```
\def\@thehead{\@oddhead} % initialization
\def\@thefoot{\@oddfoot}
```

End of historical LaTeX 2.09 comments.

`\clearpage` The tests at the beginning are an experimental attempt to avoid a completely empty page after a `\twocolumn[...]`. This prevents the text from the argument vanishing into a float box, never to be seen again. We hope that it does not produce wrong formatting in other cases.

```
\def\clearpage{%
  \ifvmode
    \ifnum \@dbltopnum =\m@ne
      \ifdim \pagetotal <\topskip
        \hbox{}%
      \fi
    \fi
  \fi
  \newpage
  \write\m@ne{}%
```

```
  \vbox{}%
  \penalty -\@Mi
}
```

(*End of definition for* \clearpage.)

\cleardoublepage

```
\def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
    \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
⟨/2ekernel⟩
```

(*End of definition for* \cleardoublepage.)

\onecolumn

```
⟨*2ekernel | fltrace⟩
\def\onecolumn{%
  \clearpage
  \global\columnwidth\textwidth
  \global\hsize\columnwidth
  \global\linewidth\columnwidth
  \global\@twocolumnfalse
  \col@number \@ne
  \@floatplacement}
```

(*End of definition for* \onecolumn.)

\newpage The two checks at the beginning ensure that an item label or run-in section title immediately before a \newpage get printed on the correct page, the one before the page break.

All three tests are largely to make error processing more robust; that is why they all reset the flags explicitly, even when it would appear that this would be done by a \leavevmode.

```
⟨/2ekernel | fltrace⟩
⟨latexrelease⟩\IncludeInRelease{2017/04/15}%
⟨latexrelease⟩                  {\newpage}{Check depth of page}%
⟨*2ekernel | latexrelease | fltrace⟩
\def \newpage {%
  \if@noskipsec
    \ifx \@nodocument\relax
      \leavevmode
      \global \@noskipsecfalse
    \fi
  \fi
  \if@inlabel
    \leavevmode
    \global \@inlabelfalse
  \fi
  \if@nobreak \@nobreakfalse \everypar{}\fi
  \par
```

The \vfil at the end of the macro before the break penalty will normally result in the page being run short, even with \flushbottom in effect (in contrast to the behavior of \pagebreak). However, if there is some explicit stretch on the page, say, a \vfill, it has the undesired side-effect, that the last line will not align at its baseline if it contains characters going below the baseline, as the value of \prevdepth is no longer taken into

account by TEX. So we back up by that amount (or by \maxdepth if it is really huge), to mimic the normal behavior without the \newpage.

```
  \ifdim\prevdepth>\z@
     \vskip -%
       \ifdim\prevdepth>\maxdepth
         \maxdepth
       \else
         \prevdepth
       \fi
  \fi
  \vfil
  \penalty -\@M}
⟨/2ekernel | latexrelease | fltrace⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\newpage}{Check depth of page}%
⟨latexrelease⟩\def \newpage {%
⟨latexrelease⟩  \if@noskipsec
⟨latexrelease⟩    \ifx \@nodocument\relax
⟨latexrelease⟩      \leavevmode
⟨latexrelease⟩      \global \@noskipsecfalse
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@inlabel
⟨latexrelease⟩    \leavevmode
⟨latexrelease⟩    \global \@inlabelfalse
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@nobreak \@nobreakfalse \everypar{}\fi
⟨latexrelease⟩  \par
⟨latexrelease⟩  \vfil
⟨latexrelease⟩  \penalty -\@M}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel | fltrace⟩
```

(*End of definition for* \newpage.)

\@emptycol

It may be better to use an invisible rule rather than an empty box here.

```
\def \@emptycol {\vbox{}\penalty -\@M}
```

(*End of definition for* \@emptycol.)

\twocolumn
\@topnewpage

There are several bug fixes to the two-column stuff here.

```
\def \twocolumn {%
  \clearpage
  \global\columnwidth\textwidth
  \global\advance\columnwidth-\columnsep
  \global\divide\columnwidth\tw@
  \global\hsize\columnwidth
  \global\linewidth\columnwidth
  \global\@twocolumntrue
  \global\@firstcolumntrue
  \col@number \tw@
```

There is no reason to put a \@dblfloatplacement here since \@topnewpage ignores these settings. The \@floatplacement is needed in case this comes after some changes.

```
  \@ifnextchar [\@topnewpage\@floatplacement
}
```

Note that here, getting a box from the freelist can assume success since this comes just after a \clearpage.

```
\long\def \@topnewpage [#1]{%
  \@nodocument
  \@next\@currbox\@freelist{}{}%
  \global \setbox\@currbox
    \color@vbox
      \normalcolor
      \vbox {%
        \hsize\textwidth
        \@parboxrestore
        \col@number \@ne
        #1%
        \vskip -\dbltextfloatsep
             }%
    \color@endbox
```

Added size test and warning message; perhaps we should use an error message.

```
  \ifdim \ht\@currbox>\textheight
    \ht\@currbox \textheight
  \fi
```

This next line is not essential but it is more robust to make this value non-zero, in case of weird errors.

This next bit is what is needed from \@addtodblcol, plus some extra checks for error trapping.

```
  \global \count\@currbox \tw@
  \@tempdima -\ht\@currbox
  \advance \@tempdima -\dbltextfloatsep
  \global \advance \@colht \@tempdima
  \ifx \@dbltoplist \@empty
  \else
    \@latex@error{Float(s) lost}\@ehb
    \let \@dbltoplist \@empty
  \fi
  \@cons \@dbltoplist \@currbox
```

This setting of \@dbltopnum is used only to change the typesetting in \@combinedblfloats.

```
  \global \@dbltopnum \m@ne
⟨*trace⟩
    \fl@trace{dbltopnum set to -1 (= \the \@dbltopnum) (topnewpage)}%
⟨/trace⟩
```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present; but note that this value is larger than that used when checking that page is too full of normal floats.

If there is little room left we just force a page-break, OK? This involves producing two empty columns. The second empty column may be produced by \output, in which case an extra, misleading, warning will be generated, OK? (This happens only when there

is too little room left on the page for any float.) Otherwise (i.e. if the size is such that it is allowed as a normal float) the extra `\@emptycol` will be invoked in the second column by the conditional code guarded by the `\if@firstcolumn` test.

I now think that the cut-off point here should be `3\baselineskip`, but we make it a bit less so that 3 lines of text will be allowed, OK?

Since this happens only when there is nothing on the page but the 'top-box', the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

Here we need two page-ends since both columns need to be empty.

```
  \ifdim \@colht<2.5\baselineskip
    \@latex@warning@no@line {Optional argument of \noexpand\twocolumn
                  too tall on page \thepage}%
    \@emptycol
    \if@firstcolumn
    \else
      \@emptycol
    \fi
  \else
    \global \vsize \@colht
    \global \@colroom \@colht
    \@floatplacement
  \fi
}
```

(*End of definition for* `\twocolumn` *and* `\@topnewpage`.)

`\output` `\@specialoutput`

This needs some small adjustments. We cannot guarantee that the float mechanism will interact correctly with this stuff, but that mechanism does not always work properly with footnotes already.

RmS 91/09/29:

added reset of `\par` to the output routine. This avoids problems when the output routine is called within a list where `\par` may be a no-op.

```
\output {%
  \let \par \@@par
  \ifnum \outputpenalty<-\@M
    \@specialoutput
  \else
    \@makecol
    \@opcol
```

Moved to `\@opcol`: `\@floatplacement`.

```
    \@startcolumn
```

This loop could be replaced by an `\expandafter` tail recursion in `\@startcolumn`.

```
    \@whilesw \if@fcolmade \fi
      {%
⟨*trace⟩
        \fl@trace{PAGE: float \if@twocolumn column \else page \fi
                      completed}%
⟨/trace⟩
        \@opcol\@startcolumn}%
  \fi
  \ifnum \outputpenalty>-\@Miv
```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present. If there is little room left we just force a page-break, OK?

This bit is essential only if a float has just been processed so maybe it should be moved; but this is the natural place at which to set the vsize and a test would need to be done anyway. A check has been added to ensure that there really has been a change in the value of `\@colroom`.

Since this happens only when there is nothing on the page but floats, the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

The twocolumn case does not need any extra code here since this is the `\output` itself; in the second column there will still not be enough room left so `\@emptycol` will be executed again when the OR is called by the-page builder when it gets to the penalty inserted by the first execution. (The page-builder is never invoked whilst the OR is being executed since it builds a inner vlist; thus any conditional code for the two-column case within `\output` may not get executed with the correct value of `\if@firstcolumn`.

```
    \ifdim \@colroom<1.5\baselineskip
      \ifdim \@colroom<\textheight
        \@latex@warning@no@line {Text page \thepage\space
                                 contains only floats}%
        \@emptycol
%         \if@twocolumn
%           \if@firstcolumn
%           \else
%             \@emptycol
%           \fi
%         \fi
      \else
        \global \vsize \@colroom
      \fi
    \else
      \global \vsize \@colroom
    \fi
  \else
    \global \vsize \maxdimen
  \fi
}
```

Historical LaTeX 2.09 comments (not necessarily accurate any more):

CHANGES TO `\@specialoutput:`

* `\penalty\z@` changed to `\penalty\interlinepenalty` so `\samepage` works properly with figure and table environments.
  (Changed 23 Oct 86)

* Definition of `\@specialoutput` changed 26 Feb 88 so `\@pageht` and `\@pagedp` aren't changed for a marginal note.
  (Change suggested by Chris Rowley.)

End of historical LaTeX 2.09 comments.

```
\gdef\@specialoutput{%
   \ifnum \outputpenalty>-\@Mii
     \@doclearpage
```

```
  \else
    \ifnum \outputpenalty<-\@Miii
      \ifnum \outputpenalty<-\@MM \deadcycles \z@ \fi
      \global \setbox\@holdpg \vbox {\unvbox\@cclv}%
    \else
```

Note that `\boxmaxdepth` should not be set here since we wish to record the natural depth of the holdpg box.

This is changed so as to not lose anything, such as writes and marks, which may get into box 255 and should be returned to the list. This should only happen when the first penalty in the mechanism is discarded and therefore `\@holdpg` should always be void in this case. This can happen because a penalty is discarded whenever there is no box on the list.

It was just: `\setbox\@tempboxa \box \@cclv`.

The last box which is removed is the box put there by the double-penalty mechanism. The `\unskip` then removes the `\topskip` which is put there since the box is the first on the page.

```
      \global \setbox\@holdpg \vbox{%
                      \unvbox\@holdpg
                      \unvbox\@cclv
```

We must now remove the box added by the float mechanism and the `\topskip` glue therefore added above it by TEX.

```
                      \setbox\@tempboxa \lastbox
                      \unskip
                                    }%
```

These two are needed as separate dimensions only by `\@addmarginpar`; for other purposes we put the whole size into `\@pageht` (see below).

```
      \@pagedp \dp\@holdpg
      \@pageht \ht\@holdpg
      \unvbox \@holdpg
      \@next\@currbox\@currlist{%
        \ifnum \count\@currbox>\z@
```

Putting the whole size into `\@pageht` (see above).

```
          \advance \@pageht \@pagedp
          \ifvoid\footins \else
            \advance \@pageht \ht\footins
            \advance \@pageht \skip\footins
            \advance \@pageht \dp\footins
          \fi
          \ifvbox \@kludgeins
```

We want to make the adjustment due to this insert only if the non-star form is used. The *-form will probably not work with floats, but maybe it still could make some adjustment here even so?

```
            \ifdim \wd\@kludgeins=\z@
              \advance \@pageht \ht\@kludgeins
⟨*trace⟩
              \fl@trace {Extra size added: \the \ht\@kludgeins}%
⟨/trace⟩
            \fi
          \fi
```

This version puts the inserts back just before the additional material; it could be moved earlier, before unboxing the page-so-far. Neither is guaranteed not to put things on the wrong page. This version is similar to the original version.

```
            \@reinserts
            \@addtocurcol
          \else
            \@reinserts
            \@addmarginpar
          \fi
          }\@latexbug
```

A 2e change: use `\addpenalty` instead of `\penalty` here. Some penalty is needed to create a potential break-point immediately after the reinserts (or the marginal). Otherwise there can be no possibility to break here and this can cause the reinserts or the marginal to appear on the next page (which is often incorrect). However, if the nobreak flag is true, a `\nobreak` must be correct.

```
        \ifnum \outputpenalty<\z@
          \if@nobreak
            \nobreak
          \else
            \addpenalty \interlinepenalty
          \fi
        \fi
      \fi
    \fi
}
⟨/2ekernel | fltrace⟩
```

(*End of definition for* `\output` *and* `\@specialoutput`.)

`\@testwrongwidth` `\f@depth`

Test if the float box has the wrong width when trying to place it into some area. (Actually the test is for a conventional depth setting rather than for the width of the float. For that reason the box depth was explicitly tailored when the float was created).

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\@testwrongwidth}{float order in 2-column}%
⟨*2ekernel | latexrelease | fltrace⟩
\def\@testwrongwidth #1{%
  \ifdim\dp#1=\f@depth
⟨*trace⟩
    \fl@trace{\string#1
              \ifdim\f@depth=\z@ single \else double \fi
              column float -- ok}%
⟨/trace⟩
  \else
    \global\@testtrue
⟨*trace⟩
    \fl@trace{\string#1
              \ifdim\f@depth=\z@ double \else single \fi
              column float -- wrong}%
⟨/trace⟩
  \fi}%
```

Normally looking for single column floats, which have zero depth.

```
\let\f@depth\z@
```

```
⟨/2ekernel | latexrelease | fltrace⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\@testwrongwidth}{float order in 2-column}%
⟨latexrelease⟩\let\@testwrongwidth\@undefined
⟨latexrelease⟩\let\f@depth\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\@testwrongwidth` *and* `\f@depth`.)

`\@doclearpage` This is a very much an emergency action, just dumping everything: footnotes first then floats. A more sophisticated version is needed; but even more urgent is a bug-free version (see, for example, pr/3528).

Also, it puts any left-over non-boxes (writes, specials, etc.) back after any float pages created: this is a very bad bug since, for example, a kludge insert will be in quite the wrong place and, worse, be irremovable and uncancelable.

All the remaining changes are replacing the double column defer list or inserting the extra test `\@testwrongwidth{`⟨*box*⟩`}` at suitable places. That is at places where a box is taken off the deferlist.

```
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@doclearpage}%
⟨latexrelease⟩                              {float order in 2-column}%
⟨*2ekernel | latexrelease⟩
\def \@doclearpage {%
     \ifvoid\footins
       \ifvbox\@kludgeins
         {\setbox \@tempboxa \box \@kludgeins}%
⟨*trace⟩
         \fl@trace {kludgeins box made void}%
⟨/trace⟩
       \fi
       \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
       \setbox\@tempboxa\box\@cclv
       \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
       \global \let \@toplist \@empty
       \global \let \@botlist \@empty
       \global \@colroom \@colht
       \ifx \@currlist\@empty
       \else
          \@latex@error{Float(s) lost}\@ehb
          \global \let \@currlist \@empty
       \fi
       \@makefcolumn\@deferlist
       \@whilesw\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
       \if@twocolumn
         \if@firstcolumn
           \xdef\@deferlist{\@dbltoplist\@deferlist}%
           \global \let \@dbltoplist \@empty
           \global \@colht \textheight
           \begingroup
              \@dblfloatplacement
```

```
              \@makefcolumn\@deferlist
              \@whilesw\if@fcolmade \fi{\@outputpage
                                        \@makefcolumn\@deferlist}%
            \endgroup
          \else
            \vbox{}\clearpage
          \fi
        \fi
```

the next line is needed to avoid losing floats in certain circumstances a single call to the original `\doclearpage` will now no longer output all floats.

```
        \ifx\@deferlist\@empty \else\clearpage \fi
      \else
        \setbox\@cclv\vbox{\box\@cclv\vfil}%
        \@makecol\@opcol
        \clearpage
      \fi
}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@doclearpage}%
⟨latexrelease⟩                              {float order in 2-column}%
⟨latexrelease⟩\def \@doclearpage {%
⟨latexrelease⟩      \ifvoid\footins
```

We empty any left over kludge insert box here; this is a temporary fix. It should perhaps be applied to one page of cleared floats, but who cares? The whole of this stuff needs completely redoing for many such reasons.

```
⟨latexrelease⟩        \ifvbox\@kludgeins
⟨latexrelease⟩          {\setbox \@tempboxa \box \@kludgeins}%
⟨*trace⟩
⟨latexrelease⟩          \fl@trace {kludgeins box made void}%
⟨/trace⟩
⟨latexrelease⟩        \fi
⟨latexrelease⟩        \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
⟨latexrelease⟩        \setbox\@tempboxa\box\@cclv
⟨latexrelease⟩        \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
⟨latexrelease⟩        \global \let \@toplist \@empty
⟨latexrelease⟩        \global \let \@botlist \@empty
⟨latexrelease⟩        \global \@colroom \@colht
⟨latexrelease⟩        \ifx \@currlist\@empty
⟨latexrelease⟩        \else
⟨latexrelease⟩           \@latexerr{Float(s) lost}\@ehb
⟨latexrelease⟩           \global \let \@currlist \@empty
⟨latexrelease⟩        \fi
⟨latexrelease⟩        \@makefcolumn\@deferlist
⟨latexrelease⟩        \@whilesw\if@fcolmade \fi
⟨latexrelease⟩                      {\@opcol\@makefcolumn\@deferlist}%
⟨latexrelease⟩        \if@twocolumn
⟨latexrelease⟩          \if@firstcolumn
⟨latexrelease⟩            \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%
```

```
⟨latexrelease⟩            \global \let \@dbltoplist \@empty
⟨latexrelease⟩            \global \@colht \textheight
⟨latexrelease⟩            \begingroup
⟨latexrelease⟩               \@dblfloatplacement
⟨latexrelease⟩               \@makefcolumn\@dbldeferlist
⟨latexrelease⟩               \@whilesw\if@fcolmade \fi
⟨latexrelease⟩                     {\@outputpage\@makefcolumn\@dbldeferlist}%
⟨latexrelease⟩            \endgroup
⟨latexrelease⟩          \else
⟨latexrelease⟩            \vbox{}\clearpage
⟨latexrelease⟩          \fi
⟨latexrelease⟩        \fi
⟨latexrelease⟩      \else
⟨latexrelease⟩        \setbox\@cclv\vbox{\box\@cclv\vfil}%
⟨latexrelease⟩        \@makecol\@opcol
⟨latexrelease⟩        \clearpage
⟨latexrelease⟩      \fi
⟨latexrelease⟩   }%
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\@doclearpage`.)

`\@opcol` Several changes in detail here.

```
⟨*2ekernel | fltrace⟩
\def \@opcol {%
  \if@twocolumn
```

The funny-looking internal commands are interfacing with the new marks mechanism. We make sure (elsewhere) that those are always defined, even when we roll back, so here we add them unconditionally. This still need turning into a hook or config point eventually:

```
    \@expl@@@mark@update@dblcol@structures@@
    \@outputdblcol
  \else
    \@expl@@@mark@update@singlecol@structures@@
    \@outputpage
⟨*trace⟩
    \fl@trace{PAGE: one column (float? see above) page completed}%
⟨/trace⟩
```

Not needed since it comes after `\@outputpage`:

```
%   \global\@colht\textheight
  \fi
```

These do not need to be done every time `\@opcol` is used: they should be grouped together since they all need to be done at the end of the non-special output routine, or at the end of a clearpage one.

```
  \global \@mparbottom \z@ \global \@textfloatsheight \z@
  \@floatplacement
}
⟨/2ekernel | fltrace⟩
```

(*End of definition for* `\@opcol`.)

\@makecol We must rewrite this macro to allow for variations in page-makeup required by changes in page-length.

This uses a different macro if a special-length column is being produced.

```
⟨*2ekernel⟩
\gdef \@makecol {%
  \ifvoid\footins
    \setbox\@outputbox \box\@cclv
  \else
    \setbox\@outputbox \vbox {%
```

This \boxmaxdepth setting is to ensure that deep footnotes do not overwrite the footer (on account of the negative skip added later): it should use \@maxdepth otherwise the change is pointless when there are footnotes.

But see also its use when combining floats.

```
      \boxmaxdepth \@maxdepth
%       \@tempdima\dp\@cclv
      \unvbox \@cclv
%       \vskip-\@tempdima
      \vskip \skip\footins

      \color@begingroup
        \normalcolor
        \footnoterule
        \unvbox \footins
      \color@endgroup
      }%
  \fi
```

The h floats have now been finally committed to this page so we can reset their list. The top and bottom floats are then added to the page.

```
  \let\@elt\relax
  \xdef\@freelist{\@freelist\@midlist}%
  \global \let \@midlist \@empty
  \@combinefloats
```

The variations start here in case \enlargethispage has been used.

```
  \ifvbox\@kludgeins
    \@makespecialcolbox
  \else
```

This extra reboxing is only needed to add the \@texttop and \@textbotttom but this could be done earlier, when the floats are added.

The \boxmaxdepth resetting here will have no effect unless \@textbottom ends with a box or rule. So is this (or possibly \@maxdepth) the correct value?

The \vskip -\dimen@ ensures that the visible depth of the box does not affect the placement of anything on the page. Thus very deep pages will overprint the footer; but these should have been prevented by suitable settings of the maxdepths at appropriate times.

If \@textbottom ends with a box or rule of non-zero depth then this skip adjustment should be done again after it.

I think that the final boxing of the main text page could have a common ending which may make it simpler to see what is going on.

This needs further investigation, especially in the 'special case'.

Also, the `\boxmaxdepth` setting here affects what happens within `\@texttop` and `\@textbottom`, should it? Is it needed at all?

RmS 91/10/22: Replaced `\dimen128` by `\dimen@`.

```
    \setbox\@outputbox \vbox to\@colht {%
%       \boxmaxdepth \maxdepth                  %??
      \@texttop
      \dimen@ \dp\@outputbox
      \unvbox \@outputbox
      \vskip -\dimen@
      \@textbottom
      }%
  \fi
  \global \maxdepth \@maxdepth
}
```

(*End of definition for* `\@makecol`.)

`\@reinserts` This is the code which reinserts the inserts. It puts them all in one place; this can make some of them come out on the wrong page. It has been put into a separate macro to expedite experimentation.

```
\gdef \@reinserts{%
  \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
  \ifvbox\@kludgeins\insert\@kludgeins
                      {\unvbox\@kludgeins}\fi
}
⟨/2ekernel⟩
```

(*End of definition for* `\@reinserts`.)

`\@makespecialcolbox` This implements certain variations in page-makeup.

```
⟨*2ekernel | fltrace⟩
\gdef \@makespecialcolbox {%
⟨*trace⟩
  \fl@trace{Kludgeins ht \the\ht\@kludgeins\space
                          dp \the\dp\@kludgeins\space
                          wd \the\wd\@kludgeins}%
⟨/trace⟩
```

First we find the natural height of the column.

See above for discussion of what is happening here.

This needs further investigation, especially in this 'special case'.

```
  \setbox\@outputbox \vbox {%
    \@texttop
    \dimen@ \dp\@outputbox
    \unvbox\@outputbox
    \vskip-\dimen@
    }%
  \@tempdima \@colht
  \ifdim \wd\@kludgeins>\z@
```

Note that in this case (the *-version), the height of the `\@kludgeins` box is not used since its value is somewhat arbitrary: it need only be big enough to ensure that the page-break is not taken prematurely.

Here we calculate how much vertical space needs to be added in order to enable the column to fit into a box of size `\@colht` using the best information we have about the amount of shrink available (another thing which is known internally about a box, but cannot be accessed at the TEX level!).

This needs TEX3 otherwise `\pageshrink` is zero anyway; it may not be exactly the figure we wish as it is the total available from the all the material collected before the page-break decision is made. It will, we think, always be an overestimate of the actual shrink in the box; therefore this should always force the shortest possible column with the possibility of an overfull box.

This should work for both flush- and ragged-bottom setting since it makes the contents no smaller than the size (`\@colht`) of the box into which they are put.

Their should perhaps be an upper limit, of 0pt?, on the extra space added to force shrinking.

See above for a discussion of the `\boxmaxdepth` setting here.

```
    \advance \@tempdima -\ht\@outputbox
    \advance \@tempdima \pageshrink
⟨*trace⟩
    \fl@trace {Natural ht of col: \the \ht\@outputbox}%
    \fl@trace {\string \@colht: \the \@colht}%
    \fl@trace {Pageshrink added: \the \pageshrink}%
    \fl@trace {Hence, space added: \the \@tempdima}%
⟨/trace⟩
    \setbox\@outputbox \vbox to \@colht {%
%       \boxmaxdepth \maxdepth
      \unvbox\@outputbox
      \vskip \@tempdima
      \@textbottom
      }%
```

For the unstarred version, the final size of the page is precisely specified. Therefore, at least for the flush-bottom case, we need to ensure that, visually, it has this size exactly.

Thus we calculate this size and set the material in a box of this size, which is then put into a box of size `\@colht` with `\vss` at the bottom.

```
  \else
    \advance \@tempdima -\ht\@kludgeins
⟨*trace⟩
    \fl@trace {Natural ht of col: \the \ht\@outputbox}%
    \fl@trace {\string \@colht: \the \@colht}%
    \fl@trace {Extra size added: -\the \ht \@kludgeins}%
    \fl@trace {Hence, height of inner box: \the \@tempdima}%
    \fl@trace {Max? pageshrink available: \the \pageshrink}%
⟨/trace⟩
```

This type of final packaging could be done always; this may simplify all of this page-makeup.

It is not necessary to set `\boxmaxdepth` here since the `\@outputbox` ends with glue.

```
    \setbox \@outputbox \vbox to \@colht {%
      \vbox to \@tempdima {%
```

```
          \unvbox\@outputbox
          \@textbottom}%
        \vss}%
    \fi
```

Finally we need to explicitly make the insert box void.

```
    {\setbox \@tempboxa \box \@kludgeins}%
⟨*trace⟩
      \fl@trace {kludgeins box made void}%
⟨/trace⟩
}
⟨/2ekernel | fltrace⟩
```

(*End of definition for* `\@makespecialcolbox`.)

`\@texttop`
`\@textbottom`

These do nothing as a default.

```
⟨*2ekernel⟩
\let \@texttop \relax
\let \@textbottom \relax
```

(*End of definition for* `\@texttop` *and* `\@textbottom`.)

`\@resetactivechars`
`\@activechar@info`

RmS 93/09/06: added hook to protect against certain active characters in the output routine. Default checks are for active space and end-of-line.

```
\def\@activechar@info #1{%
      \@latex@info@no@line {Active #1 character found while
                            output routine is active
                            \MessageBreak
                            This may be a bug in a package file
                            you are using}%
}
```

Do not put any spaces in this next bit!

```
\begingroup
\obeylines\obeyspaces%
\catcode`\'\active%
\gdef\@resetactivechars{%
\def^^M{\@activechar@info{EOL}\space}%
\def {\@activechar@info{space}\space}%
\let'\active@math@prime}%
\endgroup
```

(*End of definition for* `\@resetactivechars` *and* `\@activechar@info`.)

`\@outputpage`
`\@shipoutsetup`
`\@writesetup`

The `\color@hbox` hooks here are used to avoid putting just a colour special into an otherwise empty box (in a header or footer). These boxes are often set to be completely empty and so adding a special produces a very underfull box message.

There has been extensive tidying up of the old code here; including the removal of a level of grouping.

The setting of `\protect` immediately before the `\shipout` is needed so that protected commands within `\write`s are handled correctly.

Within shipout's vbox it is reset to its default value, `\relax`.

Resetting it to its default value after the shipout has been completed (and the contents of the writes have been expanded) must be done by use of `\aftergroup`. This

is because it must have the value `\relax` before macros coming from other uses of `\aftergroup` within this box are expanded.

Putting this into the `\aftergroup` token list does not affect the definition used in expanding the `\write`s because the aftergroup token list is only constructed when popping the save-stack, it is not expanded until after the shipout is completed.

Question: should things from an `\aftergroup` within the shipped out box be executed in the environment set up for the writes, or after it finishes?

A lot of this code has been in-lined to prevent mis-use of internal commands as hooks.

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2017/04/15}%
⟨latexrelease⟩  {\@outputpage}{Reset language for hyphenation}%
⟨*2ekernel | latexrelease⟩
\def\@outputpage{%
```

The `\endgroup` is put in by `\aftergroup`.

```
\begingroup
```

Now all the set-up stuff has been in-lined for Frank.

First the stuff for the writes.

From here ... was in the command `\@writesetup`.

```
 \let \protect \noexpand
```

RmS 93/08/19: Redefined accents to allow changes in font encoding; but exactly why was this needed?

Reset `\language` to the value current at `\begin{document}`. In particular this ensures that a pagebreak in `verbatim` does not prevent hyphenation in the page head.

```
 \language\document@default@language
```

The `\catcode`\ = 10` was removed as it was considered useless (presumably because nothing gets tokenized during shipout).

This was put in as some error produced active spaces in a mark, I think.

Why was the hyphen reset?

```
 \@resetactivechars
```

If a page break happens between the start of a list and its first item the `@newlist` will be true and this will mess up any list that is used in the header or footer of the page. So we have to reset that flag.

```
 \global\let\@@if@newlist\if@newlist
 \global\@newlistfalse
```

This next hook replaces the following:

```
 \let\-\@dischyph
 \let\'\@acci\let\`\@accii\let\=\@acciii
 \let\\\@normalcr
 \let\par\@@par %% 15 Sep 87 (this was once inside the box)
```

and it does more than they did; in particular it sets:

```
\parindent\z@
\parskip\z@skip
\everypar{}%
\leftskip\z@skip
\rightskip\z@skip
\parfillskip\@flushglue
\lineskip\normallineskip
\baselineskip\normalbaselineskip
\sloppy
```

```
\@parboxrestore
```

... to here was in the command `\@writesetup`.

```
\shipout \vbox{%
  \set@typeset@protect
  \aftergroup \endgroup
```

Correct? or just restore by ending the group?

```
  \aftergroup \set@typeset@protect
```

This first bit has been moved inside the shipped out box.

Now the setup inside the shipped out box; this should contain all the stuff that could only affect typesetting; other stuff may need to be reset for the writes also.

From here ... was in the command `\@shipoutsetup`.

```
\if@specialpage
  \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
\fi
\if@twoside
  \ifodd\count\z@ \let\@thehead\@oddhead \let\@thefoot\@oddfoot
       \let\@themargin\oddsidemargin
  \else \let\@thehead\@evenhead
     \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
  \fi
\fi
```

The rest was always inside the box.

RmS 91/08/15: added this line:

```
\reset@font
```

RmS 93/08/06 Added `\lineskiplimit=0pt` to guard against it being nonzero: e.g. by `\offinterlineskip` being in effect.

There are probably lots of other things that may need resetting.

```
\normalsize
```

Reset the space factors.

```
\normalsfcodes
```

Reset these here (previously reset separately for head and foot)

```
\let\label\@gobble
\let\index\@gobble
\let\glossary\@gobble
```

```
\baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
```

... to here was in the command \@shipoutsetup.

```
    \@begindvi
    \vskip \topmargin
    \moveright\@themargin \vbox {%
      \setbox\@tempboxa \vbox to\headheight{%
        \vfil
        \color@hbox
          \normalcolor
          \hb@xt@\textwidth{\@thehead}%
        \color@endbox
```

22 Feb 87

```
        }%
      \dp\@tempboxa \z@
      \box\@tempboxa
      \vskip \headsep
      \box\@outputbox
      \baselineskip \footskip
      \color@hbox
        \normalcolor
        \hb@xt@\textwidth{\@thefoot}%
      \color@endbox
      }%
    }%
```

\endgroup now inserted by \aftergroup

Restore \if@newlist

```
  \global\let\if@newlist\@@if@newlist
```

```
  \global \@colht \textheight
  \stepcounter{page}%
```

It is now clear that this does something useful, thanks to Piet van Oostrum. It is needed because a float page is made without using TeX's page-builder; thus the output routine is never called so the marks are not updated.

```
  \let\firstmark\botmark
}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\@outputpage}{Reset language for hyphenation}%
⟨latexrelease⟩\def\@outputpage{%
⟨latexrelease⟩\begingroup
⟨latexrelease⟩  \let \protect \noexpand
⟨latexrelease⟩  \@resetactivechars
⟨latexrelease⟩  \global\let\@@if@newlist\if@newlist
⟨latexrelease⟩  \global\@newlistfalse
⟨latexrelease⟩  \@parboxrestore
⟨latexrelease⟩  \shipout \vbox{%
⟨latexrelease⟩    \set@typeset@protect
⟨latexrelease⟩    \aftergroup \endgroup
⟨latexrelease⟩    \aftergroup \set@typeset@protect
⟨latexrelease⟩  \if@specialpage
⟨latexrelease⟩    \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
⟨latexrelease⟩  \fi
```

```
⟨latexrelease⟩  \if@twoside
⟨latexrelease⟩    \ifodd\count\z@
⟨latexrelease⟩         \let\@thehead\@oddhead \let\@thefoot\@oddfoot
⟨latexrelease⟩         \let\@themargin\oddsidemargin
⟨latexrelease⟩    \else \let\@thehead\@evenhead
⟨latexrelease⟩       \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
⟨latexrelease⟩    \fi
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \reset@font
⟨latexrelease⟩  \normalsize
⟨latexrelease⟩  \normalsfcodes
⟨latexrelease⟩  \let\label\@gobble
⟨latexrelease⟩  \let\index\@gobble
⟨latexrelease⟩  \let\glossary\@gobble
⟨latexrelease⟩  \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
⟨latexrelease⟩    \@begindvi
⟨latexrelease⟩    \vskip \topmargin
⟨latexrelease⟩    \moveright\@themargin \vbox {%
⟨latexrelease⟩      \setbox\@tempboxa \vbox to\headheight{%
⟨latexrelease⟩        \vfil
⟨latexrelease⟩        \color@hbox
⟨latexrelease⟩          \normalcolor
⟨latexrelease⟩          \hb@xt@\textwidth{\@thehead}%
⟨latexrelease⟩        \color@endbox
⟨latexrelease⟩        }%
⟨latexrelease⟩      \dp\@tempboxa \z@
⟨latexrelease⟩      \box\@tempboxa
⟨latexrelease⟩      \vskip \headsep
⟨latexrelease⟩      \box\@outputbox
⟨latexrelease⟩      \baselineskip \footskip
⟨latexrelease⟩      \color@hbox
⟨latexrelease⟩        \normalcolor
⟨latexrelease⟩        \hb@xt@\textwidth{\@thefoot}%
⟨latexrelease⟩      \color@endbox
⟨latexrelease⟩      }%
⟨latexrelease⟩    }%
⟨latexrelease⟩  \global\let\if@newlist\@@if@newlist
⟨latexrelease⟩  \global \@colht \textheight
⟨latexrelease⟩  \stepcounter{page}%
⟨latexrelease⟩  \let\firstmark\botmark
⟨latexrelease⟩}
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@outputpage`, `\@shipoutsetup`, *and* `\@writesetup`.)

`\@begindvi` This unboxes stuff that must appear before anything else in the `.dvi` file, then returns that box register to the free list and cancels itself.

The stuff in the box should not add any typeset material to the page.

```
\def \@begindvi{%
  \unvbox \@begindvibox
  \global\let \@begindvi \@empty
}
```

(*End of definition for* `\@begindvi`.)

\@combinefloats
\@cflt
\@cflb

The \boxmaxdepth setting here was not made local to a box so was dangerous. It is needed only within the box made by \@cflt (and not normally even there), so it has been moved there; this also agrees with the original pseudocode.

```
\def \@combinefloats {%
%    \boxmaxdepth \maxdepth
    \ifx \@toplist\@empty \else \@cflt \fi
    \ifx \@botlist\@empty \else \@cflb \fi
}
```

```
\def \@cflt{%
    \let \@elt \@comflelt
    \setbox\@tempboxa \vbox{}%
    \@toplist
    \setbox\@outputbox \vbox{%
                              \boxmaxdepth \maxdepth
                              \unvbox\@tempboxa
                              \vskip -\floatsep
                              \topfigrule
                              \vskip \textfloatsep
                              \unvbox\@outputbox
                              }%
    \let\@elt\relax
    \xdef\@freelist{\@freelist\@toplist}%
    \global\let\@toplist\@empty
}
```

```
\def \@cflb {%
    \let\@elt\@comflelt
    \setbox\@tempboxa \vbox{}%
    \@botlist
    \setbox\@outputbox \vbox{%
                              \unvbox\@outputbox
                              \vskip \textfloatsep
                              \botfigrule
                              \unvbox\@tempboxa
                              \vskip -\floatsep
                              }%
    \let\@elt\relax
    \xdef\@freelist{\@freelist\@botlist}%
    \global \let \@botlist\@empty
}
```

(*End of definition for* \@combinefloats*,* \@cflt*, and* \@cflb*.*)

\@comflelt
\@comdblflelt
\@combinedblfloats

```
\def\@comflelt#1{\setbox\@tempboxa
      \vbox{\unvbox\@tempboxa\box #1\vskip\floatsep}}
```

```
\def\@comdblflelt#1{\setbox\@tempboxa
      \vbox{\unvbox\@tempboxa\box #1\vskip\dblfloatsep}}
```

```
\def \@combinedblfloats{%
  \ifx \@dbltoplist \@empty
  \else
    \setbox\@tempboxa \vbox{}%
    \let \@elt \@comdblflelt
```

```
    \@dbltoplist
    \let \@elt \relax
    \xdef \@freelist {\@freelist\@dbltoplist}%
    \global\let \@dbltoplist \@empty
    \setbox\@outputbox \vbox to\textheight
```

The setting of `\boxmaxdepth` here has no effect since the `\@outputbox` should already have depth zero. Even so, it would have no effect on the layout of the page.

```
      {%\boxmaxdepth\maxdepth   %% probably not needed, CAR
       \unvbox\@tempboxa\vskip-\dblfloatsep
```

Here we need different typesetting if the top float comes from `\@topnewpage`.

```
       \ifnum \@dbltopnum>\m@ne
         \dblfigrule
       \fi
       \vskip \dbltextfloatsep
```

If pdf links are present in the galley and those links get broken across pages they have to end up being on the same level of boxing (even if not actually in the same structure) due to some engine restrictions in pdfTEX and LuaTEX. We therefore unbox `\@outputbox` here (which only contains a single `\hbox`) so that this case has the same boxing level as a normal twocolumn page without top floats.

```
       \unvbox\@outputbox
       }%
  \fi
}
⟨/2ekernel⟩
```

(*End of definition for* `\@comflelt`, `\@comdblflelt`, *and* `\@combinedblfloats`.)

`\@startcolumn`
`\@startdblcolumn`

We could combine (most of) these two into `\@startcol <list>`. Note that `\@xstartcol` was only used once (i.e. in `\@startcolumn`); it has therefore been removed. This is not quite as efficient but it now has the same structure as `\@startdblcolumn`.

The empty-list test has been moved to `\@tryfcolumn`.

```
⟨*2ekernel | fltrace⟩
\def \@startcolumn {%
  \global \@colroom \@colht
  \@tryfcolumn \@deferlist
  \if@fcolmade
⟨*trace⟩
    \fl@trace{PAGE: float \if@twocolumn column \else page \fi
                  completed}%
⟨/trace⟩
  \else

    \begingroup
      \let \reserved@b \@deferlist
      \global \let \@deferlist \@empty
      \let \@elt \@scolelt
      \reserved@b
    \endgroup
  \fi
}
```

This one does not need to set `\@colht`.

```
⟨/2ekernel | fltrace⟩
⟨latexrelease | fltrace⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease | fltrace⟩  {\@startdblcolumn}{float order in 2-column}%
⟨*2ekernel | latexrelease | fltrace⟩
\def \@startdblcolumn {%
  \@tryfcolumn \@deferlist
  \if@fcolmade
⟨fltrace⟩    \fl@trace{PAGE: double float page completed}%
  \else
    \begingroup
      \let \reserved@b \@deferlist
      \global \let \@deferlist \@empty
      \let \@elt \@sdblcolelt
      \reserved@b
    \endgroup
  \fi
}%
⟨/2ekernel | latexrelease | fltrace⟩
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨latexrelease | fltrace⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace⟩  {\@startdblcolumn}{float order in 2-column}%
⟨latexrelease | fltrace⟩\def \@startdblcolumn {%
```

Not needed since this always comes after `\@outputpage`:

```
⟨latexrelease | fltrace⟩% \global \@colht \textheight
⟨latexrelease | fltrace⟩  \@tryfcolumn \@dbldeferlist
⟨latexrelease | fltrace⟩  \if@fcolmade
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{PAGE: double float page completed}%
⟨/trace⟩
⟨latexrelease | fltrace⟩  \else

⟨latexrelease | fltrace⟩    \begingroup
⟨latexrelease | fltrace⟩      \let \reserved@b \@dbldeferlist
⟨latexrelease | fltrace⟩      \global \let \@dbldeferlist \@empty
⟨latexrelease | fltrace⟩      \let \@elt \@sdblcolelt
⟨latexrelease | fltrace⟩      \reserved@b
⟨latexrelease | fltrace⟩    \endgroup
⟨latexrelease | fltrace⟩  \fi
⟨latexrelease | fltrace⟩}%
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨*2ekernel | fltrace⟩
```

(*End of definition for* `\@startcolumn` *and* `\@startdblcolumn`.)

`\@tryfcolumn` Now tests if its list is empty before any further exertion.

```
\def \@tryfcolumn #1{%
  \global \@fcolmadefalse
  \ifx #1\@empty
  \else
⟨*trace⟩
      \fl@trace{PAGE: try float \if@twocolumn column/page\else page\fi
                  ---\string #1}%
```

```
      \fl@trace{----- \string #1: #1}%
⟨/trace⟩
    \xdef\@trylist{#1}%
    \global \let \@failedlist \@empty
    \begingroup
      \let \@elt \@xtryfc \@trylist
    \endgroup
    \if@fcolmade
      \@vtryfc #1%
    \fi
  \fi
}
⟨/2ekernel | fltrace⟩
```

(*End of definition for* `\@tryfcolumn`.)

```
⟨*2ekernel⟩
```

`\@scolelt`

```
\def\@scolelt#1{\def\@currbox{#1}\@addtonextcol}
```

(*End of definition for* `\@scolelt`.)

`\@sdblcolelt`

```
\def\@sdblcolelt#1{\def\@currbox{#1}\@addtodblcol}
```

(*End of definition for* `\@sdblcolelt`.)

`\@vtryfc`

```
\def\@vtryfc #1{%
  \global\setbox\@outputbox\vbox{}%
  \let\@elt\@wtryfc
  \@flsucceed
  \global\setbox\@outputbox \vbox to\@colht{%
    \vskip \@fptop
    \vskip -\@fpsep
    \unvbox \@outputbox
    \vskip \@fpbot}%
  \let\@elt\relax
  \xdef #1{\@failedlist\@flfail}%
  \xdef\@freelist{\@freelist\@flsucceed}}
```

(*End of definition for* `\@vtryfc`.)

`\@wtryfc`

```
\def\@wtryfc #1{%
  \global\setbox\@outputbox\vbox{%
    \unvbox\@outputbox
    \vskip\@fpsep
    \box #1}}
```

(*End of definition for* `\@wtryfc`.)

\@xtryfc

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{\@xtryfc}%
⟨latexrelease⟩                              {float order in 2-column}%
⟨*2ekernel | latexrelease⟩
\def\@xtryfc #1{%
  \@next\reserved@a\@trylist{}{}%
  \@currtype \count #1%
  \divide\@currtype\@xxxii
  \multiply\@currtype\@xxxii
  \@bitor \@currtype \@failedlist
  \@testfp #1%
  \@testwrongwidth #1%
  \ifdim \ht #1>\@colht
     \@testtrue
  \fi
  \if@test
    \@cons\@failedlist #1%
  \else
    \@ytryfc #1%
  \fi}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@xtryfc}%
⟨latexrelease⟩                              {float order in 2-column}%
⟨latexrelease⟩\def\@xtryfc #1{%
⟨latexrelease⟩  \@next\reserved@a\@trylist{}{}%
⟨latexrelease⟩  \@currtype \count #1%
⟨latexrelease⟩  \divide\@currtype\@xxxii
⟨latexrelease⟩  \multiply\@currtype\@xxxii
⟨latexrelease⟩  \@bitor \@currtype \@failedlist
⟨latexrelease⟩  \@testfp #1%
⟨latexrelease⟩  \ifdim \ht #1>\@colht
⟨latexrelease⟩    \@testtrue
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \if@test
⟨latexrelease⟩    \@cons\@failedlist #1%
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@ytryfc #1%
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\@xtryfc`.)

\@ytryfc

```
\def\@ytryfc #1{%
  \begingroup
    \gdef\@flsucceed{\@elt #1}%
    \global\let\@flfail\@empty
    \@tempdima\ht #1%
    \let\@elt\@ztryfc
    \@trylist
```

```
    \ifdim \@tempdima >\@fpmin
      \global\@fcolmadetrue
    \else
      \@cons\@failedlist #1%
    \fi
  \endgroup
  \if@fcolmade
    \let\@elt\@gobble
  \fi}
```

(*End of definition for* `\@ytryfc`.)

`\@ztryfc`

```
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}{@ztryfc}%
⟨latexrelease⟩                              {float order in 2-column}%
⟨*2ekernel | latexrelease⟩
\def\@ztryfc #1{%
  \@tempcnta\count #1%
  \divide\@tempcnta\@xxxii
  \multiply\@tempcnta\@xxxii
  \@bitor \@tempcnta {\@failedlist \@flfail}%
  \@testfp #1%
```

not in fixfloats?

```
  \@testwrongwidth #1%
  \@tempdimb\@tempdima
  \advance\@tempdimb\ht #1%
  \advance\@tempdimb\@fpsep
  \ifdim \@tempdimb >\@colht
    \@testtrue
  \fi
  \if@test
    \@cons\@flfail #1%
  \else
    \@cons\@flsucceed #1%
    \@tempdima\@tempdimb
  \fi}%
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{@ztryfc}%
⟨latexrelease⟩                              {float order in 2-column}%
⟨latexrelease⟩\def\@ztryfc #1{%
⟨latexrelease⟩  \@tempcnta \count#1%
⟨latexrelease⟩  \divide\@tempcnta\@xxxii
⟨latexrelease⟩  \multiply\@tempcnta\@xxxii
⟨latexrelease⟩  \@bitor \@tempcnta {\@failedlist \@flfail}%
⟨latexrelease⟩  \@testfp #1%
⟨latexrelease⟩  \@tempdimb\@tempdima
⟨latexrelease⟩  \advance\@tempdimb \ht#1%
⟨latexrelease⟩  \advance\@tempdimb\@fpsep
⟨latexrelease⟩  \ifdim \@tempdimb >\@colht
⟨latexrelease⟩    \@testtrue
⟨latexrelease⟩  \fi
```

```
⟨latexrelease⟩  \if@test
⟨latexrelease⟩    \@cons\@flfail #1%
⟨latexrelease⟩  \else
⟨latexrelease⟩    \@cons\@flsucceed #1%
⟨latexrelease⟩    \@tempdima\@tempdimb
⟨latexrelease⟩  \fi}%
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\@ztryfc`.)

The major changes for float suppression and the changes to the float mechanism to make it conform to the documentation are in these next macros.

`\@addtobot` Lots of changes.

```
⟨*2ekernel | fltrace⟩
\def \@addtobot {%
⟨*trace⟩
   \fl@trace{***Start addtobot}%
⟨/trace⟩
   \@getfpsbit 4\relax
⟨*trace⟩
   \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi bot:
                                                        \the \@fpstype}%
⟨/trace⟩
   \ifodd \@tempcnta
     \@flsetnum \@botnum
     \ifnum \@botnum>\z@
       \@tempswafalse
       \@flcheckspace \@botroom \@botlist
       \if@tempswa
```

This next line means that this page is produced with box 255 having depth zero, rather than the normal maxdepth: is this needed, useful?

```
         \global \maxdepth \z@
         \@flupdates \@botnum \@botroom \@botlist
⟨*trace⟩
         \fl@trace{colroom (after-bot) = \the \@colroom}%
         \fl@trace{colnum (after-bot) = \the \@colnum}%
         \fl@trace{botnum (after-bot) = \the \@botnum}%
         \fl@trace{***Success: bot}%
⟨/trace⟩
         \@inserttrue
       \fi
⟨*trace⟩
     \else
       \fl@trace{Fail: botnum = \the \@botnum:
                                 fpstype \the \@fpstype=ORD?}%
       \ifnum \@fpstype<\sixt@@n
         \fl@trace{ERROR: !b float not successful (addtobot)}%
       \fi
⟨/trace⟩
     \fi
   \fi
}
```

(*End of definition for* `\@addtobot`.)

\@addtotoporbot Lots of changes.

```
\def \@addtotoporbot {%
⟨*trace⟩
   \fl@trace{***Start addtotoporbot}%
⟨/trace⟩
   \@getfpsbit \tw@
⟨*trace⟩
   \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi top:
                                                        \the \@fpstype}%
⟨/trace⟩
   \ifodd \@tempcnta
     \@flsetnum \@topnum
     \ifnum \@topnum>\z@
       \@tempswafalse
       \@flcheckspace \@toproom \@toplist
       \if@tempswa
         \@bitor\@currtype{\@midlist\@botlist}%
⟨*trace⟩
           \fl@trace{(mid+bot)list: \@midlist, \@botlist:
                              (addtotoporbot-before)}%
⟨/trace⟩
         \if@test
⟨*trace⟩
         \fl@trace{type already on list: mid or bot---sent to addtobot}%
⟨/trace⟩
         \else
          \@flupdates \@topnum \@toproom \@toplist
⟨*trace⟩
          \fl@trace{colroom (after-top) = \the \@colroom}%
          \fl@trace{colnum (after-top) = \the \@colnum}%
          \fl@trace{topnum (after-top) = \the \@topnum}%
          \fl@trace{***Success: top}%
⟨/trace⟩
          \@inserttrue
         \fi
       \fi
⟨*trace⟩
     \else
       \fl@trace{Fail: topnum = \the \@topnum: fpstype
                                          \the \@fpstype=ORD?}%
       \ifnum \@fpstype<\sixt@@n
         \fl@trace{ERROR: !t float not successful (addtotoporbot)}%
       \fi
⟨/trace⟩
     \fi
   \fi
   \if@insert
   \else
⟨*trace⟩
     \fl@trace{sent to addtobot (addtotoporbot)}%
⟨/trace⟩
     \@addtobot
   \fi
}
```

```
⟨/2ekernel | fltrace⟩
```

(*End of definition for* `\@addtotoporbot`.)

`\@addtocurcol` Lots of changes.

```
⟨latexrelease | fltrace | flafter⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease | fltrace | flafter⟩  {\@addtocurcol}{float order in 2-column}%
⟨*2ekernel | latexrelease | fltrace | flafter⟩
\def \@addtocurcol {%
⟨*trace⟩
  \fl@trace{***Start addtocurcol}%
⟨/trace⟩
   \@insertfalse
   \@setfloattypecounts
   \ifnum \@fpstype=8
⟨*trace⟩
     \fl@trace{fpstype !p only (addtocurcol): \the \@fpstype = 8?}%
⟨/trace⟩
   \else
     \ifnum \@fpstype=24
⟨*trace⟩
       \fl@trace{fpstype p only (addtocurcol): \the \@fpstype = 24?}%
⟨/trace⟩
     \else
       \@flsettextmin
```

This is a new adjustment which is quite a major change in functionality; but it implements the documentation. Note that `\@reqcolroom` will include the whole of the page-so-far, and hence includes `\@textfloatsheight` of floats, so before comparing it with `\@textmin`, we add this to `\@textmin` also.

```
⟨*trace⟩
       \fl@trace{textfloatsheight (before) = \the \@textfloatsheight}%
⟨/trace⟩
       \advance \@textmin \@textfloatsheight
       \@reqcolroom \@pageht
```

This line must be removed since `\@specialoutput` changed.

```
%       \advance \@reqcolroom \@pagedp
⟨*trace⟩
       \fl@trace{textmin + textfloatsheight: \the \@textmin}%
       \fl@trace{page-so-far: \the \@reqcolroom}%
⟨/trace⟩
       \ifdim \@textmin>\@reqcolroom
         \@reqcolroom \@textmin
⟨*trace⟩
         \fl@trace{ORD? textmin being used}%
⟨/trace⟩
       \fi
       \advance \@reqcolroom \ht\@currbox
⟨*trace⟩
       \fl@trace{float size = \the \ht \@currbox (addtocurcol)}%
       \fl@trace{colroom = \the \@colroom (addtocurcol)}%
       \fl@trace{reqcolroom = \the \@reqcolroom (addtocurcol)}%
⟨/trace⟩
```

```
        \ifdim \@colroom>\@reqcolroom
          \@flsetnum \@colnum
          \ifnum \@colnum>\z@
            \@bitor\@currtype\@deferlist
```

We need to defer the float also if its width doesn't fit.

```
           \@testwrongwidth\@currbox
⟨*trace⟩
            \fl@trace{deferlist: \@deferlist: (addtocurcol-before)}%
⟨/trace⟩
            \if@test
⟨*trace⟩
              \fl@trace{type already on list: defer (addtocurcol)}%
⟨/trace⟩
            \else
              \@bitor\@currtype\@botlist
⟨*trace⟩
            \fl@trace{botlist: \@botlist: (addtocurcol-before)}%
⟨/trace⟩
              \if@test
⟨*trace⟩
                \fl@trace{type already on list: bot---sent to addtobot}%
⟨/trace⟩
                \@addtobot
              \else
⟨*trace⟩
                \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi
                       here: \the \@fpstype}%
⟨/trace⟩
                \ifodd \count\@currbox
                  \advance \@reqcolroom \intextsep
                  \ifdim \@colroom>\@reqcolroom
                    \global \advance \@colnum \m@ne
                    \global \advance \@textfloatsheight \ht\@currbox
```

This may sometimes give an overestimate.

```
                    \global \advance \@textfloatsheight 2\intextsep
                    \@cons \@midlist \@currbox
⟨*trace⟩
                  \fl@trace{***Success: here}%
                  \fl@trace{textfloatsheight (after-here) =
                       \the \@textfloatsheight}%
                  \fl@trace{colnum (after-here) = \the \@colnum}%
⟨/trace⟩
```

CHANGE TO `\@addtocurcol`:

`\penalty\z@` changed to `\penalty\interlinepenalty` so `\samepage` works properly with figure and table environments. (Changed 23 Oct 86)

There is also an `\addpenalty\interlinepenalty` above.

Although it is best to use `\addvspace` in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of `\parskip` here in case the float is added between paragraphs (i.e. when in vertical mode).

If the nobreak switch is true we need to reset it and clear `\everypar` since the float may not reset the flag and cannot reset the `\everypar` globally.

Typesetting starts here (we are in vertical mode).

```
                    \if@nobreak
                      \nobreak
                      \@nobreakfalse
                      \everypar{}%
                    \else
                      \addpenalty \interlinepenalty
                    \fi
                    \vskip \intextsep
                    \box\@currbox
                    \penalty\interlinepenalty
                    \vskip\intextsep
                    \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi
```

Typesetting ends here.

```
                    \outputpenalty \z@
                    \@inserttrue
⟨*trace⟩
                  \else
                    \fl@trace{Fail---no room at 2nd test of colroom
                                  (addtocorcol \string\intextsep)}%
⟨/trace⟩
                  \fi
                \fi
                \if@insert
                \else
```

Next set of docstrip guards are a bit weird, essentially `\@addtotoporbot` ends up inside the kernel and the `fltrace` package and `\@addtobot` shows up in the `flafter` package. Guess that could have been done a bit more obvious :-)

```
⟨*2ekernel | fltrace | latexrelease⟩
⟨*trace⟩
                    \fl@trace{not here: sent to addtotoporbot}%
⟨/trace⟩
                    \@addtotoporbot
⟨/2ekernel | fltrace | latexrelease⟩
⟨*!2ekernel&!fltrace&!latexrelease⟩
⟨*trace⟩
                    \fl@trace{not here: sent to addtobot}%
⟨/trace⟩
                    \@addtobot
⟨/!2ekernel&!fltrace&!latexrelease⟩
                \fi
              \fi
            \fi
⟨*trace⟩
          \else
            \fl@trace{Fail: colnum = \the \@colnum:
                          fpstype \the \@fpstype=ORD?}%
            \ifnum \@fpstype<\sixt@@n
              \fl@trace{ERROR: BANG float not successful (addtocurcol)}%
            \fi
```

```
⟨/trace⟩
          \fi
⟨*trace⟩
        \else
          \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
                                                    (addtocurcol)}%
⟨/trace⟩
        \fi
      \fi
    \fi
    \if@insert
    \else
      \@resethfps
⟨*trace⟩
      \fl@trace{put on deferlist (addtocurcol)}%
⟨/trace⟩
      \@cons\@deferlist\@currbox
⟨*trace⟩
      \fl@trace{deferlist: \@deferlist: (addtocurcol-after)}%
⟨/trace⟩
    \fi
}%
⟨/2ekernel | latexrelease | fltrace | flafter⟩
⟨latexrelease | fltrace | flafter⟩\EndIncludeInRelease
⟨latexrelease | fltrace | flafter⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace | flafter⟩  {\@addtocurcol}{float order in 2-column}%
⟨latexrelease | fltrace | flafter⟩\def \@addtocurcol {%
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩  \fl@trace{***Start addtocurcol}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩   \@insertfalse
⟨latexrelease | fltrace | flafter⟩   \@setfloattypecounts
⟨latexrelease | fltrace | flafter⟩   \ifnum \@fpstype=8
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩     \fl@trace{fpstype !p only (addtocurcol):
⟨latexrelease | fltrace | flafter⟩                            \the \@fpstype = 8?}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩   \else
⟨latexrelease | fltrace | flafter⟩     \ifnum \@fpstype=24
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩      \fl@trace{fpstype p only (addtocurcol):
⟨latexrelease | fltrace | flafter⟩                           \the \@fpstype = 24?}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩     \else
⟨latexrelease | fltrace | flafter⟩       \@flsettextmin
```

This is a new adjustment which is quite a major change in functionality; but it implements the documentation. Note that `\@reqcolroom` will include the whole of the page-so-far, and hence includes `\@textfloatsheight` of floats, so before comparing it with `\@textmin`, we add this to `\@textmin` also.

```
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩       \fl@trace{textfloatsheight (before) =
⟨latexrelease | fltrace | flafter⟩                        \the \@textfloatsheight}%
⟨/trace⟩
```

```
⟨latexrelease | fltrace | flafter⟩        \advance \@textmin \@textfloatsheight
⟨latexrelease | fltrace | flafter⟩        \@reqcolroom \@pageht
```

This line must be removed since `\@specialoutput` changed.

```
%        \advance \@reqcolroom \@pagedp
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩        \fl@trace{textmin + textfloatsheight:
⟨latexrelease | fltrace | flafter⟩                                 \the \@textmin}%
⟨latexrelease | fltrace | flafter⟩        \fl@trace{page-so-far: \the \@reqcolroom}%
⟨latexrelease | fltrace | flafter⟩
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩        \ifdim \@textmin>\@reqcolroom
⟨latexrelease | fltrace | flafter⟩          \@reqcolroom \@textmin
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩          \fl@trace{ORD? textmin being used}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩        \fi
⟨latexrelease | fltrace | flafter⟩        \advance \@reqcolroom \ht\@currbox
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩        \fl@trace{float size =
⟨latexrelease | fltrace | flafter⟩                \the \ht \@currbox (addtocurcol)}%
⟨latexrelease | fltrace | flafter⟩        \fl@trace{colroom =
⟨latexrelease | fltrace | flafter⟩                    \the \@colroom (addtocurcol)}%
⟨latexrelease | fltrace | flafter⟩        \fl@trace{reqcolroom =
⟨latexrelease | fltrace | flafter⟩                 \the \@reqcolroom (addtocurcol)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩        \ifdim \@colroom>\@reqcolroom
⟨latexrelease | fltrace | flafter⟩          \@flsetnum \@colnum
⟨latexrelease | fltrace | flafter⟩          \ifnum \@colnum>\z@
⟨latexrelease | fltrace | flafter⟩            \@bitor\@currtype\@deferlist
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩            \fl@trace{deferlist:
⟨latexrelease | fltrace | flafter⟩               \@deferlist: (addtocurcol-before)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩            \if@test
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩              \fl@trace{type already on list:
⟨latexrelease | fltrace | flafter⟩                             defer (addtocurcol)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩            \else
⟨latexrelease | fltrace | flafter⟩              \@bitor\@currtype\@botlist
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩            \fl@trace{botlist: \@botlist:
⟨latexrelease | fltrace | flafter⟩                            (addtocurcol-before)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩              \if@test
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩                \fl@trace{type already on list:
⟨latexrelease | fltrace | flafter⟩                          bot---sent to addtobot}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩                \@addtobot
⟨latexrelease | fltrace | flafter⟩              \else
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩                \fl@trace{fpstype
⟨latexrelease | fltrace | flafter⟩                \ifodd \@tempcnta OK \else not \fi
```

```
⟨latexrelease | fltrace | flafter⟩                here: \the \@fpstype}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩                \ifodd \count\@currbox
⟨latexrelease | fltrace | flafter⟩                  \advance \@reqcolroom \intextsep
⟨latexrelease | fltrace | flafter⟩                  \ifdim \@colroom>\@reqcolroom
⟨latexrelease | fltrace | flafter⟩                   \global \advance \@colnum \m@ne
⟨latexrelease | fltrace | flafter⟩                   \global \advance
⟨latexrelease | fltrace | flafter⟩                    \@textfloatsheight\ht\@currbox
```

This may sometimes give an overestimate.

```
⟨latexrelease | fltrace | flafter⟩                   \global \advance
⟨latexrelease | fltrace | flafter⟩                    \@textfloatsheight 2\intextsep
⟨latexrelease | fltrace | flafter⟩                    \@cons \@midlist \@currbox
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩                  \fl@trace{***Success: here}%
⟨latexrelease | fltrace | flafter⟩                  \fl@trace{textfloatsheight
⟨latexrelease | fltrace | flafter⟩                       (after-here) =
⟨latexrelease | fltrace | flafter⟩                       \the \@textfloatsheight}%
⟨latexrelease | fltrace | flafter⟩                  \fl@trace{colnum (after-here) =
⟨latexrelease | fltrace | flafter⟩                           \the \@colnum}%
⟨/trace⟩
```

CHANGE TO `\@addtocurcol`:

`\penalty\z@` changed to `\penalty\interlinepenalty` so `\samepage` works properly with figure and table environments. (Changed 23 Oct 86)

There is also an `\addpenalty\interlinepenalty` above.

Although it is best to use `\addvspace` in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of `\parskip` here in case the float is added between paragraphs (i.e. when in vertical mode).

If the nobreak switch is true we need to reset it and clear `\everypar` since the float may not reset the flag and cannot reset the `\everypar` globally.

Typesetting starts here (we are in vertical mode).

```
⟨latexrelease | fltrace | flafter⟩                  \if@nobreak
⟨latexrelease | fltrace | flafter⟩                    \nobreak
⟨latexrelease | fltrace | flafter⟩                    \@nobreakfalse
⟨latexrelease | fltrace | flafter⟩                    \everypar{}%
⟨latexrelease | fltrace | flafter⟩                  \else
⟨latexrelease | fltrace | flafter⟩                    \addpenalty\interlinepenalty
⟨latexrelease | fltrace | flafter⟩                  \fi
⟨latexrelease | fltrace | flafter⟩                  \vskip \intextsep
⟨latexrelease | fltrace | flafter⟩                  \box\@currbox
⟨latexrelease | fltrace | flafter⟩                  \penalty\interlinepenalty
⟨latexrelease | fltrace | flafter⟩                  \vskip\intextsep
⟨latexrelease | fltrace | flafter⟩                  \ifnum\outputpenalty
⟨latexrelease | fltrace | flafter⟩                              <-\@Mii \vskip
⟨latexrelease | fltrace | flafter⟩                       -\parskip\fi
```

Typesetting ends here.

```
⟨latexrelease | fltrace | flafter⟩                  \outputpenalty \z@
⟨latexrelease | fltrace | flafter⟩                  \@inserttrue
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩                \else
```

```
⟨latexrelease | fltrace | flafter⟩  \fl@trace{Fail---no room at 2nd test of colroom
⟨latexrelease | fltrace | flafter⟩                (addtocorcol \string\intextsep)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩                 \fi
⟨latexrelease | fltrace | flafter⟩               \fi
⟨latexrelease | fltrace | flafter⟩               \if@insert
⟨latexrelease | fltrace | flafter⟩               \else
```

Next set of docstrip guards are a bit weird, essentially `\@addtotoporbot` ends up inside the kernel and the `fltrace` package and `\@addtotoporbot` shows up in the `flafter` package. Guess that could have been done a bit more obvious :-)

```
⟨*2ekernel | fltrace⟩
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩      \fl@trace{not here: sent to addtotoporbot}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩                 \@addtotoporbot
⟨/2ekernel | fltrace⟩
⟨*!2ekernel&!autoload&!fltrace⟩
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩       \fl@trace{not here: sent to addtobot}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩                 \@addtobot
⟨/!2ekernel&!autoload&!fltrace⟩
⟨latexrelease | fltrace | flafter⟩               \fi
⟨latexrelease | fltrace | flafter⟩             \fi
⟨latexrelease | fltrace | flafter⟩           \fi
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩         \else
⟨latexrelease | fltrace | flafter⟩         \fl@trace{Fail: colnum = \the \@colnum:
⟨latexrelease | fltrace | flafter⟩                   fpstype \the \@fpstype=ORD?}%
⟨latexrelease | fltrace | flafter⟩         \ifnum \@fpstype<\sixt@@n
⟨latexrelease | fltrace | flafter⟩  \fl@trace{ERROR: BANG float not successful
⟨latexrelease | fltrace | flafter⟩                                (addtocurcol)}%
⟨latexrelease | fltrace | flafter⟩           \fi
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩         \fi
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩       \else
⟨latexrelease | fltrace | flafter⟩         \fl@trace{Fail---no room: fl box ht:
⟨latexrelease | fltrace | flafter⟩               \the \ht \@currbox (addtocurcol)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩       \fi
⟨latexrelease | fltrace | flafter⟩     \fi
⟨latexrelease | fltrace | flafter⟩   \fi
⟨latexrelease | fltrace | flafter⟩   \if@insert
⟨latexrelease | fltrace | flafter⟩   \else
⟨latexrelease | fltrace | flafter⟩     \@resethfps
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩     \fl@trace{put on deferlist (addtocurcol)}%
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩     \@cons\@deferlist\@currbox
⟨*trace⟩
⟨latexrelease | fltrace | flafter⟩     \fl@trace{deferlist: \@deferlist:
⟨latexrelease | fltrace | flafter⟩                         (addtocurcol-after)}%
```

```
⟨/trace⟩
⟨latexrelease | fltrace | flafter⟩    \fi
⟨latexrelease | fltrace | flafter⟩   }%
⟨latexrelease | fltrace | flafter⟩\EndIncludeInRelease
```

(*End of definition for* `\@addtocurcol`.)

`\@addtonextcol` Lots of changes.

```
⟨latexrelease | fltrace⟩\IncludeInRelease{2015/01/01}
⟨latexrelease | fltrace⟩  {\@addtonextcol}{float order in 2-column}%
⟨*2ekernel | latexrelease | fltrace⟩
\def\@addtonextcol{%
  \begingroup
⟨*trace⟩
   \fl@trace{***Start addtonextcol}%
⟨/trace⟩
   \@insertfalse
   \@setfloattypecounts
   \ifnum \@fpstype=8
⟨*trace⟩
     \fl@trace{fpstype not curcol: \the \@fpstype = 8?}%
⟨/trace⟩
   \else
     \ifnum \@fpstype=24
⟨*trace⟩
       \fl@trace{fpstype not curcol: \the \@fpstype = 24?}%
⟨/trace⟩
     \else
       \@flsettextmin
⟨*trace⟩
       \fl@trace{text-so-far: 0pt (top of col)}%
⟨/trace⟩
       \@reqcolroom \ht\@currbox
⟨*trace⟩
       \fl@trace{float size: \the \@reqcolroom (addtonextcol)}%
⟨/trace⟩
       \advance \@reqcolroom \@textmin
⟨*trace⟩
       \fl@trace{colroom = \the \@colroom (addtonextcol)}%
       \fl@trace{reqcolroom = \the \@reqcolroom (addtonextcol)}%
⟨/trace⟩
       \ifdim \@colroom>\@reqcolroom
         \@flsetnum \@colnum
         \ifnum\@colnum>\z@
            \@bitor\@currtype\@deferlist
⟨*trace⟩
            \fl@trace{deferlist: \@deferlist: (addtonextcol-before)}%
⟨/trace⟩
            \@testwrongwidth\@currbox
            \if@test
⟨*trace⟩
              \fl@trace{type already on list: defer (addtonextcol)}%
⟨/trace⟩
```

```
            \else
⟨*trace⟩
              \fl@trace{sent to addtotoporbot (addtonextcol)}%
⟨/trace⟩
              \@addtotoporbot
            \fi
          \fi
⟨*trace⟩
        \else
          \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
                                                  (addtonextcol)}%
⟨/trace⟩
        \fi
      \fi
   \fi
   \if@insert
   \else
⟨*trace⟩
      \fl@trace{put back on deferlist (addtonextcol)}%
⟨/trace⟩
      \@cons\@deferlist\@currbox
⟨*trace⟩
      \fl@trace{deferlist: \@deferlist: (addtonextcol-after)}%
⟨/trace⟩
   \fi
⟨*trace⟩
  \fl@trace{End of addtonextcol -- locally counts:}%
  \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
⟨/trace⟩
  \endgroup
⟨*trace⟩
 \fl@trace{End of addtonextcol -- globally counts:}%
 \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
⟨/trace⟩
}%
⟨/2ekernel | latexrelease | fltrace⟩
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨latexrelease | fltrace⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace⟩  {\@addtonextcol}{float order in 2-column}%
⟨latexrelease | fltrace⟩\def\@addtonextcol{%
⟨latexrelease | fltrace⟩  \begingroup
⟨*trace⟩
⟨latexrelease | fltrace⟩   \fl@trace{***Start addtonextcol}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \@insertfalse
⟨latexrelease | fltrace⟩   \@setfloattypecounts
⟨latexrelease | fltrace⟩   \ifnum \@fpstype=8
⟨*trace⟩
⟨latexrelease | fltrace⟩     \fl@trace{fpstype not curcol:
⟨latexrelease | fltrace⟩                     \the \@fpstype = 8?}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \else
⟨latexrelease | fltrace⟩     \ifnum \@fpstype=24
⟨*trace⟩
```

```
⟨latexrelease | fltrace⟩        \fl@trace{fpstype not curcol:
⟨latexrelease | fltrace⟩                        \the \@fpstype = 24?}%
⟨/trace⟩
⟨latexrelease | fltrace⟩      \else
⟨latexrelease | fltrace⟩        \@flsettextmin
⟨*trace⟩
⟨latexrelease | fltrace⟩        \fl@trace{text-so-far: 0pt (top of col)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩        \@reqcolroom \ht\@currbox
⟨*trace⟩
⟨latexrelease | fltrace⟩        \fl@trace{float size:
⟨latexrelease | fltrace⟩                \the \@reqcolroom (addtonextcol)}%
⟨latexrelease | fltrace⟩
⟨/trace⟩
⟨latexrelease | fltrace⟩        \advance \@reqcolroom \@textmin
⟨*trace⟩
⟨latexrelease | fltrace⟩        \fl@trace{colroom =
⟨latexrelease | fltrace⟩                  \the \@colroom (addtonextcol)}%
⟨latexrelease | fltrace⟩        \fl@trace{reqcolroom =
⟨latexrelease | fltrace⟩                \the \@reqcolroom (addtonextcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩        \ifdim \@colroom>\@reqcolroom
⟨latexrelease | fltrace⟩          \@flsetnum \@colnum
⟨latexrelease | fltrace⟩          \ifnum\@colnum>\z@
⟨latexrelease | fltrace⟩             \@bitor\@currtype\@deferlist
⟨*trace⟩
⟨latexrelease | fltrace⟩             \fl@trace{deferlist: \@deferlist:
⟨latexrelease | fltrace⟩                          (addtonextcol-before)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩             \if@test
⟨*trace⟩
⟨latexrelease | fltrace⟩               \fl@trace{type already on list:
⟨latexrelease | fltrace⟩                            defer (addtonextcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩             \else
⟨*trace⟩
⟨latexrelease | fltrace⟩               \fl@trace{sent to addtotoporbot
⟨latexrelease | fltrace⟩                                  (addtonextcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩               \@addtotoporbot
⟨latexrelease | fltrace⟩             \fi
⟨latexrelease | fltrace⟩          \fi
⟨*trace⟩
⟨latexrelease | fltrace⟩        \else
⟨latexrelease | fltrace⟩          \fl@trace{Fail---no room: fl box ht:
⟨latexrelease | fltrace⟩               \the \ht \@currbox (addtonextcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩        \fi
⟨latexrelease | fltrace⟩      \fi
⟨latexrelease | fltrace⟩    \fi
⟨latexrelease | fltrace⟩    \if@insert
⟨latexrelease | fltrace⟩    \else
⟨*trace⟩
⟨latexrelease | fltrace⟩      \fl@trace{put back on deferlist
```

```
⟨latexrelease | fltrace⟩                                  (addtonextcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩      \@cons\@deferlist\@currbox
⟨*trace⟩
⟨latexrelease | fltrace⟩      \fl@trace{deferlist: \@deferlist:
⟨latexrelease | fltrace⟩                            (addtonextcol-after)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩    \fi
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{End of addtonextcol --
⟨latexrelease | fltrace⟩                              locally counts:}%
⟨latexrelease | fltrace⟩    \fl@trace{col: \the \@colnum.
⟨latexrelease | fltrace⟩       top: \the \@topnum. bot: \the \@botnum.}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \endgroup
⟨*trace⟩
⟨latexrelease | fltrace⟩   \fl@trace{End of addtonextcol --
⟨latexrelease | fltrace⟩                            globally counts:}%
⟨latexrelease | fltrace⟩   \fl@trace{col: \the \@colnum.
⟨latexrelease | fltrace⟩        top: \the \@topnum. bot: \the \@botnum.}%
⟨/trace⟩
⟨latexrelease | fltrace⟩}%
⟨latexrelease | fltrace⟩\EndIncludeInRelease
```

(*End of definition for* `\@addtonextcol`.)

`\@addtodblcol` Lots of changes.

```
⟨latexrelease | fltrace⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease | fltrace⟩   {\@addtodblcol}{float order in 2-column}%
⟨*2ekernel | latexrelease | fltrace⟩
\def\@addtodblcol{%
  \begingroup
⟨*trace⟩
  \fl@trace{***Start addtodblcol}%
⟨/trace⟩
   \@insertfalse
   \@setfloattypecounts
   \@getfpsbit \tw@
⟨*trace⟩
   \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi dbltop:
                                                        \the \@fpstype}%
⟨/trace⟩
   \ifodd\@tempcnta
     \@flsetnum \@dbltopnum
     \ifnum \@dbltopnum>\z@
       \@tempswafalse
       \ifdim \@dbltoproom>\ht\@currbox
         \@tempswatrue
⟨*trace⟩
         \fl@trace{Space OK: \@dbltoproom =
                \the \@dbltoproom > \the \ht \@currbox
                                          (dbltoproom)}%
⟨/trace⟩
       \else
```

```
⟨*trace⟩
          \fl@trace{fpstype: \the \@fpstype (addtodblcol)}%
⟨/trace⟩
          \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
            \fl@trace{BANG float ignoring \@dbltoproom}%
            \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
                            Ht float: \the \ht \@currbox-BANG}%
⟨/trace⟩
```

Need to check that there is room on the page, using the local value of `\@textmin` to make the necessary adjustment to `\@dbltoproom`.

```
            \advance \@dbltoproom \@textmin
⟨*trace⟩
            \fl@trace{Local value of texmin: \the\@textmin}%
            \fl@trace{\@spaces space on page = \the \@dbltoproom.
                            Ht float: \the \ht \@currbox-BANG}%
⟨/trace⟩
            \ifdim \@dbltoproom>\ht\@currbox
              \@tempswatrue
⟨*trace⟩
              \fl@trace{Space OK BANG: space on page =
                          \the \@dbltoproom > \the \ht \@currbox}%
            \else
              \fl@trace{fpstype: \the \@fpstype}%
              \fl@trace{Fail---no room dbltoproom-BANG?:}%
              \fl@trace{\@spaces space on page = \the \@dbltoproom.
                            Ht float: \the \ht \@currbox}%
⟨/trace⟩
            \fi
            \advance \@dbltoproom -\@textmin
⟨*trace⟩
          \else
            \fl@trace{fpstype: \the \@fpstype}%
            \fl@trace{Fail---no room dbltoproom-ORD?:}%
            \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
                            Ht float: \the \ht \@currbox}%
⟨/trace⟩
          \fi
        \fi
        \if@tempswa
            \@bitor \@currtype \@deferlist
⟨*trace⟩
            \fl@trace{(dbl)deferlist: \@deferlist: (before)}%
⟨/trace⟩
```

not in fixfloats?

```
          \@testwrongwidth\@currbox
            \if@test
⟨*trace⟩
               \fl@trace{type already on list: (dbl)defer}%
⟨/trace⟩
            \else
               \@tempdima -\ht\@currbox
```

```
              \advance\@tempdima
                -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
                                          \dblfloatsep \fi
              \global \advance \@dbltoproom \@tempdima
              \global \advance \@colht \@tempdima
              \global \advance \@dbltopnum \m@ne
              \@cons \@dbltoplist \@currbox
⟨*trace⟩
              \fl@trace{dbltopnum (after) = \the \@dbltopnum}%
              \fl@trace{***Success: dbltop}%
⟨/trace⟩
              \@inserttrue
           \fi
       \fi
⟨*trace⟩
     \else
       \fl@trace{Fail: dbltopnum = \the \@dbltopnum: fpstype
                                                 \the \@fpstype=ORD?}%
       \ifnum \@fpstype<\sixt@@n
         \fl@trace{ERROR: !t float not successful (addtodblcol)}%
       \fi
⟨/trace⟩
     \fi
   \fi
   \if@insert
   \else
⟨*trace⟩
     \fl@trace{put on deferlist}%
⟨/trace⟩
     \@cons\@deferlist\@currbox
⟨*trace⟩
     \fl@trace{(dbl)deferlist: \@deferlist: (after)}%
⟨/trace⟩
   \fi
⟨*trace⟩
   \fl@trace{End of addtodblcol -- locally count:}%
   \fl@trace{ dbltop: \the \@dbltopnum.}%
⟨/trace⟩
  \endgroup
⟨*trace⟩
  \fl@trace{End of addtodblcol -- globally count:}%
  \fl@trace{dbltop: \the \@dbltopnum.}%
⟨/trace⟩
}%
⟨/2ekernel | latexrelease | fltrace⟩
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨latexrelease | fltrace⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace⟩  {\@addtodblcol}{float order in 2-column}%
⟨latexrelease | fltrace⟩\def\@addtodblcol{%
⟨latexrelease | fltrace⟩  \begingroup
⟨*trace⟩
⟨latexrelease | fltrace⟩  \fl@trace{***Start addtodblcol}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \@insertfalse
```

```
⟨latexrelease | fltrace⟩    \@setfloattypecounts
⟨latexrelease | fltrace⟩    \@getfpsbit \tw@
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{fpstype \ifodd \@tempcnta OK
⟨latexrelease | fltrace⟩                \else not \fi dbltop: \the \@fpstype}%
⟨/trace⟩
⟨latexrelease | fltrace⟩    \ifodd\@tempcnta
⟨latexrelease | fltrace⟩      \@flsetnum \@dbltopnum
⟨latexrelease | fltrace⟩      \ifnum \@dbltopnum>\z@
⟨latexrelease | fltrace⟩        \@tempswafalse
⟨latexrelease | fltrace⟩        \ifdim \@dbltoproom>\ht\@currbox
⟨latexrelease | fltrace⟩          \@tempswatrue
⟨*trace⟩
⟨latexrelease | fltrace⟩          \fl@trace{Space OK: \@dbltoproom =
⟨latexrelease | fltrace⟩                \the \@dbltoproom > \the \ht \@currbox
⟨latexrelease | fltrace⟩                                        (dbltoproom)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩        \else
⟨*trace⟩
⟨latexrelease | fltrace⟩       \fl@trace{fpstype: \the \@fpstype (addtodblcol)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩          \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
⟨latexrelease | fltrace⟩            \fl@trace{BANG float ignoring \@dbltoproom}%
⟨latexrelease | fltrace⟩            \fl@trace{\@spaces \@dbltoproom =
⟨latexrelease | fltrace⟩                     \the \@dbltoproom.
⟨latexrelease | fltrace⟩                     Ht float: \the \ht \@currbox-BANG}%
⟨/trace⟩
```

Need to check that there is room on the page, using the local value of `\@textmin` to make the necessary adjustment to `\@dbltoproom`.

```
⟨latexrelease | fltrace⟩            \advance \@dbltoproom \@textmin
⟨*trace⟩
⟨latexrelease | fltrace⟩        \fl@trace{Local value of texmin: \the\@textmin}%
⟨latexrelease | fltrace⟩        \fl@trace{\@spaces space on page =
⟨latexrelease | fltrace⟩                    \the \@dbltoproom.
⟨latexrelease | fltrace⟩                     Ht float: \the \ht \@currbox-BANG}%
⟨/trace⟩
⟨latexrelease | fltrace⟩            \ifdim \@dbltoproom>\ht\@currbox
⟨latexrelease | fltrace⟩              \@tempswatrue
⟨*trace⟩
⟨latexrelease | fltrace⟩          \fl@trace{Space OK BANG: space on page =
⟨latexrelease | fltrace⟩                  \the\@dbltoproom > \the\ht\@currbox}%
⟨latexrelease | fltrace⟩            \else
⟨latexrelease | fltrace⟩          \fl@trace{fpstype: \the \@fpstype}%
⟨latexrelease | fltrace⟩          \fl@trace{Fail---no room dbltoproom-BANG?:}%
⟨latexrelease | fltrace⟩          \fl@trace{\@spaces space on page =
⟨latexrelease | fltrace⟩                      \the \@dbltoproom.
⟨latexrelease | fltrace⟩                       Ht float: \the \ht \@currbox}%
⟨/trace⟩
⟨latexrelease | fltrace⟩            \fi
⟨latexrelease | fltrace⟩            \advance \@dbltoproom -\@textmin
⟨*trace⟩
⟨latexrelease | fltrace⟩          \else
```

```
⟨latexrelease | fltrace⟩            \fl@trace{fpstype: \the \@fpstype}%
⟨latexrelease | fltrace⟩            \fl@trace{Fail---no room dbltoproom-ORD?:}%
⟨latexrelease | fltrace⟩            \fl@trace{\@spaces \@dbltoproom =
⟨latexrelease | fltrace⟩               \the \@dbltoproom.
⟨latexrelease | fltrace⟩               Ht float: \the \ht \@currbox}%
⟨/trace⟩
⟨latexrelease | fltrace⟩          \fi
⟨latexrelease | fltrace⟩        \fi
⟨latexrelease | fltrace⟩        \if@tempswa
⟨latexrelease | fltrace⟩            \@bitor \@currtype \@dbldeferlist
⟨*trace⟩
⟨latexrelease | fltrace⟩            \fl@trace{dbldeferlist:
⟨latexrelease | fltrace⟩                       \@dbldeferlist: (before)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩            \if@test
⟨*trace⟩
⟨latexrelease | fltrace⟩              \fl@trace{type already on list: dbldefer}%
⟨/trace⟩
⟨latexrelease | fltrace⟩            \else
⟨latexrelease | fltrace⟩               \@tempdima -\ht\@currbox
⟨latexrelease | fltrace⟩               \advance\@tempdima
⟨latexrelease | fltrace⟩                 -\ifx \@dbltoplist\@empty
⟨latexrelease | fltrace⟩                       \dbltextfloatsep
⟨latexrelease | fltrace⟩                  \else \dblfloatsep \fi
⟨latexrelease | fltrace⟩               \global \advance \@dbltoproom \@tempdima
⟨latexrelease | fltrace⟩               \global \advance \@colht \@tempdima
⟨latexrelease | fltrace⟩               \global \advance \@dbltopnum \m@ne
⟨latexrelease | fltrace⟩               \@cons \@dbltoplist \@currbox
⟨*trace⟩
⟨latexrelease | fltrace⟩               \fl@trace{dbltopnum (after) =
⟨latexrelease | fltrace⟩                                   \the \@dbltopnum}%
⟨latexrelease | fltrace⟩               \fl@trace{***Success: dbltop}%
⟨/trace⟩
⟨latexrelease | fltrace⟩               \@inserttrue
⟨latexrelease | fltrace⟩            \fi
⟨latexrelease | fltrace⟩        \fi
⟨*trace⟩
⟨latexrelease | fltrace⟩      \else
⟨latexrelease | fltrace⟩        \fl@trace{Fail: dbltopnum = \the \@dbltopnum:
⟨latexrelease | fltrace⟩                          fpstype \the \@fpstype=ORD?}%
⟨latexrelease | fltrace⟩        \ifnum \@fpstype<\sixt@@n
⟨latexrelease | fltrace⟩          \fl@trace{ERROR: !t float not successful
⟨latexrelease | fltrace⟩                                        (addtodblcol)}%
⟨latexrelease | fltrace⟩        \fi
⟨/trace⟩
⟨latexrelease | fltrace⟩      \fi
⟨latexrelease | fltrace⟩    \fi
⟨latexrelease | fltrace⟩    \if@insert
⟨latexrelease | fltrace⟩    \else
⟨*trace⟩
⟨latexrelease | fltrace⟩      \fl@trace{put on dbldeferlist}%
⟨/trace⟩
⟨latexrelease | fltrace⟩      \@cons\@dbldeferlist\@currbox
⟨*trace⟩
```

```
⟨latexrelease | fltrace⟩      \fl@trace{dbldeferlist: \@dbldeferlist: (after)}%
⟨/trace⟩
⟨latexrelease | fltrace⟩    \fi
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{End of addtodblcol -- locally count:}%
⟨latexrelease | fltrace⟩    \fl@trace{ dbltop: \the \@dbltopnum.}%
⟨/trace⟩
⟨latexrelease | fltrace⟩   \endgroup
⟨*trace⟩
⟨latexrelease | fltrace⟩   \fl@trace{End of addtodblcol -- globally count:}%
⟨latexrelease | fltrace⟩   \fl@trace{dbltop: \the \@dbltopnum.}%
⟨/trace⟩
⟨latexrelease | fltrace⟩}%
⟨latexrelease | fltrace⟩\EndIncludeInRelease
```

(*End of definition for* `\@addtodblcol`.)

`\@addmarginpar`

```
⟨*2ekernel⟩
\def\@addmarginpar{\@next\@marbox\@currlist{\@cons\@freelist\@marbox
    \@cons\@freelist\@currbox}\@latexbug\@tempcnta\@ne
    \if@twocolumn
        \if@firstcolumn \@tempcnta\m@ne \fi
    \else
      \if@mparswitch
         \ifodd\c@page \else\@tempcnta\m@ne \fi
      \fi
      \if@reversemargin \@tempcnta -\@tempcnta \fi
    \fi
    \ifnum\@tempcnta <\z@  \global\setbox\@marbox\box\@currbox \fi
    \@tempdima\@mparbottom
    \advance\@tempdima -\@pageht
    \advance\@tempdima\ht\@marbox
    \ifdim\@tempdima >\z@
      \@latex@warning@no@line {Marginpar on page \thepage\space moved}%
    \else
      \@tempdima\z@
    \fi
    \global\@mparbottom\@pageht
    \global\advance\@mparbottom\@tempdima
    \global\advance\@mparbottom\dp\@marbox
    \global\advance\@mparbottom\marginparpush
    \advance\@tempdima -\ht\@marbox
```

Putting box movement inside the ‘marbox’:

```
    \global\setbox \@marbox
                   \vbox {\vskip \@tempdima
                          \box \@marbox}%
    \global \ht\@marbox \z@
    \global \dp\@marbox \z@
```

Sticking (rather than gluing:-) the ‘marbox’ to the line above, changed vskip to kern:

```
    \kern -\@pagedp
    \nointerlineskip
    \hb@xt@\columnwidth
```

```
      {\ifnum \@tempcnta >\z@
          \hskip\columnwidth \hskip\marginparsep
       \else
          \hskip -\marginparsep \hskip -\marginparwidth
       \fi
       \box\@marbox \hss}%
```

For this reason the following code can vanish:

```
\nobreak              %% No longer needed.  CAR92/12
\vskip -\@tempdima    %% No longer needed.  CAR92/12
```

```
    \nointerlineskip
    \hbox{\vrule \@height\z@ \@width\z@ \@depth\@pagedp}}
```

(*End of definition for* `\@addmarginpar`.)

1.1.1 Kludgeins

This part of the file is part of the implementation of the following two new commands for LaTeX2e.

`\enlargethispage{<dim>}`

Adds `<dim>` to the height of the current column only. On the printed page the bottom of this column is extended downwards by exactly `<dim>` without having any effect on the placement of the footer; this may result in an overprinting.

`\enlargethispage*{<dim>}`

Similar to `\enlargethispage` but it tries to squeeze the column to be printed in as small a space as possible, ie it uses any shrinkability in the column. If the column was not explicitly broken (e.g. with `\pagebreak`) this may result in an overfull box message but except for this it will come out as expected (if you know what to expect).

The star form of this command is dedicated to Leslie Lamport, the other we need for ourselves (FMi, CAR).

These commands may well have unwanted effects if used soon before a `\clearpage`: please give keep them clear of such places.

`\@kludgeins` The insert which makes TeX do a lot of the necessary work. All we need to put into it is the amount by which the pagegoal should be changed.

```
\newinsert \@kludgeins
\global\dimen\@kludgeins \maxdimen
\global\count\@kludgeins 1000
```

(*End of definition for* `\@kludgeins`.)

`\enlargethispage` `\enlargethispage*` The user command.

```
\gdef \enlargethispage {%
   \@ifstar
      {%
⟨*trace⟩
       \fl@trace{Enlarging page height * }%
⟨/trace⟩
       \@enlargepage{\hbox{\kern\p@}}}%
```

```
     {%
⟨*trace⟩
      \fl@trace{Enlarging page height exactly---}%
⟨/trace⟩
      \@enlargepage\@empty}%
}
```

(*End of definition for* `\enlargethispage` *and* `\enlargethispage*`.)

`\@enlargepage` This actually inserts the insert, after checking for extreme values of the change.

```
\gdef\@enlargepage#1#2{%
⟨*trace⟩
   \fl@trace{\@spaces\@spaces by #2}%
⟨/trace⟩
   \@tempskipa#2\relax
   \ifdim \@tempskipa>.5\maxdimen
     \@latex@error{Suggested\space extra\space height\space
                   (\the\@tempskipa)\space dangerously\space
                   large}\@eha
   \else
     \ifdim \vsize<.5\maxdimen
⟨*trace⟩
       \fl@trace {Kludgeins added--pagegoal before: \the\pagegoal}%
⟨/trace⟩
       \@bsphack
         \insert\@kludgeins{#1\vskip-\@tempskipa}%
       \@esphack
```

This next bit is for tracing only:

```
⟨*trace⟩
       \ifvmode \par
         \fl@trace {Kludgeins added--pagegoal after: \the \pagegoal}%
       \fi
⟨/trace⟩
     \else
       \@latex@error{Page\space height\space already\space
                     too\space large}\@eha
     \fi
   \fi
}
```

(*End of definition for* `\@enlargepage`.)

`\ShowFloat` This command provides some information about the contenta of a float register. Float registers have internal names of the form `\bx@`⟨*Uppercase-letter(s)-or numbers*⟩ and you specify just this letter or letters as the argument, e.g., `\ShowFloat{A}`. (There is not much error recovery if you specify something that isn't a float.)

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
⟨latexrelease⟩                 {\ShowFloat}{Show float register contents}%
\def\ShowFloat#1{\begingroup
    \let \fl@trace \fl@tracemessage
    \fl@trace{***Float #1 details:}%
    \ifcsname bx@#1\endcsname
```

```
      \expandafter\fl@ShowFloat\csname bx@#1\endcsname
    \else
      \fl@trace{Not a float!}%
    \fi
  \endgroup
}
\def\fl@ShowFloat#1{%
  \fl@traceval{\count#1}%  % this here should be interpreted on day
  \fl@traceval{\ht#1}%
  \fl@traceval{\dp#1}%
  \fl@traceval{\wd#1}%
  {\tracingonline1\showboxbreadth10\showboxdepth3\showbox#1}%
}
```

Here are two definitions from `fltrace` that make the above code work:

```
\def \fl@traceval #1{\fl@trace{\string #1 = \the #1}}
\def \fl@tracemessage #1{{\let\@elt\@empty\typeout{LaTeX2e: #1}}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                 {\ShowFloat}{Show float register contents}%
⟨latexrelease⟩
⟨latexrelease⟩\let\ShowFloat\@undefined
⟨latexrelease⟩\let\fl@ShowFloat\@undefined
⟨latexrelease⟩\let\fl@traceval\@undefined
⟨latexrelease⟩\let\fl@tracemessage\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

(*End of definition for* `\ShowFloat`.)

1.1.2 Float control

This part implements controllable floats and other changes to the float mechanism.

It provides, at the document level, the following command for inclusion in LaTeX2e.

`\suppressfloats`

This suppresses all further floats on the current page.

With an optional argument it suppresses only floats only in certain positions on the current page.

`[t]` suppresses only floats at the top of the page `[b]` suppresses only floats at the bottom of the page

It also enables the use of an extra specifier, `!`, in the location optional argument of a float. If this is present then, just for this particular float, whenever it is processed by the float mechanism the following are ignored:

- all restrictions on the number of floats which can appear;
- all explicit restrictions on the amount of space which should (not) be occupied by floats and/or text.

The mechanism will still attempt to ensure that pages are not overfull.

These specifiers override, for the single float, the suppression commands described above.

In its current form, it also supplies a reasonably exhaustive, and somewhat baroque, means of tracing some aspects of the float mechanism.

More tracing.

`\fl@trace` `\tracefloatsoff` `\tracefloats` `\fl@traceval` `\tracefloatvals` `\fl@tracemessage`

Set-up tracing for floats independent of other tracing as it produces mega-output. Default is no tracing.

```
⟨*fltrace⟩
\def \fl@tracemessage #1{{\let\@elt\@empty\typeout{LaTeX2e: #1}}}
\def \tracefloats{\let \fl@trace \fl@tracemessage}
\def \tracefloatsoff {\let \fl@trace \@gobble}
\tracefloatsoff
\def \fl@traceval #1{\fl@trace{\string #1 = \the #1}}
\IncludeInRelease{2015/01/01}{\tracefloatvals}%
                          {trace float vals}%
\def \tracefloatvals{%
```

As `\@dblfloatplacement` sets `\f@depth` it needs to be run inside a group, otherwise the float placement will test for the wrong value.[60]

```
 \begingroup
```

When the user requests `\tracefloatvals` then they should show regardless of the tracing state, so locally we make sure that it is activated.

```
  \tracefloats
  \@dblfloatplacement
  \@floatplacement
  \fl@trace{***Float placement parameters:}%
  \fl@traceval\@colnum
  \fl@traceval\@colroom
  \fl@traceval\@topnum
  \fl@traceval\@toproom
  \fl@traceval\@botnum
  \fl@traceval\@botroom
  \fl@traceval\@fpmin
  \fl@trace{\string\textfraction = \textfraction}%
  \fl@traceval\@dbltopnum
  \fl@traceval\@dbltoproom
  \fl@trace{\string\textfraction = \textfraction}%
  \fl@trace{toplist: \@toplist}%
  \fl@trace{botlist: \@botlist}%
  \fl@trace{midlist: \@midlist}%
  \fl@trace{deferlist: \@deferlist}%
  \fl@trace{dbltoplist: \@dbltoplist}%
%FMi  \fl@trace{dbldeferlist: \@dbldeferlist}%
 \endgroup
}
\EndIncludeInRelease
\IncludeInRelease{0000/00/00}{\tracefloatvals}%
                          {trace float vals}%
\def \tracefloatvals{%
```

[60] This is a somewhat questionable design.

```
 \begingroup
  \tracefloats
  \@dblfloatplacement
  \@floatplacement
  \fl@trace{***Float placement parameters:}%
  \fl@traceval\@colnum
  \fl@traceval\@colroom
  \fl@traceval\@topnum
  \fl@traceval\@toproom
  \fl@traceval\@botnum
  \fl@traceval\@botroom
  \fl@traceval\@fpmin
  \fl@trace{\string\textfraction = \textfraction}%
  \fl@traceval\@dbltopnum
  \fl@traceval\@dbltoproom
  \fl@trace{\string\textfraction = \textfraction}%
  \fl@trace{toplist: \@toplist}%
  \fl@trace{botlist: \@botlist}%
  \fl@trace{midlist: \@midlist}%
  \fl@trace{deferlist: \@deferlist}%
  \fl@trace{dbltoplist: \@dbltoplist}%
% next line only in old releases
  \fl@trace{dbldeferlist: \@dbldeferlist}%
 \endgroup
}
\EndIncludeInRelease
```

We need to make sure that `fltrace` comes before `flafter` to make the tracing work.

```
\@ifpackageloaded{flafter}
 {\PackageWarningNoLine
      {fltrace}{Load 'fltrace' before 'flafter'\MessageBreak
                Attempting to recover by reloading 'flafter'}%
```

Hide the fact that `flafter` was already loaded and then request it anew.

```
    \expandafter\let\csname ver@flafter.sty\endcsname\relax
    \def\reserved@a#1{%
      \expandafter\let\csname\string#1+flafter+IIR\endcsname\relax}%
    \reserved@a\@addtocurcol
    \reserved@a\@addtonextcol
    \RequirePackage{flafter}}{}
⟨/fltrace⟩
```

As the code for `flafter` will contain tracing calls so that it works in conjunction with `fltrace` we need to provide a dummy definition for `\fl@trace` in that package.

```
⟨*flafter⟩
\providecommand\fl@trace[1]{}
⟨/flafter⟩
```

(*End of definition for* `\fl@trace` *and others.*)

`\suppressfloats` `\@flstop` Float suppression commands: these set the relevant counter globally to zero. Thus they are overridden for a particular float by an ! specifier.

```
⟨*2ekernel⟩
\def \suppressfloats {%
   \@ifnextchar [%
```

```
      \@flstop
     {\global \@colnum \z@}%
}
```

Maybe this should be a loop over #1?

```
\def \@flstop [#1]{%
   \if t#1%
     \global \@topnum \z@
   \fi
   \if b#1%
     \global \@botnum \z@
   \fi
}
```

(*End of definition for* `\suppressfloats` *and* `\@flstop`.)

Manipulation of float placement and type; both their strings and the corresponding count registers.

`\@fpstype` `\@reqcolroom` `\@textfloatsheight`

First a new count register to go with `\@currtype`.

Then a new skip register, for information needed to remove the `\@maxsep` conservatism: it is possible that this could use a temporary register.

Finally a dimension register to hold the total height of in-text floats on the current page. This is needed to implement a major change in the functionality of `\@addtocurcol` which is, nevertheless, a bug fix. It is not local and therefore cannot be a temporary register.

```
\newcount \@fpstype
\newdimen \@reqcolroom
\newdimen \@textfloatsheight
⟨/2ekernel⟩
```

(*End of definition for* `\@fpstype`, `\@reqcolroom`, *and* `\@textfloatsheight`.)

`\@fpsadddefault`

Adds the default placement to what is already there.

Should not need to change this, but could do it as follows:

```
def \@fpsadddefault {%
  \@temptokena \expandafter\expandafter\expandafter
               {\csname fps@\@captype \endcsname}%
  \edef \reserved@a {\the\@temptokena}%
  \@onelevel@sanitize \reserved@a
  \edef \@fps {\@fps\reserved@a}%
```

```
⟨*2ekernel | fltrace⟩
\def \@fpsadddefault {%
⟨*trace⟩
   \fl@trace{fps changed from: \@fps}%
⟨/trace⟩
   \edef \@fps {\@fps\csname fps@\@captype \endcsname}%
   \@latex@warning {%
     No positions in optional float specifier.\MessageBreak
     Default added (so using ‘\@fps’)}%
}
```

(*End of definition for* `\@fpsadddefault`.)

\@setfloattypecounts Sets counters \@fpstype and \@currtype.

BANG == bit4 of \count\@currbox = 0.

```
\def \@setfloattypecounts {%
  \@currtype \count\@currbox
  \@fpstype \count\@currbox
  \divide\@currtype\@xxxii \multiply\@currtype\@xxxii
  \advance \@fpstype -\@currtype
⟨*trace⟩
  \fl@trace{(mod 32) fpstype: \the \@fpstype}%
  \fl@trace{(mult of 32) currtype: \the \@currtype}%
% Tracing only: but some should be changed into real errors/warnings?
  \ifnum \@fpstype<\sixt@@n
    \ifnum \@fpstype=\z@
      \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 0?}%
    \fi
    \ifnum \@fpstype=\@ne
      \fl@trace{WARNING: only h, fpstype = \the \@fpstype = 1?}%
    \fi
    \fl@trace{BANG float}%
  \else
    \ifnum \@fpstype=\sixt@@n
      \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 16?}%
    \fi
    \ifnum \@fpstype=17
      \fl@trace{WARNING: only h, fpstype = \the \@fpstype = 17?}%
    \fi
    \fl@trace{ORD float}%
  \fi
⟨/trace⟩
}
⟨/2ekernel | fltrace⟩
```

(*End of definition for* \@setfloattypecounts.)

Macros for getting, testing and setting bits of the fps.

\@getfpsbit Sets \@tempcnta to required bit of \count\@currbox.

```
⟨*2ekernel⟩
\def \@getfpsbit {%
   \@boxfpsbit \@currbox
}
```

(*End of definition for* \@getfpsbit.)

\@boxfpsbit Used above.

```
\def \@boxfpsbit #1#2{%
   \@tempcnta \count#1%
   \divide \@tempcnta #2\relax
}
```

(*End of definition for* \@boxfpsbit.)

\@testfp New definition of the float page test.

```
\def \@testfp #1{%
   \@boxfpsbit #18\relax % Really ‘#1 8’ for human readers!
```

```
   \ifodd \@tempcnta
   \else
     \@testtrue
   \fi
}
```

(*End of definition for* `\@testfp`.)

`\@setfpsbit` Sets required bit of `\@tempcnta` (to 1).

```
\def \@setfpsbit #1{%
   \@tempcntb \@tempcnta
   \divide \@tempcntb #1\relax
   \ifodd \@tempcntb
   \else
     \advance \@tempcnta #1\relax
   \fi
}
⟨/2ekernel⟩
```

(*End of definition for* `\@setfpsbit`.)

`\@resethfps` Globally adds t as a possible location for an h or !h only placement: this must be done using the count.

Although it will leave `\@fpstype` set to 17 even if it was originally 1, this does not matter since it is the last thing in `\@addtocurcol`.

```
⟨*2ekernel | fltrace⟩
\def \@resethfps {%
   \let\reserved@a\@empty
   \ifnum \@fpstype=\@ne
      \def \reserved@a {!}%
      \@fpstype 17
   \fi
   \ifnum \@fpstype=17
     \global \advance \count\@currbox \tw@
     \@latex@warning@no@line {%
       `\reserved@a h' float specifier changed to `\reserved@a ht'}%
⟨*trace⟩
     \fl@trace{%
        `t' added to `\reserved@a h'- new Count: \the \count\@currbox}%
⟨/trace⟩
   \fi
}
```

(*End of definition for* `\@resethfps`.)

Special stuff for BANG floats.

`\@flsetnum` Ignores any zero float counter value in case BANG.

It uses a local assignment to the normally global counter: a bit naughty, perhaps?

These assignments are safe so long as the counter involved is only consulted once (i.e. only for the 'bang float') with the changed value. This is the case within `\@addtocurcol` because it is used only once within a call of the output routine (which forms a group).

For `\@addtonextcol` this is achieved by putting a group around its code; this is needed because it is called (by `\@startcolumn`) for each float which was on the deferlist.

Almost identical considerations pertain to \@addtodblcol. There may be more efficient ways to handle this, but the group seems to be the simplest.

```
\def \@flsetnum #1{%
⟨*trace⟩
   \fl@trace{fpstype: \the \@fpstype (flsetnum \string#1)}%
⟨/trace⟩
   \ifnum \@fpstype<\sixt@@n
     \ifnum #1=\z@
⟨*trace⟩
       \fl@trace{BANG float resetting \string#1 to 1}%
⟨/trace⟩
       #1\@ne
     \fi
   \fi
⟨*trace⟩
   \fl@trace{#1 (before) = \the #1}%
⟨/trace⟩
}
```

(*End of definition for* \@flsetnum.)

\@flsettextmin This ignores \textfraction space restriction in case BANG.

```
\def \@flsettextmin {%
⟨*trace⟩
   \fl@trace{fpstype: \the \@fpstype (flsettextmin)}%
⟨/trace⟩
   \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
     \fl@trace{BANG ignoring textmin}%
⟨/trace⟩
     \@textmin \z@
   \else
     \@textmin \textfraction\@colht
⟨*trace⟩
     \fl@trace{ORD textmin = \the \@textmin}%
⟨/trace⟩
   \fi
}
```

(*End of definition for* \@flsettextmin.)

\@flcheckspace This ignores space restriction in case BANG; this is still slightly conservative since it does not allow for the fact that, if there is no text in the column then \textfloatsep is not needed. Sets @tempswa true if there is room for \@currbox.

```
\def \@flcheckspace #1#2{%
   \advance \@reqcolroom
     \ifx #2\@empty \textfloatsep \else \floatsep \fi
⟨*trace⟩
   \fl@trace{colroom = \the \@colroom
                                   (flcheckspace \string#1 \string#2)}%
   \fl@trace{reqcolroom = \the \@reqcolroom
                                   (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
```

```
   \ifdim \@colroom>\@reqcolroom
     \ifdim #1>\ht\@currbox
       \@tempswatrue
⟨*trace⟩
       \fl@trace{Space OK: #1 = \the #1 > \the \ht \@currbox
                                   (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
     \else
⟨*trace⟩
       \fl@trace{fpstype: \the \@fpstype
                                   (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
       \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
         \fl@trace{BANG float ignoring #1
                                   (flcheckspace \string#1 \string#2):}%
         \fl@trace{\@spaces #1 = \the #1.  Ht float: \the \ht \@currbox
                                                          BANG}%
⟨/trace⟩
         \@tempswatrue
⟨*trace⟩
       \else
         \fl@trace{Fail---no room (flcheckspace \string#1 \string#2)
                       (fpstype \the \@fpstype=ORD?):}%
         \fl@trace{\@spaces #1 = \the #1.  Ht float: \the \ht \@currbox
                                                          ORD?}%
⟨/trace⟩
       \fi
     \fi
⟨*trace⟩
   \else
     \fl@trace{Fail---no room at 2nd test of colroom
                   (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
   \fi
}
⟨/2ekernel | fltrace⟩
```

(*End of definition for* `\@flcheckspace`.)

`\@flupdates` This updates everything when a float is placed.

```
⟨*2ekernel⟩
\def \@flupdates #1#2#3{%
   \global \advance #1\m@ne
   \global \advance \@colnum \m@ne
   \@tempdima -\ht\@currbox
   \advance \@tempdima
     -\ifx #3\@empty \textfloatsep \else \floatsep \fi
   \global \advance #2\@tempdima
   \global \advance \@colroom \@tempdima
   \@cons #3\@currbox
}
⟨/2ekernel⟩
```

(*End of definition for* `\@flupdates`.)

Interesting facts about float mechanisms past and present, together with a summary of various features, some unresolved:

1. The value `\textfraction` does not affect the processing of doublecol floats: this seems sensible, but should be documented.
2. `\twocolumn` floatplacement was wrong: dbl not needed, ord needed.
3. `\@floatplacement` was not called after `\@startdblcol` or `\@topnewpage`. This has been changed; it is clearly a bug fix.
4. The use `\@topnewpage` when `\dblfigrule` is non-trivial produced a rule in the wrong place. This has been fixed by not using `\dblfigrule` when processing the 'float' from `\@topnewpage`.
5. If the specifier was just h and the float could not be put here, it went on the deferlist and stayed there until a clearpage. It now gets changed to a 'th': this is only an error-recovery action, putting just h or !h should be deprecated.
6. `\@dblmaxsep` was 'the maximum of `\dblfloatsep` and `\dbltexfloatsep`'. But it was never used! Now gone completely, like `\@maxsep`.
7. After an h float is put on a page, it was counted as text when applying the `\textfraction` test; this is possibly too big a change although it is a bug fix?
8. Two consecutive h floats are separated by twice `\intextsep`: this could be changed to one by use of `\addvspace`, OK? Note that it would also mean that less space is put in if an h float immediately follows other spaces. This is also possibly too big a change, at least for compatibility mode? Or it may be simply wrong! It has not been changed.
9. Now `\@addtocurcol` checks first for just p fps. I think that this is an increase in efficiency, but maybe the coding should be made even more efficient.
10. `\@tryfcolumn` now tests if the list is empty first, otherwise lots of wasted time! Thus this test has been removed from `\@startcolumn`. As Frank pointed out, this makes `\@startcolumn` less efficient. But it is now the same as `\@startdblcolumn`: I can see no reason why they should be different, but which is best?
11. Why is `\@colroom` set in `\@doclearpage`?
12. Footnotes. Check what `\clearpage` does when footnotes are left over. Footnotes are not put on float pages and, also, `\@addtonextcol` ignores the existence of held-over footnotes in deciding what floats can go on the page. Not changed.
13. `\clearpage` can still lose non-boxes, at least when floats are involved. It also moves some to the 'wrong page', but this may be a coding problem.
14. The ! option makes it necessary to check in `\output` that there is enough room left on the page after adding a float. (This would have been necessary anyway if anyone set `\@textmin` too close to zero! A similar danger existed also if the text in a `\twocolumn[text]` entity gets too large.) The current implementation of this also makes the normal case a little less efficient, OK? Not enough room means, at present, less than `\baselineskip`, with a warning: is this OK? Should it be made generic (another parameter)?

15. There are four possibilities for supporting this:

    `\twocolumn[\maketitle more text]`

    One is to change `\maketitle` slightly to allow this. Another is to change `\@topnewpage` so that more than one `\twocolumn[]` command is allowed; in this case `\maketitle\twocolumn[more text]` will work. The former is more robust from the user's viewpoint, but makes the code for `\maketitle` rather ad hoc (maybe it is already?). Another is to misuse the global twocolumn flag locally within `\@topnewpage`. Yet another is to move the column count register from the multicol package into the kernel. This has been done.

16. Where should the reinserts be put to maximise the probability that footnotes come out on the correct page? Or should we go for as much compatibility as possible (but see next item)?

17. Should we continue to support (as much as possible) `\samepage`? Some of its intended functionality is now advertised as being provided by `\enlargethispage`. Use of either is likely to result in wrongly placed footnotes, marginals, etc. Which should have priority: obeying the pagination instructions, or correct placement of notes/marginalia?

18. Is the adjustment of space to cause shrinking in the kludge-* case correct? Should it be limited to 0pt?

19. Is the setting of `\boxmaxdepth` in makecol and friends needed? It only has any effect if `\@textbottom` ends with a box or rule, in which case the vskip to allow for its depth should also be added. If it is kept, it should probably be the last thing in the box. It has now been removed.

    It would perhaps be better to document that `\@textbottom` and `\@texttop` must have natural height 0pt.

20. I cannot see why the vskip adjustment for the depth is needed if boxmaxdepth is used to ensure that there is never a too deep box.

21. The value of `\boxmaxdepth` should be explicitly set whenever necessary: it is too risky to assume that it has any particular value. Care is needed in deciding what to set it to.

    It is interesting to note that the value of `\boxmaxdepth` is unique in being read before the local settings for the box group are reset; all other parameter settings which affect the box construction use their values outside the box group.

22. Should `\@maxdepth` store the setting of `\maxdepth` from lplain? Or should we provide a proper interface to class files for setting these?

An analysis of various other macros.

`\@opcol` should do `\@floatplacement`, but where? Right at the end, since it always occurs at the start of a column.

```
\def\@opcol{%
  % Why is this done first?
  \global \@mparbottom \z@
  \if@twocolumn
```

```
    \@outputdblcol
  \else
    \@outputpage
    % This is not needed since it is done at the end of
    %   |\@outputpage|:
    \global \@colht \textheight
  \fi}
```

Only tracing has been added to these.

```
⟨latexrelease | fltrace⟩\IncludeInRelease{2017/01/01}%
⟨latexrelease | fltrace⟩  {\@makefcolumn}{negative height floats}%
⟨*2ekernel | fltrace | latexrelease⟩
\def\@makefcolumn #1{%
  \begingroup
    \@fpmin -\maxdimen
    \let \@testfp \@gobble
    \@tryfcolumn #1%
  \endgroup
⟨*trace⟩
  \if@fcolmade
    \fl@trace{PAGE: in \string\clearpage
                              \if@twocolumn ---twocolumn\fi---}%
    \fl@trace{----- float column/page completed from \string#1}%
  \fi
⟨/trace⟩
}
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨latexrelease | fltrace⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace⟩  {\@makefcolumn}{negative height floats}%
⟨latexrelease | fltrace⟩\def\@makefcolumn #1{%
⟨latexrelease | fltrace⟩  \begingroup
⟨latexrelease | fltrace⟩    \@fpmin \z@
⟨latexrelease | fltrace⟩    \let \@testfp \@gobble
⟨latexrelease | fltrace⟩    \@tryfcolumn #1%
⟨latexrelease | fltrace⟩  \endgroup
⟨*trace⟩
⟨latexrelease | fltrace⟩  \if@fcolmade
⟨latexrelease | fltrace⟩    \fl@trace{PAGE: in \string\clearpage
⟨latexrelease | fltrace⟩                      \if@twocolumn ---twocolumn\fi---}%
⟨latexrelease | fltrace⟩    \fl@trace{----- float column/page completed
⟨latexrelease | fltrace⟩                      from \string#1}%
⟨latexrelease | fltrace⟩  \fi
⟨/trace⟩
⟨latexrelease | fltrace⟩}
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨/2ekernel | fltrace | latexrelease⟩
```

This will line up the last baselines in the two columns provided they are constructed in the normal way: i.e. ending in a skip of minus the original depth, with `\@textbottom` adding nothing.

Thus again it is essential for `\@textbottom` to have depth 0pt.

```
⟨latexrelease | fltrace⟩\IncludeInRelease{2015/01/01}%
```

```
⟨latexrelease | fltrace⟩   {\@outputdblcol}{2 column marks}%
⟨*2ekernel | fltrace | latexrelease⟩
```

This is just a change to the single command `\@outputdblcol` so that it saves mark information for the first column and restores it in the second column.

```
\def\@outputdblcol{%
  \if@firstcolumn
    \global\@firstcolumnfalse
```

Save the left column

```
    \global\setbox\@leftcolumn\copy\@outputbox
⟨fltrace⟩     \fl@trace{PAGE: first column boxed}%
```

Remember the marks from the first column

```
    \splitmaxdepth\maxdimen
    \vbadness\maxdimen
```

In case of `\enlargethispage` we will have infinite negative glue at the bottom of the page (coming from `\vss`) and that will earn us an error message if we `\vsplit` to get at the marks. So we need to remove the last glue (if any) at the end of `\@outputbox` as we are only interested in marks that change doesn't matter.

```
      \setbox\@outputbox\vbox{\unvbox\@outputbox\unskip}%
      \setbox\@outputbox\vsplit\@outputbox to\maxdimen
```

One minor difference from the current `fixmarks` package, pass the marks through a token register to stop any `#` tokens causing an error in a `\def`.

```
    \toks@\expandafter{\topmark}%
    \xdef\@firstcoltopmark{\the\toks@}%
    \toks@\expandafter{\splitfirstmark}%
    \xdef\@firstcolfirstmark{\the\toks@}%
```

This test does not work if truly empty marks have been inserted, but LaTeX marks should always have (at least) two brace groups. (Except before the first mark is used, when the marks are empty, but that is OK here.)

```
    \ifx\@firstcolfirstmark\@empty
      \global\let\@setmarks\relax
    \else
      \gdef\@setmarks{%
        \let\firstmark\@firstcolfirstmark
        \let\topmark\@firstcoltopmark}%
    \fi
```

End of change

```
  \else
    \global\@firstcolumntrue
    \setbox\@outputbox\vbox{%
     \hb@xt@\textwidth{%
        \hb@xt@\columnwidth{\box\@leftcolumn \hss}%
        \hfil
```

The color of the `\vrule` should be `\normalcolor` as to not inherit the color from the column.

```
        {\normalcolor\vrule \@width\columnseprule}%
        \hfil
       \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
⟨fltrace⟩     \fl@trace{PAGE: second column also boxed}%
  \@combinedblfloats
```

Override current first and top with those of first column if necessary

```
    \@setmarks
```

End of change

```
    \@outputpage
⟨fltrace⟩    \fl@trace{PAGE: two column page completed}%
    \begingroup
      \@dblfloatplacement
      \@startdblcolumn
      \@whilesw\if@fcolmade \fi{\@outputpage
⟨fltrace⟩      \fl@trace{PAGE: double float page completed}%
     \@startdblcolumn}%
    \endgroup
  \fi}%
```

```
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨latexrelease | fltrace⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease | fltrace⟩  {\@outputdblcol}{2 column marks}%
⟨latexrelease | fltrace⟩\def\@outputdblcol{%
⟨latexrelease | fltrace⟩  \if@firstcolumn
⟨latexrelease | fltrace⟩    \global \@firstcolumnfalse
⟨latexrelease | fltrace⟩    \global \setbox\@leftcolumn \box\@outputbox
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{PAGE: first column boxed}%
⟨/trace⟩
⟨latexrelease | fltrace⟩  \else
⟨latexrelease | fltrace⟩    \global \@firstcolumntrue
⟨latexrelease | fltrace⟩    \setbox\@outputbox \vbox {%
⟨latexrelease | fltrace⟩                        \hb@xt@\textwidth {%
⟨latexrelease | fltrace⟩                          \hb@xt@\columnwidth {%
⟨latexrelease | fltrace⟩                            \box\@leftcolumn \hss}%
⟨latexrelease | fltrace⟩                          \hfil
⟨latexrelease | fltrace⟩                          {\normalcolor\vrule
⟨latexrelease | fltrace⟩                              \@width\columnseprule}%
⟨latexrelease | fltrace⟩                          \hfil
⟨latexrelease | fltrace⟩                          \hb@xt@\columnwidth {%
⟨latexrelease | fltrace⟩                            \box\@outputbox \hss}%
⟨latexrelease | fltrace⟩                                                }%
⟨latexrelease | fltrace⟩                             }%
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{PAGE: second column also boxed}%
⟨/trace⟩
⟨latexrelease | fltrace⟩    \@combinedblfloats
⟨latexrelease | fltrace⟩    \@outputpage
⟨*trace⟩
⟨latexrelease | fltrace⟩    \fl@trace{PAGE: two column page completed}%
⟨/trace⟩
⟨latexrelease | fltrace⟩    \begingroup
⟨latexrelease | fltrace⟩      \@dblfloatplacement
⟨latexrelease | fltrace⟩      \@startdblcolumn
```

This loop could be replaced by an `\expandafter` tail recursion in `\@startdblcolumn`.

```
⟨latexrelease | fltrace⟩      \@whilesw\if@fcolmade \fi
⟨latexrelease | fltrace⟩        {\@outputpage
```

```
⟨*trace⟩
⟨latexrelease | fltrace⟩       \fl@trace{PAGE: double float page completed}%
⟨/trace⟩
⟨latexrelease | fltrace⟩          \@startdblcolumn}%
⟨latexrelease | fltrace⟩     \endgroup
⟨latexrelease | fltrace⟩   \fi
⟨latexrelease | fltrace⟩}%
⟨latexrelease | fltrace⟩\EndIncludeInRelease
⟨/2ekernel | fltrace | latexrelease⟩
```

1.1.3 Float placement parameters

The main purpose of this section is to ensure that all the float-placement parameters which need to be set in a class file or package have been declared. It also describes their use and sets values for them which are reasonable for typical documents using US letter or A4 sized paper.

Limits for the placement of floating objects

`\c@topnumber` This counter holds the maximum number of floats that can appear at the top of a text page or column.

```
⟨*2ekernel⟩
\newcount\c@topnumber
\setcounter{topnumber}{2}
```

(*End of definition for* `\c@topnumber`.)

`\topfraction` This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by floats at the top.

```
\newcommand\topfraction{.7}
```

(*End of definition for* `\topfraction`.)

`\c@bottomnumber` This counter holds the maximum number of floats that can appear at the bottom of a text page or column.

```
\newcount\c@bottomnumber
\setcounter{bottomnumber}{1}
```

(*End of definition for* `\c@bottomnumber`.)

`\bottomfraction` This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by floats at the bottom.

```
\newcommand\bottomfraction{.3}
```

(*End of definition for* `\bottomfraction`.)

`\c@totalnumber` This counter holds the maximum number of floats that can appear on any text page or column.

```
\newcount\c@totalnumber
\setcounter{totalnumber}{3}
```

(*End of definition for* `\c@totalnumber`.)

`\textfraction` This macro holds the minimum proportion (as a decimal number) of a text page or column that must be occupied by text.

```
\newcommand\textfraction{.2}
```

(*End of definition for* `\textfraction`.)

`\floatpagefraction` This macro holds the minimum proportion (as a decimal number) of a page or column that must be occupied by floating objects before a 'float page' is produced.

```
\newcommand\floatpagefraction{.5}
```

(*End of definition for* `\floatpagefraction`.)

`\c@dbltopnumber` This counter holds the maximum number of double-column floats that can appear on the top of a two-column text page.

```
\newcount\c@dbltopnumber
\setcounter{dbltopnumber}{2}
```

(*End of definition for* `\c@dbltopnumber`.)

`\dbltopfraction` This macro holds the maximum proportion (as a decimal number) of a two-column text page that can be occupied by double-column floats at the top.

```
\newcommand\dbltopfraction{.7}
```

(*End of definition for* `\dbltopfraction`.)

`\dblfloatpagefraction` This macro holds the minimum proportion (as a decimal number) of a page that must be occupied by double-column floating objects before a 'double-column float page' is produced.

```
\newcommand\dblfloatpagefraction{.5}
```

(*End of definition for* `\dblfloatpagefraction`.)

Floats on a text page

`\floatsep` `\textfloatsep` `\intextsep` When a floating object is placed on a page with text, these parameters control the separation between the float and the other objects on the page. These parameters are used for both one-column mode and single-column floats in two-column mode. They are all rubber lengths.

`\floatsep` is the space between adjacent floats that are placed at the top or bottom of the text page or column.

`\textfloatsep` is the space between the main text and floats at the top or bottom of the page or column.

`\intextsep` is the space between in-text floats and the text.

```
\newskip\floatsep
\newskip\textfloatsep
\newskip\intextsep
\setlength\floatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
\setlength\intextsep   {12\p@ \@plus 2\p@ \@minus 2\p@}
```

(*End of definition for* `\floatsep`, `\textfloatsep`, *and* `\intextsep`.)

\dblfloatsep
\dbltextfloatsep

When double-column floats (floating objects that span the whole \textwidth) are placed at the top of a text page in two-column mode, the separation between the float and the text is controlled by \dblfloatsep and \dbltextfloatsep. They are rubber lengths.

\dblfloatsep is the space between adjacent double-column floats placed at the top of the text page.

\dbltextfloatsep is the space between the main text and double-column floats at the top of the page.

```
\newskip\dblfloatsep
\newskip\dbltextfloatsep
\setlength\dblfloatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
```

(*End of definition for* \dblfloatsep *and* \dbltextfloatsep.)

Floats on their own page or column

\@fptop
\@fpsep
\@fpbot

When floating objects are placed on a separate page or column, called a 'float page', the layout of the page is controlled by these parameters, which are rubber lengths.

At the top of the page \@fptop is inserted; typically this supplies some stretchable whitespace. At the bottom of the page \@fpbot is inserted. Between adjacent floats \@fpsep is inserted.

These parameters are used for all floating objects on a 'float page' in one-column mode, and for single-column floats in two-column mode.

Note that at least one of the two parameters \@fptop and \@fpbot should contain a plus ...fil so as to fill the remaining empty space.

```
\newskip\@fptop
\newskip\@fpsep
\newskip\@fpbot
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{8\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
```

(*End of definition for* \@fptop, \@fpsep, *and* \@fpbot.)

\@dblfptop
\@dblfpsep
\@dblfpbot

Double-column 'float pages' in two-column mode use similar parameters.

```
\newskip\@dblfptop
\newskip\@dblfpsep
\newskip\@dblfpbot
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{8\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
```

(*End of definition for* \@dblfptop, \@dblfpsep, *and* \@dblfpbot.)

\topfigrule
\botfigrule
\dblfigrule

The macros can be used to put in rules between floats and text; whatever they insert should be vertical mode material which takes up zero space.

```
\let\topfigrule=\relax
\let\botfigrule=\relax
\let\dblfigrule=\relax
⟨/2ekernel⟩
```

(*End of definition for* \topfigrule, \botfigrule, *and* \dblfigrule.)

File 55
lttagging.dtx

```
⟨*2ekernel | latexrelease⟩
\ExplSyntaxOn
```

1 General support for tagged output

\SuspendTagging
\ResumeTagging
\tag_suspend:n
\tag_resume:n

The are places in code where it is import top stop any tagging activities, e.g., when we are doing trial typesetting that it is done several times. In such a case one must tag only the final version that is actually used, otherwise tagging structures are generated which then do not end up in the PDF and confuse the mechanism. For this we have two commands that can be used in packages: `\SuspendTagging` and `\ResumeTagging` (with corresponding L3 programming layer commands). They are available as part of the LaTeX kernel, so that they can be safely used in packages whether or not tagging is requested. They all take a string argument that is used for debugging to easily identify why tagging was suspended or restarted, for example, in tabularx you find `\SuspendTagging{tabularx}`. By default these four commands do nothing.

The argument is used literally (in `\typeout` messages) without any expansion when debugging is turned on and otherwise it is not used at all. This means it is safe to write something like `\SuspendTagging{\foo}` or even `\SuspendTagging\foo` which means LaTeX has to parse only a single token instead of putting a string of characters into the argument. This means a tiny speed improvement but with many such debugging strings...

\UseTaggingSocket
\tag_socket_use:n
\tag_socket_use:nn
\tag_socket_use:nnn

Given that we sometimes have to suspend tagging, it would be fairly inefficient to put different plugs into these sockets whenever that happens. We therefore offer `\UseTaggingSocket` which is like `\UseSocket` except that is expects a socket starting with `tagsupport/` but the socket name is specified without this prefix, i.e.,

$$\texttt{\UseTaggingSocket\{foo\}} \rightarrow \texttt{\UseSocket\{tagsupport/foo\}}$$

.

Beside being slightly shorter, the big advantage is that this way we can change `\UseTaggingSocket` to do nothing by switching a boolean instead of changing the plugs of the tagging support sockets back and forth.

Usually, these sockets have (beside the default plug defined for every socket) one additional plug defined and directly assigned. This plug is used when tagging is active. There may be more plugs, e.g., tagging with special debugging or special behaviour depending on the class or PDF version etc., but right now it is usually just on or off.

When tagging is suspended they all have the same predefined behaviour: The sockets with zero arguments do nothing. The sockets with one argument gobble their argument. The sockets with two arguments will drop their first argument and pass the second unchanged.

It is possible to use the tagging support sockets with `\UseSocket` directly, but in this case the socket remains active if `\SuspendTagging` is in force. There may be reasons for doing that but in general we expect to always use `\UseTaggingSocket`.

\UseExpandableTaggingSocket
\tag_socket_use_expandable:n

For special cases like in some `\halign` contexts we need a fully expandable version of the commend. For these cases, `\UseExpandableTaggingSocket` can be used. To allow

being expandable, it does not output any debugging information if \DebugSocketsOn is in effect and therefore should be avoided whenever possible.

The L3 programming layer versions \tag_socket_use_expandable:n, \tag_socket_use:n, \tag_socket_use:nn, and \tag_socket_use:nnn are slightly more efficient than \UseTaggingSocket because they do not have to determine how many arguments the socket takes when disabling it.

2 Implementation

\tag_suspend:n
\tag_resume:n
\SuspendTagging
\ResumeTagging

In the kernel, these commands get dummy definitions so that they can be used without harm in packages. The real definition is used when tagging gets enabled.

```
\cs_new_eq:NN \tag_suspend:n \use_none:n
\cs_new_eq:NN \tag_resume:n  \use_none:n
\cs_new_protected:Npn \SuspendTagging #1 { \tag_suspend:n {#1} }
\cs_new_protected:Npn \ResumeTagging #1  { \tag_resume:n {#1} }
```

(*End of definition for* \tag_suspend:n *and others.*)

\tag_socket_use:n
\tag_socket_use:nn
\tag_socket_use:nnn
\tag_socket_use_expandable:n
\UseTaggingSocket
\UseExpandableTaggingSocket

Again this is not the final definition for the kernel; it is just a version to get going while some parts of the kernel support are still missing.

```
\AddToHook{begindocument}[kernel]{
  \cs_if_exist:NF \tag_if_active:T
     {
       \prg_new_conditional:Npnn \tag_if_active: { p , T , TF, F }
           { \prg_return_false: }
     }
}
```

Dummy definitions in the kernel. These definitions will get updated in tagpdf. The default in the kernel is just to get rid of the first argument, the second is preserved if present:

```
\cs_new:Npn \tag_socket_use_expandable:n #1 { }
\cs_new_protected:Npn \tag_socket_use:n #1 { }
\cs_new_protected:Npn \tag_socket_use:nn #1#2 { }
\cs_new_protected:Npn \tag_socket_use:nnn #1#2#3 { #3 }
\cs_new_protected:Npn \UseTaggingSocket #1 {
  \int_case:nnF
      { \int_use:c { c__socket_tagsupport/#1_args_int } }
         {
           0 \prg_do_nothing:
           1 \use_none:n
           2 \use_ii:nn
```

We do not expect tagging sockets with more than one or two arguments, so for now we only provide those.

```
         }
         \ERROR usetaggingsocket  % that should get a proper error message
}
```

The same as an expandable command:

```
\cs_new:Npn \UseExpandableTaggingSocket #1 {
  \int_case:nnF
      { \int_use:c { c__socket_tagsupport/#1_args_int } }
```

```
        {
          0 \prg_do_nothing:
          1 \use_none:n
          2 \use_ii:nn
        }
        \ERROR usetaggingsocket  % that should get a proper error message
}
```

(*End of definition for* `\tag_socket_use:n` *and others.*)

2.1 Tagging sockets

This collects tagging sockets that should be generally available so that they can also be used even if the tagging code is not loaded.

2.1.1 Tagging support for paragraph setup

Paragraphs are tagged through the code in the para/hooks. This code is sometimes adjusted, e.g. to produce a "flattened" paragraph or to use a different tag. Sockets related to such code parts are collected here.

`\l__tag_block_flattened_level_int` The block code needs to know if they are nested blockenvs inside a flattened environment. For this it uses a counter. Inside some contexts, e.g. at the begin of a minipage or a footnote this counter must be reset. We therefore define the counter here so that we can use it in the following socket.

```
\int_new:N \l__tag_block_flattened_level_int
```

(*End of definition for* `\l__tag_block_flattened_level_int`.)

`tagsupport/para/restore` (*socket*) This socket restores the para related settings to their default. It should be used in places where "normal" paragraph tagging must be ensured, for example at the begin of a footnote.

```
\NewSocket{tagsupport/para/restore}{0}
```

`default` (*plug*)

```
\NewSocketPlug{tagsupport/para/restore}{default}
 {
   \tl_set:Nn    \l__tag_para_main_tag_tl {text-unit}
   \tl_set_eq:NN \l__tag_para_tag_tl\l__tag_para_tag_default_tl
   \bool_set_false:N\l__tag_para_flattened_bool
   \int_zero:N   \l__tag_block_flattened_level_int
   \bool_set_true:N \l__tag_para_bool
 }
\AssignSocketPlug{tagsupport/para/restore}{default}
```

`tagsupport/para/begin` (*socket*)
`tagsupport/para/end` (*socket*)
These sockets are currently defined in tagpdf. They overwrite definitions in the latex-lab-block code. There is also a simpler definition that probably should be a general socket too. TODO: move this into lttagging.

2.1.2 Tagging socket for targets

gsupport/refstepcounter (*socket*)

When tagging is active we want to track the current structure number when targets are set. This will be mostly used in `\refstepcounter` but also if targets are set manually.

```
\NewSocket{tagsupport/recordtarget}{0}
```

tagsupport/recordtarget) (*plug*)

```
%
\NewSocketPlug{tagsupport/recordtarget}{kernel}
  {
    \tl_if_blank:VF \@currentHref
    {
      \prop_gput:Nee
        \g__tag_struct_dest_num_prop
        {\@currentHref}
        {\tag_get:n{struct_num}}
    }
  }
\AssignSocketPlug{tagsupport/recordtarget}{kernel}
\ExplSyntaxOff
```

2.1.3 Tagging sockets for toc

toc/contentsline/before (*socket*)
/toc/contentsline/after (*socket*)

Tagging sockets at the begin and end of contentsline. They receive *all* contentsline arguments as one argument in four brace groups. The socket code should then use the parts it needs.

```
\NewSocket{tagsupport/toc/contentsline/before}{1}
\NewSocket{tagsupport/toc/contentsline/after}{1}
```

ort/toc/starttoc/before (*socket*)
port/toc/starttoc/after (*socket*)

Tagging sockets for the begin and end of start of `\@starttoc`. They take one argument, the extension.

```
\NewSocket{tagsupport/toc/starttoc/before}{1}
\NewSocket{tagsupport/toc/starttoc/after}{1}
```

port/toc/leaders/before (*socket*)
port/toc/leaders/after (*socket*)

Tagging sockets to make the dot leaders an artifact. They do not take an argument.

```
\NewSocket{tagsupport/toc/leaders/before}{0}
\NewSocket{tagsupport/toc/leaders/after}{0}
```

2.1.4 Tagging support for table/tabular packages

The code uses a number of sockets to inject the tagging commands. These can be easily set to a noop-plug in case the automated tagging is not wanted.

gsupport/tbl/cell/begin (*socket*)
agsupport/tbl/cell/end (*socket*)
gsupport/tbl/pcell/end (*socket*)
gsupport/tbl/pcell/end (*socket*)
gsupport/tbl/row/begin (*socket*)
tagsupport/tbl/row/end (*socket*)

At first sockets for the begin and end of cells and table rows:

```
\NewSocket{tagsupport/tbl/cell/begin}{0}
\NewSocket{tagsupport/tbl/cell/end}{0}
```

```
\NewSocket{tagsupport/tbl/row/begin}{0}
\NewSocket{tagsupport/tbl/row/end}{0}
```

Multi-line cells have their own sockets (as they start out in vertical mode and need different treatment).

```
\NewSocket{tagsupport/tbl/pcell/begin}{0}
\NewSocket{tagsupport/tbl/pcell/end}{0}
```

`tagsupport/tbl/init` (*socket*) This socket should be at the begin of the table, inside a group. It is used for settings such as disabling para-tagging inside the table. This socket can perhaps be merged later into the begin-sockets.

```
\NewSocket{tagsupport/tbl/init}{0}
```

`pport/tbl/init/celldata` (*socket*) This socket is used in `\tbl_init_cell_data_for_table`, the command that stores and initialize cell data to handle nested tables. It can be used to restore similar tagging related values

```
\NewSocket{tagsupport/tbl/init/celldata}{0}
```

`tagsupport/tbl/finalize` (*socket*) To fine tune the structure (change cells to header cells, remove unwanted structures, move a foot to the end, etc.). We also need a socket that is executed at the end of the table but *before* all the variables are restored to the outer or default values. The code in the socket can make assignments, but probably shouldn't do typesetting and not write whatsits.

```
\NewSocket{tagsupport/tbl/finalize}{0}
```

`t/tbl/restore/celldata` (*socket*) This socket is used in `\tbl_restore_outer_cell_data:`, the command that restores cell data when quitting a nested table. It can be used to restore similar tagging related values

```
\NewSocket{tagsupport/tbl/restore/celldata}{0}
```

`tagsupport/tbl/colspan` (*socket*) This socket is used to manage spanning cells, e.g., a `\multicolumn`. It expects one argument (the number of cells spanned) and if tagging is enabled set appropriate tag attributes in the background. We probably need a similar socket for row spans eventually.

```
\NewSocket{tagsupport/tbl/colspan}{1}
```

`support/tbl/hmode/begin` (*socket*)
`gsupport/tbl/hmode/end` (*socket*)
`support/tbl/vmode/begin` (*socket*)
`gsupport/tbl/vmode/end` (*socket*)

These sockets are used in the begin and end code of environments, to allow a fast enabling and disabling of the tagging. We distinguish between tables that can be used inside paragraphs and standalone tables such as `longtable` that are always in vertical mode.

```
\NewSocket{tagsupport/tbl/hmode/begin}{0}
\NewSocket{tagsupport/tbl/hmode/end}{0}
\NewSocket{tagsupport/tbl/vmode/begin}{0}
\NewSocket{tagsupport/tbl/vmode/end}{0}
```

`ort/tbl/longtable/init` (*socket*)
`/tbl/longtable/finalize` (*socket*)

`longtable` needs its own sockets to fine tune the structure. Simply switching the plug in the previous socket interferes with enabling/disabling the tagging.

```
\NewSocket{tagsupport/tbl/longtable/init}{0}
\NewSocket{tagsupport/tbl/longtable/finalize}{0}
```

`ort/tbl/longtable/head` (*socket*)
`ort/tbl/longtable/foot` (*socket*)

Header and footer boxes need special handling because they are repeatedly used.

```
\NewSocket{tagsupport/tbl/longtable/head}{0}
\NewSocket{tagsupport/tbl/longtable/foot}{0}
```

`port/tbl/leaders/begin` (*socket*)
`support/tbl/leaders/end` (*socket*)

Sockets around leaders such as rules or dotted lines, that should be tagged as artifacts, used, for example, in `\cline`.

```
\NewSocket{tagsupport/tbl/leaders/begin}{0}
\NewSocket{tagsupport/tbl/leaders/end}{0}
```

2.1.5 Tagging Support for floats

pport/float/hmode/begin (*socket*)
support/float/hmode/end (*socket*)

These sockets are used if the float is called in hmode.

```
\NewSocket{tagsupport/float/hmode/begin}{0}
\NewSocket{tagsupport/float/hmode/end}{0}
```

tagsupport/float/begin (*socket*)
tagsupport/float/end (*socket*)

These sockets start and stop the float structure.

```
\NewSocket{tagsupport/float/begin}{0}
\NewSocket{tagsupport/float/end}{0}
```

gsupport/caption/begin (*socket*)
tagsupport/caption/end (*socket*)

These sockets are used in `\@makecaption`. They open and close the `Caption` structure. Their default plugs assume that they are used in vmode. The argument of the begin socket is the structure number of the parent float. If it is empty the current structure number is used.

```
\NewSocket{tagsupport/caption/begin}{1}
\NewSocket{tagsupport/caption/end}{0}
```

rt/caption/label/begin (*socket*)
port/caption/label/end (*socket*)

These sockets are used in `\@makecaption` around the label. Their default plugs ensure that the label is outside the paragraph and that the rest of the caption uses flattened para mode. If the caption is not in a hbox, the `para/begin` socket should follow to properly start the paragraph.

```
\NewSocket{tagsupport/caption/label/begin}{0}
\NewSocket{tagsupport/caption/label/end}{0}
```

3 For lttab.dtx parked here for now

```
⟨@@=tbl⟩
\ExplSyntaxOn
```

3.1 Variables for row, column and span counting

This part needs a decision on names for various integer registers as well as a decision if those should be also made available for LaTeX 2ε-style packages in form of 2e names and or as non-internals for the L3 programming layer.

At the moment they are all internal but this probably has to change.

\g__tbl_col_int
\g__tbl_row_int
\g__tbl_span_tl
\g__tbl_table_cols_tl

`\g__tbl_row_int` holds the current row number in the table. The value `0` means we haven't yet processed the table preamble (or in case of longtable are just in front of the next chunk to be processed). It is incremented by every `\cr` including the one ending the table preamble.

TODO: due to the gymnastics needed inside the longtable code the row counter is directly exposed there rather than hidden by interfaces. This needs changing when it is decided how to manage these counters.

`\g__tbl_col_int` holds the current column number. The value `0` means we have not yet started the table or just finished a table row (with `\\` typically); any other positive value means we are currently typesetting a cell in that column in some row (denoted by the `\g__tbl_row_int`).

In a `\multicolumn` it holds the column number of the first spanned column and `\g__tbl_span_tl` the info how many cells are spanned.

`\g__tbl_span_tl` is normally `1` except in a `\multicolumn` cell.

```
\int_new:N \g__tbl_col_int
```

```
\int_new:N \g__tbl_row_int
\tl_new:N  \g__tbl_span_tl
\tl_new:N  \g__tbl_table_cols_tl

\tl_gset:Nn \g__tbl_span_tl {1}
\tl_gset:Nn \g__tbl_table_cols_tl {0}  % indicates outer level
```

(*End of definition for* `\g__tbl_col_int` *and others.*)

`\l__tbl_saved_col_tl`
`\l__tbl_saved_row_tl`
`\l__tbl_saved_span_tl`
`\l__tbl_saved_table_cols_tl`

Saving the outer values if we are nesting tables is necessary (as the above variables are globally altered). For this we always use token lists because they don't change and we do not need to blow additional integer registers.

```
\tl_new:N \l__tbl_saved_col_tl
\tl_new:N \l__tbl_saved_row_tl
\tl_new:N \l__tbl_saved_span_tl
\tl_new:N \l__tbl_saved_table_cols_tl

\tl_set:Nn \l__tbl_saved_col_tl{0}
\tl_set:Nn \l__tbl_saved_row_tl{0}
\tl_set:Nn \l__tbl_saved_span_tl{1}
\tl_set:Nn \l__tbl_saved_table_cols_tl{0}  % indicates outer level
```

(*End of definition for* `\l__tbl_saved_col_tl` *and others.*)

`\g__tbl_missingcells_int`

This will contain the number of missing cells in a row:

```
\int_new:N \g__tbl_missing_cells_int
```

(*End of definition for* `\g__tbl_missingcells_int`.)

3.2 Tracing/debugging

`\DebugTablesOn`
`\DebugTablesOff`

```
\def\DebugTablesOn{
  \cs_set_eq:NN \__tbl_trace:n \typeout
}
\def\DebugTablesOff{
  \cs_set_eq:NN \__tbl_trace:n \use_none:n
}
```

```
\cs_new_eq:NN \__tbl_trace:n \use_none:n
```

(*End of definition for* `\DebugTablesOn` *and* `\DebugTablesOff`.)

3.3 Interface commands

All interface commands for the cell number determination have to be public on some level because they are needed in other packages as well, e.g., longtable. We may or may not also want to provide 2e style names for them.

`\tbl_update_cell_data:`

Updating cell data in columns after the first means we have to increment the `\g__tbl_col_int` by the span count of the previous cell (in case it was a `\multicolumn`) and then reset the `\g__tbl_span_tl` to one (as the default).

```
\cs_new_protected:Npn \tbl_update_cell_data: {
        \int_gadd:Nn \g__tbl_col_int { \g__tbl_span_tl }
        \tl_gset:Nn  \g__tbl_span_tl {1}
}
```

(*End of definition for* `\tbl_update_cell_data:`.)

`\tbl_count_table_cols:` Current implementation of `\@mkpream` uses the scratch counter `\count@` to keep track of the number of toks registers it needs (2 per column), but this can't be used as it counts also insertions made with `!{}` and `@{}`. So similar as does longtable for `\LT@cols` we count the numbers of ampersands instead.

```
\cs_new:Npn \tbl_count_table_cols:  {
  \seq_set_split:NnV\l__tbl_tmpa_seq {&}\@preamble
  \tl_gset:Ne \g__tbl_table_cols_tl { \seq_count:N \l__tbl_tmpa_seq }
  \__tbl_trace:n { ==>~ Table~ has~ \g__tbl_table_cols_tl \space columns }
}
```

(*End of definition for* `\tbl_count_table_cols:`.)

`\l__tbl_tmpa_seq`

```
\seq_new:N \l__tbl_tmpa_seq
```

(*End of definition for* `\l__tbl_tmpa_seq`.)

`\tbl_count_missing_cells:n` We might have the situation that some table package has not implemented the `\tbl_count_table_cols:` in which case `\g__tbl_table_cols_tl` would always be zero and we would get an error below when we try to determine the missing cells, so bypass that calculation if we aren't doing tagging (there the packages should have the proper code added). Recall that this is code, that is called by `\\` and an old table packagee might rely on whatever the LaTeX kernel offers here.

```
\cs_new:Npn \tbl_count_missing_cells:n #1 {
  \tag_if_active:T {
    \int_compare:nNnT \g__tbl_col_int > 0
      {
        \int_gset:Nn \g__tbl_missing_cells_int
            {
              \g__tbl_table_cols_tl
            - \g__tbl_col_int
            - \g__tbl_span_tl
            + 1
            }
        \int_compare:nNnT \g__tbl_missing_cells_int < 0 \ERRORmissingcells % should not happe
        \__tbl_trace:n{==>~
          (#1)~
          This~ row~ needs~
          \int_use:N \g__tbl_missing_cells_int \space
          additional~ cell(s)
        }
      }
  }
}
```

(*End of definition for* `\tbl_count_missing_cells:n`.)

`\tbl_save_outer_table_cols:`

```
\cs_new_protected:Npn \tbl_save_outer_table_cols: {
  \tl_set_eq:NN \l__tbl_saved_table_cols_tl  \g__tbl_table_cols_tl
}
```

(*End of definition for* `\tbl_save_outer_table_cols:`.)

`\tbl_init_cell_data_for_table:`

```
\cs_new_protected:Npn \tbl_init_cell_data_for_table: {
  \tl_set:No \l__tbl_saved_col_tl {\int_use:N \g__tbl_col_int }
  \tl_set:No \l__tbl_saved_row_tl {\int_use:N \g__tbl_row_int }
  \tl_set_eq:NN \l__tbl_saved_span_tl  \g__tbl_span_tl
%
  \__tbl_trace:n { ==>~ saved~cell~data:~
                \l__tbl_saved_row_tl,
                \l__tbl_saved_col_tl,
                \l__tbl_saved_span_tl \space
                (
                \int_compare:nNnTF \l__tbl_saved_table_cols_tl = 0
                    { outer~ level }
                    { max:~ \l__tbl_saved_table_cols_tl }
                )
          }
```

Tagging has to initialize cell data too.

```
  \UseTaggingSocket{tbl/init/celldata}
```

These are the initial values when starting a table:

```
  \int_gzero:N \g__tbl_row_int
  \int_gzero:N \g__tbl_col_int
  \tl_gset:Nn  \g__tbl_span_tl {1}
}
```

(*End of definition for* `\tbl_init_cell_data_for_table:`.)

`\tbl_update_cell_data_for_next_row:`

```
\cs_new_protected:Npn \tbl_update_cell_data_for_next_row: {
      \int_gincr:N \g__tbl_row_int          % this row about to start
      \int_gzero:N \g__tbl_col_int          % we are before first col
}
```

(*End of definition for* `\tbl_update_cell_data_for_next_row:`.)

`\tbl_init_cell_data_for_row:` If we start processing a cell in the first column we set `\g__tbl_col_int` to `1` as we are no longer "at" but "in" the first column. We also set `\g__tbl_span_tl` to its default value (not spanning cells).

```
\cs_new_protected:Npn \tbl_init_cell_data_for_row: {
        \int_gset:Nn \g__tbl_col_int {1}
        \tl_gset:Nn  \g__tbl_span_tl {1}
}
```

(*End of definition for* `\tbl_init_cell_data_for_row:`.)

`\tbl_if_row_was_started:T`
`\tbl_if_row_was_started:TF` We use `\g__tbl_col_int` equal zero to indicate that we are just after a TR (i.e.n between rows or at the very beginning of the table). Using the row count is not so good as longtable may split the table in chunks.

These conditionals have to be expandable (i.e., unprotected) as they are sometimes executed when TEX is scanning inside a table.

```
\cs_new:Npn \tbl_if_row_was_started:T {
      \int_compare:nNnT \g__tbl_col_int > 0
```

```
}
\cs_new:Npn \tbl_if_row_was_started:TF {
      \int_compare:nNnTF \g__tbl_col_int > 0
}
```

(*End of definition for* `\tbl_if_row_was_started:T` *and* `\tbl_if_row_was_started:TF`.)

`\tbl_gzero_row_count:` `\tbl_gincr_row_count:` `\tbl_gdecr_row_count:`

This here is basically a temporary interface. What it will be in the end depends on what we decide concerning exposing row and column counters, if they stay internal we need something like this here (perhaps using `gincr` etc, or perhaps some other names in the first place).

```
\cs_new_protected:Npn \tbl_gzero_row_count: {
  \int_gzero:N \g__tbl_row_int
}
\cs_new_protected:Npn \tbl_gincr_row_count: {
  \int_gincr:N \g__tbl_row_int
}
\cs_new_protected:Npn \tbl_gdecr_row_count: {
  \int_gdecr:N \g__tbl_row_int
}
```

(*End of definition for* `\tbl_gzero_row_count:` , `\tbl_gincr_row_count:` , *and* `\tbl_gdecr_row_count:`.)

`\tbl_inbetween_rows:`

Again name is not really brilliant so far.

```
\cs_new_protected:Npn \tbl_inbetween_rows: {
       \int_gzero:N  \g__tbl_col_int
}
```

(*End of definition for* `\tbl_inbetween_rows:`.)

`\tbl_restore_outer_cell_data:`

```
\cs_new_protected:Npn \tbl_restore_outer_cell_data: {
  \int_gset:Nn \g__tbl_col_int { \l__tbl_saved_col_tl }
  \int_gset:Nn \g__tbl_row_int { \l__tbl_saved_row_tl }
  \tl_gset_eq:NN \g__tbl_span_tl \l__tbl_saved_span_tl
  \tl_gset_eq:NN \g__tbl_table_cols_tl   \l__tbl_saved_table_cols_tl
  \UseTaggingSocket{tbl/restore/celldata}
  \__tbl_trace:n { ==>~ restored~cell~data:~
                \int_use:N \g__tbl_row_int,
                \int_use:N \g__tbl_col_int,
                \l__tbl_saved_span_tl \space
                (
                \int_compare:nNnTF \g__tbl_table_cols_tl = 0
                    { outer~ level }
                    { max:~ \g__tbl_table_cols_tl }
                )
              }
}
```

(*End of definition for* `\tbl_restore_outer_cell_data:`.)

`\tbl_update_multicolumn_cell_data:n`

This macro updates `\g__tbl_col_int` and `\g__tbl_span_tl` inside a `\multicolumn` and possibly calls the tagging socket `tbl/row/begin`.

```
\cs_new_protected:Npn \tbl_update_multicolumn_cell_data:n #1 {
```

We execute socket for tagging only if this \multicolumn replaces the preamble of the first column. In that case we also have to set \g__tbl_col_int to 1 because this is no longer done in the preamble for the cell either.

```
    \int_compare:nNnTF \g__tbl_col_int = 0
        {
          \UseTaggingSocket{tbl/row/begin}
          \int_gset:Nn \g__tbl_col_int {1}
        }
```

If we are in a later column we use \g__tbl_span_tl from the previous column to update.

```
        {
          \int_gadd:Nn \g__tbl_col_int { \g__tbl_span_tl }
        }
```

Then we set the span value so that it can be use in the next column.

```
    \tl_gset:Nn \g__tbl_span_tl {#1}
}
```

(*End of definition for* \tbl_update_multicolumn_cell_data:n.)

\tbl_crcr:n This macro is used instead of the usual \crcr at the end of a table. It is deliberately defined without protection because it may get expanded by the scanning mechanism of low-level TEX after a final \cr (aka \\) in the table. In that case it shouldn't stop the expansion and the conditional inside will be false, thus it just vanishes without doing anything. If there are missing cells (in which case we also haven't see \cr yet) the macro \tbl_count_missing_cells:n is executed and then the row is finished with a final \cr.

```
\cs_new:Npn \tbl_crcr:n #1 {
    \int_compare:nNnT \g__tbl_col_int > 0
        {
          \tbl_count_missing_cells:n {#1}
        }
```

Even if we are at the start of a row we my have to do a \cr, so we do a \crcr always at the end.

```
    \crcr
}
```

(*End of definition for* \tbl_crcr:n.)

```
\ExplSyntaxOff
⟨@@=⟩
```

This is needed for longtable because \refstepcounter is setting up a target when hyperref is loaded and we don't want that in longtable. Prevent longtable patching by hyperref until hyperref does so automatically:

```
\def\hyper@nopatch@longtable{}
```

Should there be a module?

```
⟨latexrelease⟩\NewModuleRelease{2024/06/01}{lttagging}
⟨latexrelease⟩                 {Tagging support}
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{lttagging}%
⟨latexrelease⟩                 {Undo tagging support}
⟨latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩
⟨latexrelease⟩\EndModuleRelease
⟨/2ekernel | latexrelease⟩
```

File 56
lthyphen.dtx

This file contains the code for loading hyphenation patterns into LaTeX. Most of this will end up in a file called `hyphen.ltx`. If you wish to customize your LaTeX system in respect of hyphenation patterns, write a file `hyphen.cfg`. If this file exists, it will be loaded instead of `hyphen.ltx`. See the comments below for additional information.

To produce the printed version of this file the following code is used. It can be extracted with the DOCSTRIP program, or one can run this file directly through LaTeX 2_ε.

```
⟨*driver⟩
\documentclass{ltxdoc}
\begin{document}
\DocInput{lthyphen.dtx}
\end{document}
⟨/driver⟩
```

The default file `hyphen.ltx` loads hyphenation patterns for US english. If you want to load additional or other hyphenation patterns, you should create a file `hyphen.cfg`. This is best done by starting from `hyphen.ltx`.

For backward compatibility, the default file, `hyphen.ltx`, first tries to load the file `hyphen.tex`. If this file exists, an information message is issued and the appropriate defaults for TeX's internal parameters are set: `\language` is initialized to 0, and `\lefthyphenmin` and `\righthyphenmin` to 2 and 3, respectively, to disallow x- or -xx breaks.

```
⟨*default⟩
\InputIfFileExists{hyphen.tex}%
   {\message{Loading hyphenation patterns for US english.}%
    \language=0
    \lefthyphenmin=2 \righthyphenmin=3 }%
```

Otherwise, since we cannot do anything without any hyphenation patterns, an error message is printed and the IniTeX run is terminated by invoking `\@@end` (which is the LaTeX 2_ε name for TeX's `\end` primitive).

```
   {\errhelp{The configuration for hyphenation is incorrectly
             installed.^^J%
             If you don't understand this error message you need
             to seek^^Jexpert advice.}%
    \errmessage{OOPS! I can't find any hyphenation patterns for
                US english.^^J \space Think of getting some or the
                latex2e setup will never succeed}\@@end}
⟨/default⟩
```

The following example describes the possible contents of a file `hyphen.cfg` that will load both US English and German hyphenation patterns, making the former the default. It sets `\language` to 0 for the US patterns and to 1 for the German patterns. Then `\language` is set to 0 to make this the default and the default values of `\lefthyphenmin` and `\righthyphenmin` are set.

```
language=0
input hyphen % (or \input ushyphen1 if the file has been renamed)
language=1
input ghyph31
```

```
language=0
lefthyphenmin=2
righthyphenmin=3
endinput
```

Another possibility is to use the package `babel`, by Johannes Braams. That package is distributed with a suitable `hyphen.cfg` file.

File 57
ltfinal.dtx

1 Final settings

This section contains the final settings for LaTeX. It initializes some debugging and typesetting parameters, sets the default \catcodes and uc/lc codes, and inputs the hyphenation file.

1.1 Debugging

By default, LaTeX shows statistics:

```
⟨*2ekernel⟩
\tracingstats1
```

1.2 Typesetting parameters

\@lowpenalty
\@medpenalty
\@highpenalty

These are penalties used internally.

```
\newcount\@lowpenalty
\newcount\@medpenalty
\newcount\@highpenalty
```

(*End of definition for* \@lowpenalty*,* \@medpenalty*, and* \@highpenalty*.*)

\newmarks

Allocate extended marks types if etex is active. Placed here at the end of the format to increase compatibility with count allocations in earlier releases.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩                  {\newmarks}{Extended Allocation}%
\ifx\marks\@undefined\else
\def\newmarks{%
  \e@alloc\marks \e@alloc@chardef{\count256}\m@ne\e@alloc@top}
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\newmarks}{Extended Allocation}%
⟨latexrelease⟩\let\newmarks\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* \newmarks*.*)

Allocate 3 mark classes to be used in \markboth and \markright. Should be done earlier but for that definition of \newmarks needs moving (which it should I guess).

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
⟨latexrelease⟩                  {2e-left}{Delayed legacy marks}%
\NewMarkClass {2e-left}
```

```
\NewMarkClass {2e-right}
\NewMarkClass {2e-right-nonempty}
```

No rollback really, the marks will remain.

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {2e-left}{Delayed legacy marks}%
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

\newXeTeXintercharclass
\xe@alloc@intercharclass
\e@alloc@intercharclass@top

Allocate \XeTeXintercharclass types if xetex is active. previously defined in xetex.ini.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩               {\newXeTeXintercharclass}{Extended Allocation}%
```

Classes allocated 1 to 4094 (or 254 on older xetex) (In earlier XeLaTeX versions 1, 2 and 3 were pre-set for CJK).

```
\ifx\XeTeXcharclass\@undefined
\else
```

```
\ifdim\the\XeTeXversion\XeTeXrevision\p@>0.99993\p@
  \chardef\e@alloc@intercharclass@top=4095
\else
  \chardef\e@alloc@intercharclass@top=255
\fi
```

```
\def\newXeTeXintercharclass{%
 \e@alloc\XeTeXcharclass
   \chardef\xe@alloc@intercharclass\m@ne\e@alloc@intercharclass@top}
\fi
```

```
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩               {\newXeTeXintercharclass}{Extended Allocation}%
⟨latexrelease⟩ \ifx\XeTeXcharclass\@undefined
⟨latexrelease⟩ \else
⟨latexrelease⟩    \def\xe@alloc@#1#2#3#4#5{\global\advance#1\@ne
⟨latexrelease⟩     \xe@ch@ck#1#4#2%
⟨latexrelease⟩     \allocationnumber#1%
⟨latexrelease⟩     \global#3#5\allocationnumber
⟨latexrelease⟩     \wlog{\string#5=\string#2\the\allocationnumber}}
⟨latexrelease⟩    \def\xe@ch@ck#1#2#3{%
⟨latexrelease⟩     \ifnum#1<#2\else
⟨latexrelease⟩      \errmessage{No room for a new #3}%
⟨latexrelease⟩     \fi}
⟨latexrelease⟩    \def\newXeTeXintercharclass{%
⟨latexrelease⟩     \xe@alloc@\xe@alloc@intercharclass
⟨latexrelease⟩                     \XeTeXcharclass\chardef\@cclv}
⟨latexrelease⟩ \fi
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2016/02/01}%
```

```
⟨latexrelease⟩  {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
\ifx\XeTeXcharclass\@undefined
\else
  \countdef\xe@alloc@intercharclass=257
  \xe@alloc@intercharclass=\z@
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2015/01/01}%
⟨latexrelease⟩  {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
⟨latexrelease⟩ \ifx\XeTeXcharclass\@undefined
⟨latexrelease⟩ \else
⟨latexrelease⟩   \xe@alloc@intercharclass=\thr@@
⟨latexrelease⟩ \fi
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩  {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
⟨latexrelease⟩ \ifx\XeTeXcharclass\@undefined
⟨latexrelease⟩ \else
⟨latexrelease⟩   \newcount\xe@alloc@intercharclass
⟨latexrelease⟩   \xe@alloc@intercharclass=\thr@@
⟨latexrelease⟩ \fi
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\newXeTeXintercharclass`, `\xe@alloc@intercharclass`, *and* `\e@alloc@intercharclass@top`.)

`trace_stack_levels` Now define the Lua function to emulate `\tracingstacklevels` and install it in the `input_level_string` callback.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
```

In `latexrelease` mode we always remove the function from the callback, then add the correct version later.

```
⟨latexrelease⟩\ifx\directlua\@undefined
⟨latexrelease⟩\else
⟨latexrelease⟩  \directlua{%
⟨latexrelease⟩    if luatexbase.callbacktypes['input_level_string'] and %
⟨latexrelease⟩       luatexbase.in_callback('input_level_string','tracingstacklevels') then
⟨latexrelease⟩        luatexbase.remove_from_callback('input_level_string','tracingstacklevels')
⟨latexrelease⟩    end}%
⟨latexrelease⟩\fi
⟨latexrelease⟩\IncludeInRelease{2021/06/01}{trace_stack_levels}%
⟨latexrelease⟩                 {Lua trace_stack_levels function}%
\ifx\directlua\@undefined
\else
⟨*2ekernel⟩
  \expanded{%
    \everyjob{\the\everyjob
    \noexpand%\directlua
⟨/2ekernel⟩
    \directlua{%
      local function trace_stack_levels (input_ptr)
        local tracingstacklevels = tex.count.tracingstacklevels
```

```
        if tex.tracingmacros > 0 or input_ptr < tracingstacklevels then
          if tracingstacklevels > 0 then
            if input_ptr < tracingstacklevels then
              return "\string\n\string~" .. string.rep(".",input_ptr)
            else
              return "\string~\string~"
            end
          else
            return "\string\n"
          end
        else
          return ""
        end
      end
⟨latexrelease⟩    if luatexbase.callbacktypes['input_level_string'] then
      luatexbase.add_to_callback('input_level_string',
        trace_stack_levels,'tracingstacklevels')
⟨latexrelease⟩    end
    }%
⟨*2ekernel⟩
  }}%
⟨/2ekernel⟩
\fi
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩
```

Then for the full rollback, just do nothing, since the function was already taken out of the rollback above.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}{trace_stack_levels}%
⟨latexrelease⟩                 {Lua trace_stack_levels function}%
⟨latexrelease⟩% Nothing here
⟨latexrelease⟩\EndIncludeInRelease
⟨/2ekernel | latexrelease⟩
⟨*2ekernel⟩
```

(*End of definition for* `trace_stack_levels`.)

The default values of the picture and `\fbox` parameters:

```
\unitlength = 1pt
\fboxsep = 3pt
\fboxrule = .4pt
```

The saved value of TEX's `\maxdepth`:

```
\@maxdepth       = \maxdepth
```

`\vsize` initialized because a `\clearpage` with `\vsize < \topskip` causes trouble. `\@colroom` and `\@colht` also initialized because `\vsize` may be set to them if a `\clearpage` is done before the `\begin{document}`

```
\vsize = 1000pt
\@colroom = \vsize
\@colht = \vsize
```

Initialise `\textheight` `\textwidth` and page style, to avoid internal errors if they are not set by the class.

```
\textheight=.5\maxdimen
\textwidth=\textheight
\ps@empty
```

1.3 Lccodes for hyphenation

For 7- and 8-bit engines the assumption of T1 encodings is the basis for the hyphenation patterns. That's not the case for the Unicode engines, where the assumption is engine-native working. The common loader system provides access to data from the Unicode Consortium covering not only `\lccode` but also other related data. The `\lccode` part of that at least needs to be loaded before hyphenation is tackled: XeTEX follows the standard TEX route of building patterns into the format. LuaTEX doesn't require this data be loaded *here* but it does need to be loaded somewhere. Rather than test for the Unicode engines by name, the approach here is to look for the extended math mode handling both provide: any other engine developed in this area will presumably also provide `\Umathcode`.

```
\ifnum 0%
  \ifx\Umathcode\@undefined\else 1\fi
  \ifx\XeTeXmathcode\@undefined\else 1\fi
  >\z@
  \message{ Unicode character data,}
  \input{load-unicode-data}
⟨/2ekernel⟩
⟨latexrelease⟩\IncludeInRelease{2016/02/01}%
⟨latexrelease⟩  {\XeTeXintercharclasses}{XeTeX character classes}%
⟨latexrelease⟩  \ifx\XeTeXinterchartoks\undefined
⟨latexrelease⟩  \else
⟨latexrelease⟩    \begingroup
⟨latexrelease⟩      \chardef\XeTeXcharclassID = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassOP = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassCL = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassEX = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassIS = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassNS = 0 %
⟨latexrelease⟩      \chardef\XeTeXcharclassCM = 0 %
⟨latexrelease⟩      \input{load-unicode-xetex-classes}
⟨latexrelease⟩    \endgroup
⟨latexrelease⟩    \global\let\xtxHanGlue\undefined
⟨latexrelease⟩    \global\let\xtxHanSpace\undefined
⟨latexrelease⟩    \global\XeTeXinterchartoks 0 1 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 0 2 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 0 3 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 1 0 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 2 0 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 3 0 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 1 1 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 1 2 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 1 3 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 2 1 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 2 2 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 2 3 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 3 1 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 3 2 = {}
⟨latexrelease⟩    \global\XeTeXinterchartoks 3 3 = {}
⟨latexrelease⟩  \fi
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
```

```
⟨latexrelease⟩  {\XeTeXintercharclasses}{XeTeX character classes}%
⟨latexrelease⟩  \ifx\XeTeXinterchartoks\undefined
⟨latexrelease⟩  \else
⟨latexrelease⟩   \input{load-unicode-xetex-classes}
⟨latexrelease⟩   \gdef\xtxHanGlue{\hskip0pt plus 0.1em\relax}
⟨latexrelease⟩   \gdef\xtxHanSpace{\hskip0.2em plus 0.2em minus 0.1em\relax}
⟨latexrelease⟩   \global\XeTeXinterchartoks 0 1 = {\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 0 2 = {\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 0 3 = {\nobreak\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 1 0 = {\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 2 0 = {\nobreak\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 3 0 = {\xtxHanSpace}
⟨latexrelease⟩   \global\XeTeXinterchartoks 1 1 = {\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 1 2 = {\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 1 3 = {\nobreak\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 2 1 = {\nobreak\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 2 2 = {\nobreak\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 2 3 = {\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 3 1 = {\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 3 2 = {\xtxHanGlue}
⟨latexrelease⟩   \global\XeTeXinterchartoks 3 3 = {\nobreak\xtxHanGlue}
⟨latexrelease⟩  \fi
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

There is one over-ride that makes sense here (see below for the same for 8-bit engines): setting the lccode for `-` to itself.

```
  \lccode`\- =`\- % default hyphen char
```

The alternative is that a "traditional" engine is in use.

```
\else
```

We set things up so that hyphenation files can assume that the default (T1) lccodes are in use (at present this also sets up the uccodes). We temporarily define `\reserved@a` to apply `\reserved@c` to all the numbers in the range of its arguments.

```
\def\reserved@a#1#2{%
   \@tempcnta#1\relax
   \@tempcntb#2\relax
   \reserved@b
}
\def\reserved@b{%
   \ifnum\@tempcnta>\@tempcntb\else
      \reserved@c\@tempcnta
      \advance\@tempcnta\@ne
      \expandafter\reserved@b
   \fi
}
```

Depending on the TEX version, we might not be allowed to do this for non-ASCII characters.

```
\def\reserved@c#1{%
   \count@=#1\advance\count@ by -"20
   \uccode#1=\count@
   \lccode#1=#1
}
```

```
\reserved@a{`\a}{`\z}
\reserved@a{"A0}{"BC}
\reserved@a{"E0}{"FF}
```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```
\def\reserved@c#1{%
   \count@=#1\advance\count@ by "20
   \uccode#1=#1
   \lccode#1=\count@
   \sfcode#1=999
}
\reserved@a{`\A}{`\Z}
\reserved@a{"80}{"9C}
\reserved@a{"C0}{"DF}
```

Well, it would be nice if that were correct, but unfortunately, the Cork encoding contains some odd slots whose uccode or lccode isn't quite what you'd expect.

```
\uccode`\^^Y=`\I     % dotless i
\lccode`\^^Y=`\^^Y   % dotless i
\uccode`\^^Z=`\J     % dotless j, ae in OT1
\lccode`\^^Z=`\^^Z   % dotless j, ae in OT1
\lccode`\^^9d=`\i    % dotted I
\uccode`\^^9d=`\^^9d % dotted I
\lccode`\^^9e=`\^^9e % d-bar
\uccode`\^^9e=`\^^d0 % d-bar
```

Finally here is one that helps hyphenation in the OT1 encoding.

```
\lccode`\^^[=`\^^[   % oe in OT1
```

And we also set the `\lccode` of `\-` and `\textcompwordmark` so that they do not prevent hyphenation in the remainder of the word (as suggested by Lars Helström).

```
\lccode`\- =`\-   % default hyphen char
\lccode 127=127   % alternate hyphen char
\lccode 23 =23    % textcompwordmark in T1
```

End of the conditional to select either Unicode or T1 encoding defaults.

```
\fi
```

At this stage, we can install any last-minute expl3 set-up.

```
\@expl@finalise@setup@@
\def\@expl@finalise@setup@@{}
```

This is as good a place as any to active a few XeTEX-specific settings

```
\ifx\XeTeXuseglyphmetrics\@undefined
\else
  \XeTeXuseglyphmetrics=1 %
  \XeTeXdashbreakstate=1 %
\fi
```

1.4 Hyphenation

The following code will be compiled into the format file. It checks for the existence of `hyphen.cfg` in inputs that file if found. Otherwise it inputs `hyphen.ltx`. Note that these are loaded in *before* the `\catcode`s are set, so local hyphenation files can use 8-bit input.

We try to load the customized hyphenation description file.

```
\InputIfFileExists{hyphen.cfg}
          {\typeout{=========================================^^J%
                     Local configuration file hyphen.cfg used^^J%
                    ==========================================}%
            \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
          }
          {\input{hyphen.ltx}}
\let\@addtofilelist\@gobble
```

`\l@nohyphenation`

```
\ifx\l@nohyphenation \@undefined
  \newlanguage\l@nohyphenation
\fi
```

(*End of definition for* `\l@nohyphenation`.)

`\document@default@language` Default document language. -1 acts as language 0, but used as a flag in `\document` to see if it has been set in the preamble.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2017/04/15}%
⟨latexrelease⟩       {\document@default@language}{Save language for hyphenation}%
\let\document@default@language\m@ne
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩       {\document@default@language}{Save language for hyphenation}%
%
⟨latexrelease⟩\let\document@default@language\@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\document@default@language`.)

1.5 Font loading

Fonts loaded during the formatting process might already have changed the `\font@submax` from `0pt` to something higher. If so, we put out a bold warning.

```
\ifdim \font@submax >\z@
   \@font@warning{Size substitutions with differences\MessageBreak
                  up to \font@submax\space have occurred.\MessageBreak
                 \MessageBreak
                 Please check the transcript file
                 carefully\MessageBreak
                 and redo the format generation if necessary!
                 \@gobbletwo}%
   \errhelp{Only stopped, to give you time to
            read the above message.}
   \errmessage{}
```

We reset the macro. Otherwise every user will get a warning on every job.

```
\def\font@submax{0pt}
\fi
```

For pdfTEX preload and enable automatic glyph to Unicode mapping for more reliable copy and paste support.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
⟨latexrelease⟩                  {\pdfgentounicode}{Preload glyphtounicode}%
\ifx \pdfgentounicode \@undefined \else
⟨*2ekernel⟩
  \ifnum 0=0%
    \ifdefined\pdftexversion
% \pdftexversion<140 does not have \pdfgentounicode, so we only check higher values
      \ifnum \pdftexversion=140 \ifnum\pdftexrevision<22 1\fi\fi
    \fi
    \relax
⟨/2ekernel⟩
    \input glyphtounicode
⟨*2ekernel⟩
  \else
    \begingroup
      \everyeof{\noexpand}\endlinechar-1
      \edef\x{\endgroup
        \everyjob{\the\everyjob\@@input glyphtounicode }%
    }\x
  \fi
⟨/2ekernel⟩
  \pdfgentounicode=1
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

When rolling back we can't unload the glyphtounicode mappings, but we can reset `\pdfgentounicode` to ensure that they aren't used.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\pdfgentounicode}{Preload glyphtounicode}%
⟨latexrelease⟩\ifx \pdfgentounicode \@undefined \else
⟨latexrelease⟩  \pdfgentounicode=0
⟨latexrelease⟩\fi
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

1.6 Input encoding

Starting with the 2018 LATEX release default the inputencoding to UTF-8. Unless the format is being used with luatex, xetex, enctex or mltex.

This is done in a way largely compatible with older releases: `utf8.def` is input just as if

`\usepackage[utf8]{inputenc}`

had been used, however rather than input the whole package a minimal core part just enough to support loading the UTF-8 encoding files is defined here.

If a document re-specifies UTF-8 this is silently ignored.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
```

Check that a classic 8-bit tex engine is being used (LaTeX or PDFLaTeX).

```
⟨latexrelease⟩\IncludeInRelease{2018/04/01}%
⟨latexrelease⟩                  {\UTFviii@invalid}{UTF-8 default}%
```

Skip this section in Unicode TeX, or if MLTeX and EncTeX are enabled.

```
\ifnum0%
  \ifx\Umathcode\@undefined\else 1\fi
  \ifx\mubyte\@undefined\else 1\fi
  \ifx\charsubdef\@undefined\else 1\fi
  =\z@
```

```
\def\saved@space@catcode{10}
\let\@inpenc@test\relax
\def\IeC{%
  \ifx\protect\@typeset@protect
    \expandafter\@firstofone
  \else
    \noexpand\IeC
  \fi
}
```

Make characters active for UTF-8 input formats

```
\@tempcnta=1
\loop
  \catcode\@tempcnta=13  %
  \advance\@tempcnta\@ne %
\ifnum\@tempcnta<32      %
\repeat                  %
\catcode0=15  % null
\catcode9=10  % tab
\catcode10=12 % ctrl J
\catcode12=13 % ctrl L
\catcode13=5  % newline
\@tempcnta=128
\loop
  \catcode\@tempcnta=13
  \advance\@tempcnta\@ne
\ifnum\@tempcnta<256
\repeat
```

`\UseRawInputEncoding` Reset 8 bit characters to catcode 12 so the input encoding matches the "Raw" font encoding. Useful for special behaviours, or for compatibility with older LaTeX formats.

```
\def\UseRawInputEncoding{%
\let\inputencodingname\@undefined                      % revert
\let\DeclareFontEncoding@\DeclareFontEncoding@saved    % revert
\let\DeclareUnicodeCharacter\@undefined                % revert
\@tempcnta=1
\loop
  \catcode\@tempcnta=15  %
  \advance\@tempcnta\@ne %
\ifnum\@tempcnta<32      %
\repeat                  %
\catcode0=15  % null
\catcode9=10  % tab
```

```
\catcode10=12 % ctrl J
\catcode12=13 % ctrl L
\catcode13=5  % newline
\@tempcnta=128
\loop
  \catcode\@tempcnta=12
  \advance\@tempcnta\@ne
\ifnum\@tempcnta<256
\repeat
}
```

(*End of definition for* `\UseRawInputEncoding`.)

`\DeclareFontEncoding@saved` Saved version of `\DeclareFontEncoding@` before `utf8.def` modifies it for use in `\UseRawInputEncoding` above.

```
\let\DeclareFontEncoding@saved\DeclareFontEncoding@
```

(*End of definition for* `\DeclareFontEncoding@saved`.)

```
\edef\inputencodingname{utf8}%
\input{utf8.def}
\let\UTFviii@undefined@err@@\UTFviii@undefined@err
\let\UTFviii@invalid@err@@\UTFviii@invalid@err
\let\UTFviii@two@octets@@\UTFviii@two@octets
\let\UTFviii@three@octets@@\UTFviii@three@octets
\let\UTFviii@four@octets@@\UTFviii@four@octets
⟨2ekernel⟩\def\UTFviii@undefined@err#1{\@gobble#1}%
⟨2ekernel⟩\let\UTFviii@invalid@err\string
⟨2ekernel⟩\let\UTFviii@two@octets\string
⟨2ekernel⟩\let\UTFviii@three@octets\string
⟨2ekernel⟩\let\UTFviii@four@octets\string
⟨2ekernel⟩\everyjob\expandafter{\the\everyjob
⟨2ekernel⟩\let\UTFviii@undefined@err\UTFviii@undefined@err@@
⟨2ekernel⟩\let\UTFviii@invalid@err\UTFviii@invalid@err@@
⟨2ekernel⟩\let\UTFviii@two@octets\UTFviii@two@octets@@
⟨2ekernel⟩\let\UTFviii@three@octets\UTFviii@three@octets@@
⟨2ekernel⟩\let\UTFviii@four@octets\UTFviii@four@octets@@
⟨2ekernel⟩}
\let\@inpenc@test\@undefined
\let\saved@space@catcode\@undefined
```

For formats not set up for UTF-8 default, set the C0 controls to catcode 15.

```
\else
\@tempcnta=0
\loop
  \catcode\@tempcnta=15  %
  \advance\@tempcnta\@ne %
\ifnum\@tempcnta<32      %
\repeat                  %
\catcode0=15  % null
\catcode9=10  % tab
\catcode10=12 % ctrl J
\catcode12=13 % ctrl L
\catcode13=5  % newline
```

```
\let\UseRawInputEncoding\relax
```

This ends the skipped code in Unicode engines:

```
\fi
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\UTFviii@invalid}{UTF-8 default}%
```

The first block of commands got only introduced in 2019 but we revert all of Unicode support in one go not jump to the intermediate version.

```
⟨latexrelease⟩  \let\UTFviii@two@octets@combine\@undefined
⟨latexrelease⟩  \let\UTFviii@three@octets@combine\@undefined
⟨latexrelease⟩  \let\UTFviii@four@octets@combine\@undefined
⟨latexrelease⟩  \let\UTFviii@two@octets@string\@undefined
⟨latexrelease⟩  \let\UTFviii@three@octets@string\@undefined
⟨latexrelease⟩  \let\UTFviii@four@octets@string\@undefined
⟨latexrelease⟩  \let\UTFviii@two@octets@noexpand\@undefined
⟨latexrelease⟩  \let\UTFviii@three@octets@noexpand\@undefined
⟨latexrelease⟩  \let\UTFviii@four@octets@noexpand\@undefined
```

```
⟨latexrelease⟩\@tempcnta=0
⟨latexrelease⟩\loop
⟨latexrelease⟩  \catcode\@tempcnta=15
⟨latexrelease⟩  \advance\@tempcnta\@ne
⟨latexrelease⟩\ifnum\@tempcnta<32
⟨latexrelease⟩\repeat       %
⟨latexrelease⟩\catcode9=10  % tab
⟨latexrelease⟩\catcode10=12 % ctrl J
⟨latexrelease⟩\catcode12=13 % ctrl L
⟨latexrelease⟩\catcode13=5  % newline
⟨latexrelease⟩\@tempcnta=128
⟨latexrelease⟩\loop
⟨latexrelease⟩\catcode\@tempcnta=12
⟨latexrelease⟩\advance\@tempcnta\@ne
⟨latexrelease⟩\ifnum\@tempcnta<256
⟨latexrelease⟩\repeat
⟨latexrelease⟩\let\IeC\@undefined
⟨latexrelease⟩\def\DeclareFontEncoding@#1#2#3{%
⟨latexrelease⟩  \expandafter
⟨latexrelease⟩  \ifx\csname T@#1\endcsname\relax
⟨latexrelease⟩     \def\cdp@elt{\noexpand\cdp@elt}%
⟨latexrelease⟩     \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
⟨latexrelease⟩                    {\default@family}{\default@series}%
⟨latexrelease⟩                    {\default@shape}}%
⟨latexrelease⟩     \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
⟨latexrelease⟩  \else
⟨latexrelease⟩     \@font@info{Redeclaring font encoding #1}%
⟨latexrelease⟩  \fi
⟨latexrelease⟩  \global\@namedef{T@#1}{#2}%
⟨latexrelease⟩  \global\@namedef{M@#1}{\default@M#3}%
⟨latexrelease⟩  \xdef\LastDeclaredEncoding{#1}%
⟨latexrelease⟩  }
⟨latexrelease⟩  \let\UseRawInputEncoding\@undefined
⟨latexrelease⟩  \let\DeclareFontEncoding@saved\@undefined
⟨latexrelease⟩  \let\inputencodingname\@undefined
⟨latexrelease⟩\EndIncludeInRelease
```

⟨*2ekernel⟩

We temporarily define \reserved@a to apply \reserved@c to all the numbers in the range of its arguments.

```
\def\reserved@a#1#2{%
   \@tempcnta#1\relax
   \@tempcntb#2\relax
   \reserved@b
}
\def\reserved@b{%
   \ifnum\@tempcnta>\@tempcntb\else
      \reserved@c\@tempcnta
      \advance\@tempcnta\@ne
      \expandafter\reserved@b
   \fi
}
```

Set the special catcodes (although some of these are useless, since an error will have occurred if the catcodes have changed). Note that ^^J has catcode 'other' for use in warning messages.

```
\catcode`\ =10
\catcode`\#=6
\catcode`\$=3
\catcode`\%=14
\catcode`\&=4
\catcode`\\=0
\catcode`\^=7
\catcode`\_=8
\catcode`\{=1
\catcode`\}=2
\catcode`\~=13
\catcode`\@=11
\catcode`\^^I=10
\catcode`\^^J=12
\catcode`\^^L=13
\catcode`\^^M=5
```

Set the 'other' catcodes.

```
\def\reserved@c#1{\catcode#1=12\relax}
\reserved@c{`\!}
\reserved@c{`\"}
\reserved@a{`\'}{`\?}
\reserved@c{`\[}
\reserved@c{`\]}
\reserved@c{`\`}
\reserved@c{`\|}
```

Set the 'letter' catcodes.

```
\def\reserved@c#1{\catcode#1=11\relax}
\reserved@a{`\A}{`\Z}
\reserved@a{`\a}{`\z}
```

All the characters in the range 0–31 and 127–255 are illegal, *except* tab (^^I), nl (^^J), ff (^^L) and cr (^^M).

1.7 Lccodes and uccodes

We now again set up the default (T1) uc/lccodes. The lower case characters need their `\uccode` and `\lccode` values set. Some of this is a repeat of the set-up before loading hyphenation files. Depending on the TEX version, we might not be allowed to do this for non-ASCII characters. For the Unicode engines (XeTEX and LuaTEX) there is no need to do any of this: they use hyphenation data which does not alter any of the set up and so this entire block is skipped.

```
\ifnum 0%
  \ifx\Umathcode\@undefined\else 1\fi
  \ifx\XeTeXmathcode\@undefined\else 1\fi
  >\z@
\else
\def\reserved@c#1{%
   \count@=#1\advance\count@ by -"20
   \uccode#1=\count@
   \lccode#1=#1
}
\reserved@a{`\a}{`\z}
\reserved@a{"A0}{"BC}
\reserved@a{"E0}{"FF}
```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```
\def\reserved@c#1{%
   \count@=#1\advance\count@ by "20
   \uccode#1=#1
   \lccode#1=\count@
   \sfcode#1=999
}
\reserved@a{`\A}{`\Z}
\reserved@a{"80}{"9C}
\reserved@a{"C0}{"DF}
```

Well, it would be nice if that were correct, but unfortunately, the Cork encoding contains some odd slots whose uccode or lccode isn't quite what you'd expect.

```
\uccode`\^^Y=`\I     % dotless i
\lccode`\^^Y=`\^^Y   % dotless i
\uccode`\^^Z=`\J     % dotless j, ae in OT1
\lccode`\^^Z=`\^^Z   % dotless j, ae in OT1
\lccode`\^^9d=`\i    % dotted I
\uccode`\^^9d=`\^^9d % dotted I
\lccode`\^^9e=`\^^9e % d-bar
\uccode`\^^9e=`\^^d0 % d-bar
```

Finally here is one that helps hyphenation in the OT1 encoding.

```
\lccode`\^^[=`\^^[   % oe in OT1
\fi % End of reset block for 8-bit engines
```

`\BCPdata` A stub for use by babel, polyglossia, etc.

```
\ExplSyntaxOn
\newcommand*\BCPdata[1]{
  \str_case:nn {#1}
    {
      { language } { en }
```

```
      { region }   { US }
      { script }   { Latn }
      { tag }      { en-US }
    }
}
\ExplSyntaxOff
```

(*End of definition for* \BCPdata.)

\MakeUppercase
\MakeLowercase
\MakeTitlecase
\NoCaseChange
\AddToNoCaseChangeList
\CaseSwitch
\DeclareCaseChangeEquivalent
\DeclareLowercaseMapping
\DeclareTitlecaseMapping
\DeclareUppercaseMapping
\@uclclist

And whilst we're doing things with uc/lc tables, here are two commands to upper- and lower-case a string.

Wrappers around the L3 case changing functions. \protected to make them mostly safe as replacements for uppercase and \lowercase.

In

```
\markboth{\MakeUppercase\contentsname}
         {\MakeUppercase\contentsname}
```

then the uppercasing is only done to the first letter of the contents name, since the mark expands out to:

```
\mark{\MakeUppercase Table of Contents}
     {\MakeUppercase Table of Contents}
```

In order to get round this, we redefine \MakeUppercase and \MakeLowercase to grab their argument and brace it.

Earlier versions needed to process \@uclclist in an \edef to handle legacy input encodings, but recent (2022) expl3 versions handle non-UTF8 text natively so we simply call the \text_...case:n functions.

```
\ExplSyntaxOn
\keys_define:nn { __kernel }
  {
    lang   .str_set:N = \reserved@a ,
    locale .str_set:N = \reserved@a
  }
\cs_new_protected:Npn \@@text@case@aux #1#2#3
  {
    \cs_set_nopar:Npn \reserved@a { }
    \tl_if_blank:nTF {#2}
      {
        \str_set:Nx \reserved@a
          { \BCPdata { casing } }
        \str_if_empty:NT \reserved@a
          {
            \str_set:Nx \reserved@a
              { \BCPdata { language } }
          }
      }
      { \keys_set:nn { __kernel } {#2} }
    \use:c { text_ #1 case:Vn } \reserved@a {#3}
  }
```

The odd use of *three* spaces here is needed as ltcmd uses the name with one and two spaces to give a 'friendly' error message for a runaway argument: that means we can't use it here.

```
\exp_args_generate:n { cnx }
\cs_set_protected:Npn \reserved@a #1
  {
    \cs_generate_variant:cn { text_ \str_lowercase:n {#1} case:nn } { V }
    \ExpandArgs { cnx } \NewExpandableDocumentCommand
       { Make#1case }
       { O{} +m }
       { \exp_not:c { Make#1case \c_space_tl \c_space_tl \c_space_tl } [####1] {####2} }
  }
\reserved@a { Upper }
\reserved@a { Lower }
\reserved@a { Title }
```

Currently, babel uses the equivalence of `\oe` and `\OE` to force casing of some material, most notably in `\today`. To enable that to work, we have to set those commands equal even though the current case changing code does not work using this approach.

```
\cs_new_protected:cpn { MakeLowercase \c_space_tl \c_space_tl \c_space_tl } [#1] #2
  {{
    \let \OE \oe
    \@@text@case@aux { lower } {#1} {#2}
  }}
\cs_new_protected:cpn { MakeUppercase \c_space_tl \c_space_tl \c_space_tl } [#1] #2
  {{
    \let \oe \OE
    \@@text@case@aux { upper } {#1} {#2}
  }}
\cs_new_protected:cpn { MakeTitlecase \c_space_tl \c_space_tl \c_space_tl } [#1] #2
  {{
    \let \oe \OE
    \@@text@case@aux { title } {#1} {#2}
  }}
```

`\NoCaseChange` protects its argument from the case change functions.

`\AddToNoCaseChangeList` Allows new commands to protect their arguments, eg `AddToNoCaseChangeList{\eqref}` would protect the argument of `\eqref` in the same way as the argument of `\ref`.

```
\cs_new_protected_nopar:Npn\AddToNoCaseChangeList
         {\tl_put_right:Nn \l_text_case_exclude_arg_tl}
\AddToNoCaseChangeList{ \NoCaseChange }
\cs_new_protected:Npn \NoCaseChange #1 {#1}
\cs_new_eq:NN \CaseSwitch \text_case_switch:nnnn
\cs_new_eq:NN \DeclareCaseChangeEquivalent
  \text_declare_case_equivalent:Nn
\NewDocumentCommand \DeclareLowercaseMapping { o m m }
  {
    \IfNoValueTF {#1}
      { \text_declare_lowercase_mapping:nn }
      { \text_declare_lowercase_mapping:nnn {#1} }
        {#2} {#3}
  }
\NewDocumentCommand \DeclareTitlecaseMapping { o m m }
```

```
  {
    \IfNoValueTF {#1}
      { \text_declare_titlecase_mapping:nn }
      { \text_declare_titlecase_mapping:nnn {#1} }
        {#2} {#3}
  }
\NewDocumentCommand \DeclareUppercaseMapping { o m m }
  {
    \IfNoValueTF {#1}
      { \text_declare_uppercase_mapping:nn }
      { \text_declare_uppercase_mapping:nnn {#1} }
        {#2} {#3}
  }
\ExplSyntaxOff
```

```
\def\@uclclist{\oe\OE\o\O\ae\AE
      \dh\DH\dj\DJ\l\L\ng\NG\ss\SS\ij\IJ\th\TH}
```

(*End of definition for* `\MakeUppercase` *and others.*)

1.8 Applying Patch files

Between major releases, small patches will be distributed in files `ltpatch.ltx` which must be added at this point.

Patch file code removed.

```
%\IfFileExists{ltpatch.ltx}
%  {\typeout{=================================^^J%
%             Applying patch file ltpatch.ltx^^J%
%            =================================}
%   \def\fmtversion@topatch{unknown}
%   \input{ltpatch.ltx}
%   \ifx\fmtversion\fmtversion@topatch
%      \ifx\patch@level\@undefined
%        \typeout{^^J^^J^^J%
%         !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
%         !! Patch file `ltpatch.ltx' not suitable for this^^J%
%         !! version of LaTeX.^^J^^J%
%         !! Please check if initex found an old patch file:^^J%
%         !! --- if so, rename it or delete it, and redo the^^J%
%         !! initex run.^^J%
%         !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J}%
%        \batchmode \@@end
%      \else
```

The code below adds the 'patch level' string to the first `\typeout` in the startup banner.

```
%        \def\fmtversion@topatch{0}%
%        \ifx\fmtversion@topatch\patch@level\else
%          \def\reserved@a\typeout##1##2\reserved@a{%
%                 \typeout{##1 patch level \patch@level}##2}
%          \everyjob\expandafter\expandafter\expandafter{%
%             \expandafter\reserved@a\the\everyjob\reserved@a}
%          \let\reserved@a\relax
%          \the\everyjob
%        \fi
```

```
%      \fi
%   \else
%      \typeout{^^J^^J^^J%
%    !!!!
```

\@filelist
\@addtofilelist

Reset \@filelist so files input while making the format are not listed. The list built up so far may take up a lot of memory and so it is moved to \reserved@a where it will be overwritten as soon as almost any LaTeX command is issued in a class file. However the latexbug.tex program will be able to access this information and insert it into a bug report.

```
\let\@filelist\@gobble
\def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}%
```

(*End of definition for* \@filelist *and* \@addtofilelist.)

1.11 Preparation for supporting PDF in backends

At the current point in time, basic support for PDF in backends is not part of LaTeX core; it is provided by external packages. At some time in the future that work will be placed into the kernel but for now it is separate and has to be explicitly loaded in the document.

In that code there is a command \IfPDFManagementActiveTF which can be used by packages in order to execute different code depending on the whether this basic backend support is loaded.

To make this also work properly when this external package is not loaded at all, we here add this command already in the kernel (with a trivial definition); thus any package can query this loading state in all circumstances. Once this basic PDF backend support gets moved to the kernel, this definition will vanish again from here or, rather, it will be replaced by a real test.

\IfPDFManagementActiveTF

So long as the code for the basic backend support for PDF is not loaded, the test that is implicit here will always return the false branch. Once this code is loaded, this definition will get replaced by a real test (as it is then possible that the management code is either activated or not activated).

```
\let \IfPDFManagementActiveTF \@secondoftwo
```

(*End of definition for* \IfPDFManagementActiveTF.)

1.12 Do some temporary work for pre-release

This is a good place to load code that hasn't yet been integrated into the other files ...

1.13 Some last minute initializations ...

Load the first aid set of definitions for external packages that await updates.

```
\@input{latex2e-first-aid-for-external-files.ltx}
```

1.14 Dumping the format

Finally we make @ into a letter, ensure the format will be in the 'normal' error mode, and dump everything into the format file.

```
\makeatother
\errorstopmode
\dump
⟨/2ekernel⟩
```

Change History

1985-11-04 ltmath.dtx LaTeX2.09
General: produce warning message if line extends into margin. Doesn't warn about formula overprinting equation number. 795
1989-04-10 ltfssbas.dtx v1.0a
General: Starting with version numbers! `\ifmmode` added in `\math@group` 523
1989-04-10 ltfssbas.dtx v1.0b
General: `\preload@sizes` added. . . 523
`\wrong@fontshape` changed to define substitution font/shape macro. 523
1989-04-10 ltfssini.dtx v1.0a
General: Starting with version numbers `\newif` for `\@tempswa` added since this switch is unknown at the time when this file is read in. (latex.tex is loaded later.) `\math@famname` changed to `\math@version`. 634
1989-04-14 ltfssbas.dtx v1.0c
General: More documentation added. 523
1989-04-15 ltfssini.dtx v1.0b
General: `\mathfontset` renamed to `\mathversion`. 634
1989-04-19 ltfssbas.dtx v1.0d
General: Even more doc. 523
1989-04-21 ltfssbas.dtx v1.0e
General: Documentation is fun! Parameters of `\define@mathalphabet` changed. 523
1989-04-21 ltfssini.dtx v1.0c
General: Changed to conform to fam.tex. 634
1989-04-23 ltfssbas.dtx v1.0f
General: % in `\getanddefinefonts` added. 523
1989-04-26 ltfssini.dtx v1.0d
General: `\xpt` added. 634
1989-04-27 ltfssbas.dtx v1.0g
General: Documentation revised. . . 523
1989-04-27 ltfssini.dtx v1.0e
General: Definitions of LaTeX symbols corrected. 634
1989-04-29 ltfssbas.dtx v1.0h
General: Documented problem with `\halign`, and `\noalign` 523
`\mathversion`: Test if version defined added. 534
1989-04-29 ltfssbas.dtx v1.0i
General: Removed the `\halign` `\noalign` correction (wasn't bugfree) 523
1989-04-29 ltfssini.dtx v1.0f
General: Corrections to LaTeX tabular env. added. 634
1989-05-01 ltfssbas.dtx v1.0j
General: Default for `\baselinestretch` added. 523
1989-05-22 ltfssbas.dtx v1.0k
General: Lines longer than 72 characters folded. 523
1989-05-22 ltfssini.dtx v1.0g
General: Lines shortened to 72 characters 634
1989-09-14 ltfssbas.dtx v1.0m
General: Global replacement: `\group` to `\mathgroup` 523
`\mathversion`: Corrected typo: `\endscname` to `\endcsname`. . . . 534
1989-11-07 ltfssini.dtx v1.0i
General: All family, series, and shape names abbreviated. 634
1989-11-08 ltfssbas.dtx v1.0o
General: First parameter of `\define@mathalphabet` and `\define@mathgroup` changed from string to control sequence. 523
1989-11-14 ltfssbas.dtx v1.0p
`\math@version`: Math version prefix 'mv@' added. 534
1989-11-19 ltfssbas.dtx v1.0q
`\define@newfont`: Group added. . . 537
`\wrong@fontshape`: Instead of calling `\family\default@family`, etc. we directly set `\f@family`, etc. . . . 541
1989-11-22 ltfssbas.dtx v1.0r
`\math@version`: `\def` → `\edef` for `\math@version`. 534
1989-11-25 ltfssbas.dtx v1.0s
General: All `\edef\font@name` changed to `\xdef\font@name`. Necessary after introduction of `\begingroup`/`\endgroup` in v1.0q. 523
extra// → + in `\extra@def`. . . . 523

1989-11-26 ltfssbas.dtx v1.0t
\select@group: \bgroup/\egroup changed to \begingroup/\endgroup to avoid empty Ord atom on math list. . . 544
1989-12-02 ltfssini.dtx v1.1b
General: \rmmath renamed to \mathrm 634
1989-12-03 ltfssini.dtx v1.1c
General: Some internal macros renamed to make them inaccessible. 634
1989-12-05 ltfssbas.dtx v1.0u
\addto@hook: \addto@hook added. . 549
1989-12-05 ltfsstrc.dtx v1.0u fam.dtx
\every@math@size: Hook \every@size added. 581
1989-12-13 ltfsstrc.dtx v1.0f
\use@mathgroup: \expandafter added before final \fi. 584
1989-12-16 ltfssbas.dtx v1.1a
\select@group: \relax in front added. 544
Now four arguments. 544
Redefinition of alphabet now simpler. 544
Usage of '=' macro added. 544
1989-12-16 ltfsstrc.dtx v1.1a
\selectfont: Changed order of calls. 576
\use@mathgroup: Redefinition of alphabet now simpler. 583
Usage of '=' macro added. 583
1990-01-18 ltfsstrc.dtx v1.0h
General: \tracingfonts meaning changed. 572
1990-01-20 ltfssbas.dtx v1.2a
\math@bgroup: Def. placed in this file. 546
\math@egroup: Def. placed in this file. 546
\select@group: Def for alph id changed. 544
1990-01-21 ltfssbas.dtx v1.2b
\select@group: Code moved to \use@mathgroup. 544
1990-01-21 ltfsstrc.dtx v1.2b
\use@mathgroup: Macro added to allow cleaner interface. 583
1990-01-23 ltfssbas.dtx v1.2c
General: \no@version@warning renamed to \no@alphabet@error. 523
Macro \no@alphabet@help added 523
\no@alphabet@error: Changed to error call 523
1990-01-25 ltfssini.dtx v1.1e
\nfss@text: Macro added. 657
1990-01-27 ltfssbas.dtx v1.2d
\DeclarePreloadSizes: Font identifier set to \relax. 529
1990-01-28 ltfssbas.dtx v1.2e
\mathgroup: \newfam let to \new@mathgroup. 523
1990-01-28 ltfssbas.dtx v1.2f
\define@newfont: Added call to \curr@fontshape macro to allow substitution. 537
\wrong@fontshape: Warning message slightly changed. 541
1990-01-28 ltfssini.dtx v1.2b
\em: Call to \@nomath added. 654
1990-02-08 ltfssini.dtx v1.1g
General: Protected the commands \family, \series, \shape, \size, \selectfont, and \mathversion. 634
1990-02-16 ltfssbas.dtx v1.2g
General: Support for changes of \baselineskip without changing the size. 523
\math@version: \@nomath added. . . 534
1990-02-18 ltfsstrc.dtx v1.0j
\selectfont: Redefine unprotected version \p@selectfont instead of \selectfont. 576
1990-03-14 ltfsstrc.dtx v1.0k
General: Added code for TeX3. . . . 572
\extract@font: Added code for TeX3. 575
1990-03-30 ltfssbas.dtx v1.2h
\math@egroup: Changed to have one arg. 546
1990-03-30 ltfsstrc.dtx v1.2h
\use@mathgroup: Third argument removed (see \math@egroup). . . . 583
1990-04-01 ltfssbas.dtx v1.2i
General: Code added from tracefnt.dtx. 523
Support for TeX3. 523
1990-04-01 ltfsstrc.dtx v1.0l
General: Part of code moved to fam.dtx. 572
\tracingfonts: Check if \tracingfonts already defined. . 573
1990-04-01 ltfsstrc.dtx v1.0o
\tracingfonts: Check if \tracingfonts defined removed again. 573

1990-04-02 ltfssini.dtx v1.1i
General: \input of files now handled by docstrip. 634
1990-04-05 ltfsstrc.dtx v1.0m
\selectfont: Call \tracingon only if \tracingfonts greater than 3. . 576
1990-05-05 ltfsstrc.dtx v1.0n
\selectfont: \tracingon with new syntax. 576
1990-06-23 ltfssini.dtx v1.1k
\nfss@text: Changed to \mbox. . . . 657
1990-06-24 ltfssbas.dtx v1.2j
\DeclarePreloadSizes: Missing percent added. 529
1990-06-24 ltfsstrc.dtx v1.0o
\baselinestretch: Moved to tracefnt.dtx. 581
\getanddefine@fonts: \Adding tracing code. 585
\Macro moved from fam.dtx. . . . 584
Adding debug code. 585
\use@mathgroup: Tracing code added. 584
1990-06-30 ltfssbas.dtx v1.2l
\showhyphens: Macro added. 547
1990-06-30 ltfsstrc.dtx v1.0p
\use@mathgroup: Added \relax after math group number. 584
1990-07-07 ltfsstrc.dtx v1.0q
\getanddefine@fonts: Group number added to tracing. 585
\math@egroup: Tracing code added. 584
\use@mathgroup: Group number added to tracing. 584
1990-08-27 ltfsstrc.dtx 1.0r
\type@restoreinfo: Some extra tracing info. 580
1990-08-27 ltfsstrc.dtx v1.0r
\getanddefine@fonts: Correcting missing name after \tracingon. . 585
1991-03-28 ltfssini.dtx v1.1m
\copyright: Extra braces added. . . 657
1991-03-30 ltfssini.dtx v1.2g
\newfont: Definition added. 656
\symbol: Definition added. 656
1991-07-24 ltmiscen.dtx LaTeX2.09
\@verbatim: Added \penalty\interlinepenalty to definition of \par so that \samepage works 778
1991-08-14 ltmath.dtx LaTeX2.09
\cases: (RmS) inserted extra braces around entry for NFSS 789
1991-08-14 ltpictur.dtx LaTeX2.09
General: (RmS) inserted extra braces around entry for NFSS 867
1991-08-14 ltthm.dtx LaTeX2.09
\@endtheorem: Moved \itshape after \item to make it work with NFSS 897
1991-08-26 ltfssini.dtx v1.1n
\reset@font: Macro introduced . . . 657
1991-08-26 ltmiscen.dtx LaTeX2.09
\@verbatim: \@@par added 778
1991-08-26 ltpictur.dtx LaTeX2.09
\endpicture: (RmS & FMi) extra boxing level around \@picbox to guard against unboxing in math mode (proposed by John Hobby) 865
1991-08-26 ltplain.dtx LaTeX209
\tracingall: Added \errorcontextlines=\maxdimen, suggested by J. Schrod 33
1991-09-29 ltboxes.dtx LaTeX2.09
\@mpfootnotetext: (RmS) added \reset@font 832
1991-09-29 ltfloat.dtx LaTeX2.09
\@footnotetext: (RmS) added \reset@font 930
1991-09-29 ltmath.dtx LaTeX2.09
\@eqnnum: RmS: \reset@font added. 794
1991-09-29 ltsect.dtx LaTeX2.09
\@dottedtocline: (RmS) added \reset@font for page number . . 909
1991-10-17 ltcntrl.dtx LaTeX209
\@tfor: (Rms) \xdef replaced by \def (See FMi's array.doc) 387
1991-10-25 ltbibl.dtx LaTeX2.09
\@citex: added \reset@font, suggested by Bernd Raichle. . . . 939
1991-11-01 ltfloat.dtx LaTeX2.09
\footnote: (RmS) Added \let\protect\noexpand in \footnote, \footnotemark, and \footnotetext, since \xdef is used 930
1991-11-04 ltlists.dtx LaTeX2.09
\makelabel: (RmS) added default definition for \makelabel, to produce an error message. 815
1991-11-04 ltplain.dtx RmS
General: Removed \itemitem since never needed/useful with LaTeX. . 31
1991-11-06 ltbibl.dtx LaTeX2.09
\@citex: added code to remove a leading blank 939

1991-11-13 ltbibl.dtx LaTeX2.09
\@bibitem: Changed counter enumi to enumiv, as it says in the comment above 938
1991-11-21 ltfssini.dtx v1.1o
\reset@font: Added extra braces for robustness. 657
Changed to protected version of macro. 657
1991-11-22 ltfloat.dtx LaTeX2.09
\footnote: (RmS) Added \let\protect\noexpand in \@xfootnote, \@xfootnotemark, and \@xfootnotetext 930
1991-11-22 ltlists.dtx LaTeX2.09
\@item: (RmS) Changed second call to \makelabel to \unhbox\@tempboxa. Avoids problems with side effects in \makelabel and is more efficient. 815
1991-11-27 ltfssbas.dtx v1.3a
General: All \family, \shape etc. renamed to \fontfamily etc. . . . 523
1991-11-27 ltfssini.dtx v1.2a
General: All \family, \shape etc. renamed to \fontfamily etc. . . . 634
1992-01-06 ltfssini.dtx v1.2c
General: added slitex code 634
1992-01-10 ltbibl.dtx LaTeX2.09
\@bibitem: Changed \c@enumiv to \value of \@listctr 938
1992-01-10 ltmath.dtx LaTeX2.09
equation: RmS: put \hbox around \@eqnnum to typeset the equation number in text mode (as in the eqnarray env.) 794
1992-01-10 ltthm.dtx LaTeX2.09
\@othm: (RmS) Check for existence of theorem environment 896
1992-01-14 ltbibl.dtx LaTeX2.09
\@biblabel: removed \hfill 941
1992-01-14 ltsect.dtx 0.0
\@starttoc: (RmS) added \immediate to \openout as all \write commands are also executed \immediate 907
1992-02-26 ltbibl.dtx LaTeX2.09
\@lbibitem: Added \hfill to restore left-alignment of bibliography labels in alpha style 938
1992-03-18 ltdefns.dtx LaTeX209
General: (RMS) changed input channel from 0 to \@inputcheck to avoid conflicts with other channels allocated by \newread 82
1992-03-18 ltfloat.dtx LaTeX2.09
\@xympar: (RmS) added \global\@ignorefalse 925
\end@float: (RmS) changed \@esphack to \@Esphack 919
1992-03-18 ltlists.dtx 0.0
\trivlist: RmS: added \@nmbrlistfalse 810
1992-03-18 ltmiscen.dtx LaTeX2.09
\begin: Changed \@ignoretrue to \@ignorefalse (as documented) 769
1992-03-21 ltfssini.dtx v1.2d
General: Renamed \text to \nfss@text to make it internal. . 634
1992-05-12 ltfssbas.dtx v1.3c
\extract@alph@from@version: Macro added. 545
\select@group: Added call to \extract@alph@from@version. . 545
1992-07-26 ltfssbas.dtx v1.9a
\curr@fontshape: 536
\DeclareFontShape: Introduced \DeclareFontShape 524
\define@newfont: 537
\math@fonts: 544
\select@group: 544, 545
\split@name: Added splitting into \f@encoding. 536
\wrong@fontshape: 541, 542
1992-07-26 ltfsstrc.dtx v2.0b
\s@fct@: 593
\s@fct@sub: documentation fixes . . 594
\selectfont: 577
\try@simple@size: 587
\try@size@range: 591
\use@mathgroup: 584
1992-08-14 ltbibl.dtx LaTeX2.09
\@citex: added missing argument braces around \hbox, found by Ed Sznyter 939
1992-08-14 ltboxes.dtx LaTeX209
\endminipage: (RmS) replaced \vskip-\lastskip by \unskip (proposed by FMi) 832
1992-08-17 ltbibl.dtx LaTeX2.09
\@citex: simplified code for removing leading blanks in citation key (proposed by Frank Jensen and Kresten Krab Thorup) 939
1992-08-19 ltsect.dtx 0.0
\@xsect: (RmS) corrected bug: stretch and shrink in argument to \hskip

previously not negated 903
1992-08-19 ltthm.dtx LaTeX2.09
\\@othm: (RmS) Changed error message to complain about undefined counter 896
1992-08-20 ltfssini.dtx v1.4b
\\@setsize: Added \\@currsize. . . . 656
1992-08-24 ltdefns.dtx LaTeX209
\\@ifnextchar: (Rms) \\@ifnextchar didn't work if its first argument was an equal sign. 107
1992-08-24 ltmiscen.dtx LaTeX2.09
\\begin: Added code to \\begin to remember line number. Used by \\@badend to display position of non-matching \\begin. 769
\\verb: Changed \\verb and \\@sverb to work correctly in math mode . . . 783
1992-08-25 ltsect.dtx LaTeX2.09
\\@sect: (FMi) replaced explicit setting of \\@svsec by call to \\@seccntformat 902
1992-09-18 ltlists.dtx LaTeX2.09
\\item: (RmS) Added warning if \\item is used in math mode 813
1992-09-18 lttab.dtx LaTeX2.09
\\@array: Changed \\par to \\@empty to avoid starting new row e.g. after \\hline 850
1992-09-19 ltfsstrc.dtx v2.0c
\\try@simple@size: 587
1992-09-21 ltfssini.dtx v1.4d
\\not@math@alphabet: Macro defined. 655
1992-09-22 ltfssbas.dtx v1.91a
General: Introduced \\tf@size for math size. 523
1992-09-22 ltfsstrc.dtx v2.1a
\\getanddefine@fonts: Introduced \\tf@size for math size. 585
1992-11-13 ltfssini.dtx v?
\\hexnumber@: Made expandable. . . . 657
1992-11-23 ltcounts.dtx LaTeX209
\\stepcounter: Replaced {} in \\stepcounter by \\begingroup \\endgroup to avoid adding an empty ord in math mode 514
1992-11-26 ltboxes.dtx LaTeX2.09
\\@mpfootnotetext: (RmS) added protection for \\edef 832
1992-11-26 ltfloat.dtx LaTeX2.09
\\@footnotetext: (RmS) added protection for \\edef 930
\\footnote: (RmS) Changed all to 'def'protect'noexpand'protect'noexpand 930
1992-12-03 ltfssini.dtx v?
\\hexnumber@: Make it accept counters. 657
1993-03-08 preload.dtx v2.0b
General: Added 12pt preloads 683
1993-03-18 ltfssbas.dtx v2.0c
General: Changed all \\@tempdima in \\@tempdimb to avoid killing \\numberline 523
1993-03-18 ltfsstrc.dtx v2.1b
General: Changed all \\@tempdima in \\@tempdimb to avoid killing \\numberline 572
Changed all \\@tempdimb in \\@tempdimx to avoid killing \\numberline 572
1993-03-18 ltfsstrc.dtx v2.1c
\\DeclareSizeFunction: Added all args to avoid blanks problems . . 590
1993-04-09 lterror.dtx v1.0e
\\@latexerr: Mention The Companion 394
1993-04-11 lterror.dtx v1.0f
\\@latexerr: Remove setting of errorcontextlines 394
1993-05-05 ltfntcmd.dtx v2.0b
General: Removed all LaTeX related cmds 686
1993-05-16 ltfssbas.dtx v2.0e
\\showhyphens: Use \\reset@font . . 547
1993-07-16 ltfsstrc.dtx v2.1h
General: Changed layout of info messages 572
1993-07-17 ltoutenc.dtx 1.0d
General: changed \\catcoding @ . . . 469
1993-08-03 ltmiscen.dtx LaTeX2.09
\\enddocument: Changed redefinition of \\global to redefinition of \\@setckpt. 761
1993-08-05 ltpictur.dtx LaTeX2.09
\\circle: (RMS) Added error message if \\circle is used in math mode. 887
1993-08-05 ltsect.dtx LaTeX2.09
\\@sect: (RmS) Made sure that \\protect works correctly in expansion of \\the counter 902
1993-08-05 ltspace.dtx LaTeX2e
\\@hspace: (RmS) Removed superfluous \\leavevmode in \\@hspace and \\@hspacer, as suggested by CAR. 439

1993-08-05 lttab.dtx latex2e
\tabular*: Replaced \expandafter\def by \@namedef. 849
1993-08-06 ltbibl.dtx LaTeX2.09
\@citex: Moved writing to .aux file in loop over citation keys so that leading blanks are removed there as well. 939
1993-08-13 ltoutenc.dtx 1.0f
General: Protected against active @ sign. 469
1993-08-13 preload.dtx v2.0c
General: Added \relax at end of font names. 684
1993-08-16 ltoutenc.dtx 1.0g
General: Needs space after \string 469
1993-08-18 ltfssdcl.dtx v2.0e
\new@mathversion: Exchanged names of encodings in warning message of \SetSymbolFont. 614
1993-09-02 ltfsstrc.dtx v2.1i
General: Corrected name of sgen size function. 572
1993-09-03 ltmiscen.dtx LaTeX2.09
\verbatim@nolig@list: Replaced \@noligs by extensible list 783
1993-09-07 ltmiscen.dtx LaTeX2.09
\verb@balance@group: (RmS) Changed definition of \verb so that it detects a missing second delimiter. 783
1993-09-08 ltmiscen.dtx LaTeX2.09
\enddocument: Added warning in case of undefined references. 761
1993-09-15 ltfssbas.dtx v2.0g
\DeclareFontEncoding: Corrected: \default@T to \default@M. 527
1993-09-15 ltfsstrc.dtx v2.1j
General: Corrected spelling of \noxpand. 572
1993-09-19 lterror.dtx LaTeX2.09
\@invalidchar: (RmS) Error message for invalid input characters. 398
1993-11-02 ltmath.dtx LaTeX2.09
General: RmS: Corrected description of \@eqnsel, moved \@eqnsel accordingly and removed extra \tabskip assignment. 795
1993-11-03 ltmath.dtx LaTeX2e
General: RmS: Initialized \everycr to empty 795
1993-11-03 ltpictur.dtx LaTeX2.09
General: (RmS) changed \halign to \ialign to initialize \tabskip and \everycr 867
1993-11-11 ltfssini.dtx v2.1a
\normalfont: Macro added 657
1993-11-11 ltfsstrc.dtx v2.2a
General: Option concept added for LaTeX2e 572
1993-11-14 ltclass.dtx v0.2a
\@currext: Name changed from \@currextension 979
\@reset@ptions: macro added . . . 1007
\AtEndDocument: Included extension in the generated macro name for package and class hooks. 1007
\documentstyle: Added \RequirePackage \@unusedoptionlist stuff. 995
\load@onefilewithoptions: Moved resetting of \default@ds, \ds@ and \@declaredoptions here, from the end of \ProcessOptions. 1001
\NeedsTeXFormat: made more robust for alternative syntax for other formats. 997
\ProcessOptions*: Optimize 'empty option' code. 992
Stop adding the global option list inside class files. 992
1993-11-14 ltdefns.dtx v0.2a
\g@addto@macro: Made global 112
1993-11-15 ltclass.dtx v0.2b
\documentstyle: Modified to match \ProcessOption* 995
\ProcessOptions*: Star form added. 992
1993-11-17 ltclass.dtx v0.2c
\@@fileswith@pti@ns: Macro added 1007
\@badrequireerror: Macro added 1009
\@twoloadclasserror: Macro added 1009
\CurrentOption: Name changed from \@curroption 979
\DeclareOption*: Error checking added 991
\load@onefilewithoptions: Added trap for two \LoadClass commands. 1003
\NeedsTeXFormat: Name changed from \NeedsFormat 997
\ProcessOptions*: restoring \@fileswith@pti@ns added. . . . 992
1993-11-18 ltclass.dtx v0.2d
\documentstyle: Modified \RequirePackage stuff. 995
\ExecuteOptions: Use \CurrentOption not \reserved@a 995

\NeedsTeXFormat: \fmtname \fmtversion not \@... 997
1993-11-21 ltfiles.dtx LaTeX2e
\@missingfileerror: Stop infinite looping on \@er@ext 461
1993-11-21 ltmiscen.dtx v0.9a
\@verbatim: use \verbatim@font instead of \tt 778
\verb: Use \verbatim@font instead of \tt. 783
\verbatim@font: Macro added 779
1993-11-22 ltclass.dtx v0.2f
\@fileswithoptions: Made the default [] not [\@unknownversion] 999
\@ifl@ter: Added //00 so parsing never produces a runaway argument. 984
General: \@unknownversion removed 974
\load@onefilewithoptions: Made the initial version [] not [\@unknownversion] 1001
1993-11-22 ltdefns.dtx LaTeX2e
\@minus: Macro added 81
\@plus: Macro added 81
\CheckCommand: Macro added 88
\providecommand: Macro added 88
1993-11-22 lterror.dtx LaTeX2e
\c@errorcontextlines: Macro added 394
1993-11-22 ltfiles.dtx LaTeX2e
\listfiles: Removed checking for \@unknownversion 463
1993-11-22 ltlength.dtx LaTeX2e
\@settodim: Macro added 521
\@settopoint: Macro added 522
\settodepth: Macro added 521
\settoheight: Macro added 521
1993-11-22 ltlogos.dtx LaTeX2e
\LaTeXe: Macro added 441
1993-11-23 ltclass.dtx v0.2g
\@use@ption: Name changed from \@executeoption 994
General: Various macros now moved to latex.tex. 978
Warnings and errors now directly coded. 978
1993-11-23 ltdefns.dtx LaTeX2e
\@argdef: Macro added 84
\@ifundefined: Redefined to remove a trailing \fi 106
\@newcommand: Macro added 84
\@newenv: Macro interface changed . . 87
\@xargdef: Macro interface changed . 84
\@yargd@f: Avoid \@?@? token 85
Macro interface changed 85
\newcommand: Macro reimplemented and extended 83
\renewcommand: Macro reimplemented and extended 86
\renewenvironment: Macro reimplemented and extended 87
\two@digits: Macro added 80
1993-11-23 ltoutput.dtx v0.1a
\paperheight: Register added . . . 1099
\paperwidth: Register added 1099
1993-11-23 ltoutput.dtx v0.1c
\@enlargepage: Command added . 1144
\@kludgeins: Insert added 1143
\@makecol: Command changed . . . 1111
\@specialoutput: Command changed 1104
\enlargethispage*: Commands added 1143
1993-11-24 ltfntcmd.dtx v2.1a
\maybe@ic@: Use \t@st@ic 691
\t@st@ic: Macro added 692
1993-11-24 ltfssini.dtx v2.1a
General: Removed \xpt stuff 657
1993-11-24 ltlogos.dtx LaTeX2e
\LaTeX: Macro changed 441
1993-11-28 ltclass.dtx v0.2h
\@twoclasseserror: Macro added 1009
General: Assorted commands now in the kernel removed. 978
Directory syntax checking moved to dircheck.dtx 978
Primitive filenames now terminated by space not \relax. 978
\endfilecontents: Don't globally allocate a write stream (always use 15) 1010
1993-11-28 ltfiles.dtx LaTeX2e
\@missingfileerror: Use filename parser from dircheck 461
1993-11-29 ltoutput.dtx v1.0b
\@makecol: \@makespecialcolbox added 1111
\@makespecialcolbox: Command added 1112
1993-11-29 ltplain.dtx LaTeX2e
General: All accents in decimals; suggested by Paul Taylor 32
1993-11-30 ltoutput.dtx v1.0c
\fl@tracemessage: Commands added 1146
1993-12-01 fontdef.dtx v2.1a
General: Update for LaTeX2e 662
1993-12-01 ltoutput.dtx v1.0e
\@reinserts: Command added . . 1112

1993-12-03 ltboxes.dtx v0.1a
\\@argrsbox: macro removed 835
\\@begin@tempboxa: macro added . . 820
\\@end@tempboxa: macro added 820
\\@iirsbox: redefined to support \\height 835
\\@imakebox: macro modified 821
\\@irsbox: redefined to support \\height 835
\\@isavebox: color support 824
extra group 824
\\@isavepicbox: extra group 824
\\@makebox: default changed from x to c 820
\\@makepicbox: macro modified 821
\\@savebox: default c not x 824
\\bm@b: macros added 820
\\endlrbox: macro added 825
\\fbox: extra group 825
\\lrbox: color support 824
macro added 824
\\makebox: modified 819
\\mbox: extra group 820
\\minipage: Redefined to support extra optional arguments 831
\\newsavebox: Pass the whole of arg 1 to \\@ifdefinable 823
\\parbox: Redefined to support extra optional arguments 828
\\raisebox: redefined to support \\height 834
\\sbox: color support 824
extra group 824
\\set@color: color support 822
macro added 822
1993-12-03 ltclass.dtx v0.2i
\\@cls@pkg: Name changed to avoid clash with output routine. 1008
General: \\@onlypreamble: Many commands declared. 978
Removed obsolete \\@documentclass 978
1993-12-03 lterror.dtx v1.0b
\\@latexerr: Set \\c@errorcontextlines to -1 . . . 394
1993-12-03 ltfssini.dtx v2.1a
General: update for LaTeX2e 634
1993-12-04 ltfilehook.dtx v0.9b
\\unqu@tefilef@und: Macro added 1046
1993-12-04 ltfiles.dtx v0.9b
\\@iinput: Macro reimplemented . . . 460
\\@input: Macro reimplemented . . . 461
\\IfFileExists@: Macro added 457
\\input: Macro reimplemented 460
1993-12-05 ltfloat.dtx LaTeX2e
\\@dblfloatplacement: Command changed 921
\\@xfloat: Command changed 915
1993-12-05 ltoutput.dtx v1.0f
\\@addtobot: Command changed . . 1125
\\@addtocurcol: Command changed 1127
\\@addtodblcol: Command changed 1137
\\@addtonextcol: Command changed 1134
\\@addtotoporbot: Command changed 1126
\\@boxfpsbit: Command added . . 1149
\\@flcheckspace: Command added 1151
\\@flsetnum: Command added . . . 1150
\\@flsettextmin: Command added 1151
\\@flstop: Commands added 1147
\\@flupdates: Command added . . 1152
\\@fpsadddefault: Command added 1148
\\@getfpsbit: Command added . . 1149
\\@opcol: Command changed 1110
Hook added 1110
\\@outputpage: Command changed 1114
\\@resethfps: Command added . . 1150
\\@setfloattypecounts: Command added 1149
\\@setfpsbit: Command added . . 1150
\\@shipoutsetup: Command added 1114
\\@startcolumn: Command changed 1120
\\@startdblcolumn: Command changed 1120
\\@testfp: Command added 1149
\\@textfloatsheight: Commands added 1148
\\@topnewpage: Commands changed 1102
\\@tryfcolumn: Command changed 1121
\\@writesetup: \\@startpagehook added 1114
\\output: Command changed 1104
1993-12-06 ltclass.dtx v0.2k
\\ExecuteOptions: Preserve \\CurrentOption. 995
1993-12-06 ltoutput.dtx v1.0f
\\@specialoutput: Unboxing of 255 added to rescue writes 1104
1993-12-06 ltoutput.dtx v1.0g
\\@topnewpage: \\@floatplacement placement bug fixed 1102
1993-12-07 ltclass.dtx v0.2l
\\ProvidesFile: Macro added 989
1993-12-07 ltclass.dtx v0.2m
\\load@onefilewithoptions: Reset \\CurrentOption 1001

1993-12-07 ltoutenc.dtx 1.1
General: Protected all special characters with \string. 469
1993-12-07 ltoutenc.dtx v1.1
General: Made all character numbers decimal. 466
Removed a lot of equal signs and the like. 466
1993-12-08 ltboxes.dtx v0.1b
\@begin@tempboxa: Extra braces for color support (braces removed from other macros) 820
\@irsbox: fix typo 835
\@parboxto: \endgraf added due to extra group in \@begin@tempboxa 828
\lrbox: move \@endpefalse out of the inner group 824
1993-12-08 ltfntcmd.dtx v2.1b
General: Macros \rm, \bf and \sf moved to classes.dtx 694
1993-12-08 ltlists.dtx LaTeX2e
\@item: use \sbox to support colour 815
1993-12-08 ltspace.dtx LaTeX2e
\@bsphack: Command reimplemented 428
Command reimplemented; late birthday present for Chris 428
\@vbsphack: Command added 431
1993-12-09 ltboxes.dtx v0.1c
\@irsbox: fix another typo 835
1993-12-09 ltclass.dtx v0.2n
\documentstyle: input 209 compatibility file. 995
1993-12-09 ltfiles.dtx v0.9e
\document: Hook added 444
1993-12-09 ltmiscen.dtx v0.9e
\enddocument: Hook added 761
1993-12-10 ltoutenc.dtx v1.2
General: Added source code for t1enc.sty. 466
1993-12-11 ltfntcmd.dtx v3.0a
General: Complete reworking of all text commands, using just one creator function 686
italic correction now put in front of penalty before glue 686
newcommands replaced by defs . . 686
newfontswitch command corrected and changed 686
\DeclareTextFontCommand: Macro changed 688
\emph: Macro changed 689
\fix@penalty: Macro added 692
\maybe@ic: Macro name changed . . 691
\maybe@ic@: Macro and name changed 691
\sw@slant: Macro changed 692
\textup: Macros changed 689
1993-12-11 ltmath.dtx v0.9g
General: Added a group around the first argument of \frac to prevent changes (for example font changes) from modifying the contents of the second argument. 795
1993-12-11 ltoutenc.dtx v1.2a
General: Corrected for t1enc, math. 466
1993-12-11 ltsect.dtx LaTeX2e
\author: Added default 898
\title: Added default 898
1993-12-11 ltxref.dtx LaTeX2e
\@setref: Macro added 740
\pageref: Macro reimplemented . . . 740
\ref: Macro reimplemented 740
1993-12-12 ltoutput.dtx v1.0h
\@cflb: boxmaxdepth setting moved 1119
defs changed to lets 1119
\@cflt: name changed 1119
\@doclearpage: defs changed to lets 1109, 1110
\@makecol: defs changed to lets . . 1111
\@resethfps: Warnings added: minimal 1150
\@startdblcolumn: defs changed to lets 1120, 1121
\@topnewpage: braces removed . . . 1103
\@tryfcolumn: defs changed to lets 1122
\fl@tracemessage: Commands changed 1146
1993-12-13 ltclass.dtx v0.2o
General: Removed setting \errorcontextlines (now in latex.tex) 978
\documentstyle: compatibility file now latex209.sty. 995
\usepackage: Fixed error handling . 997
1993-12-13 ltdirchk.dtx v0.2a
General: on the 'docstrip' pass, do not check openin path 10
\IfFileExists: Removed interactive prompting for current directory syntax 10
\strip@prefix: modified, name changed from \stripmeaning. . . . 5
1993-12-13 ltlists.dtx latex2e
\trivlist: Initialised \@itemlabel 810
1993-12-13 ltmiscen.dtx v0.9h
\@noligs: Readded \@noligs 784
\@verbatim: Readded \@noligs . . . 778

Removed optional argument of \item 778
center: Removed optional argument of \item 775
flushleft: Removed optional argument of \item 776
flushright: Removed optional argument of \item 777
1993-12-13 ltoutenc.dtx v1.2b
General: Corrected file name in driver code. 466
1993-12-13 lttab.dtx latex2e
\tabbing: Removed optional argument of \item 844
1993-12-14 ltoutput.dtx v1.0i
General: Section added to declare all parameters 1158
1993-12-15 ltboxes.dtx v0.1d
\@iminipage: Changed default from 'c' to 's' 831
\@iparbox: Changed default from 'c' to 's' 828
\minipage: Changed default from 'c' to 's' 831
extra space removed. 831
\parbox: Changed default from 'c' to 's' 828
1993-12-15 ltclass.dtx v0.2p
General: Removed extra '.'s from \@@warnings 978
1993-12-16 ltlogos.dtx LaTeX2e
\LaTeXe: Extended logo by DPC . . 441
1993-12-16 ltmath.dtx v0.9i
\@@eqncr: use \refstepcounter instead of shortcut 796
General: use \refstepcounter instead of shortcut 795
1993-12-16 ltmiscen.dtx v0.9i
General: \literal added 784
1993-12-16 ltpage.dtx LaTeX2e
\mark: Init \mark at begin document 972
1993-12-16 ltspace.dtx LaTeX2e
\@bsphack: Corrected optimisation . 428
1993-12-16 lttab.dtx latex2e
\@xhline: Measure from middle of vertical rules 859
1993-12-17 ltclass.dtx v0.2q
\@documentclasshook: Macro added 978
\@fileswithoptions: Add \@compatibility hook 999
\documentstyle: Match Alan's new code. 995
1993-12-17 ltoutenc.dtx 1.3
General: Added this section 470
Removed all the hackery for use in \DeclareFontEncoding, and redid everything using \DeclareTextFoo. 482, 484
Removed the catcode hackery, since the file is only read as a package in the preamble, and removed all the messages on the screen, which just confuse users. Replaced them by the appropriate \ProvidesPackage commands. Added XXXenc. . . . 469
1993-12-17 ltoutenc.dtx v1.3
General: Added \EncodingSpecificAccent, \EncodingSpecificAccentedLetter and \EncodingSpecificCommand. 466
Made Rokicki's encoding a proper encoding scheme rather than a variant of OT1. 466
1993-12-17 ltoutput.dtx v1.0j
\@opcol: Hook removed 1110
\@specialoutput: Page room test added 1105
\@topnewpage: check for vsize too small added 1102
Page room test added 1104
\@writesetup: —and then removed 1114
\fl@tracemessage: tracefloatvals made a document command . . 1146
1993-12-17 ltpage.dtx LaTeX2e
\mark: Removed init \mark at begin document, since it doesn't work. 972
\rightmark: Stopgap solution to mark \leftmark and \rightmark work without initializing mark until the problem is solved. 972
1993-12-18 ltoutenc.dtx 1.3b
General: Fixed typos with \ProvidesPackage lines. Added the \NeedsTeXFormat line. Added the last argument to \DeclareEncoding. Moved the use of the encodings to after their declaration. 469
Replaced the missing last argument to \DeclareFontEncoding. 482, 484
1993-12-18 ltoutenc.dtx 1.3c
General: Rewrote for the new syntax of \EncodingSpecific. . . . 482, 484
Split \EncodingSpecificAccent up into \EncodingSpecific and \DeclareAccent. 470
1993-12-18 ltoutenc.dtx v1.3a
General: Replaced OT3 by XXX . . 466

1993-12-18 ltoutenc.dtx v1.3b
General: Corrected typos. 466
Replaced the missing last argument to `\DeclareFontEncoding`. 466
1993-12-18 ltoutenc.dtx v1.3c
General: A new syntax, separating accent-definitions from encoding-specific definitions, and allowing encoding-specific `\chardef`, `\let`, etc. 466
Rewrote for the new syntax of `\EncodingSpecific`. 466
1993-12-18 ltoutenc.dtx v1.3d
General: Some T1 stuff had drifted into the OT1 file. 466
1993-12-18 ltpage.dtx LaTeX2e
`\sloppy`: Added `\emergencystretch` 972
1993-12-19 ltclass.dtx v0.2r
`\endfilecontents`: Different message when ignoring a file 1010
1993-12-19 ltfntcmd.dtx v3.0b
General: `\@pdef` command added . . 686
Added by ASAJ. 694
Made `\@newfontswitch` produce an error if command already exists, and added `\@renewfontswitch`, ASAJ 686
Other tidying 686
Some more tidying done 686
Untidying added, so this is now a TEMPORARY version. 686
Wording changes by CAR. 694
`\DeclareOldFontCommand`: Corrected and tidied 693
`\DeclareTextFontCommand`: Corrected and tidied 688
1993-12-19 ltspace.dtx LaTeX2e
`\@bsphack`: There seem to be problems with selfmade birthday presents . 429
1993-12-20 ltdefns.dtx LaTeX2e
`\@reargdef`: Kept old version of `\@reargdef`, for array.sty 85
1993-12-20 ltfiles.dtx v0.9m
`\@obsoletefile`: Added this command, removed @oldfilewarning 463
1994-01-05 fontdef.dtx v2.1d
General: Removed nf prefix from file names. 664
1994-01-13 ltmath.dtx v0.9o
`\@@eqncr`: correcting 0.9i 796
General: correcting 0.9i 795
1994-01-14 ltdirchk.dtx v0.2d
`\IfFileExists`: Close the texsys.aux output stream 10
1994-01-15 ltfiles.dtx v0.9o
`\document`: move `\@preamblecmds` after document hook 446
1994-01-17 ltclass.dtx v0.2s
`\@fileswithoptions`: Modify to reduce parameter stack usage . . 999
General: Added many more `\@onlypreamble` commands 978
Wrapped long lines to column 72 978
`\load@onefile@withoptions`: Modify to reduce parameter stack usage 1004
1994-01-17 ltfiles.dtx LaTeX2e
`\listfiles`: New Version, adds '.tex' if needed, and lines up columns . 463
1994-01-17 ltfssbas.dtx v2.1a
General: New math font setup 523
`\curr@math@size`: New math font setup 536
`\everydisplay`: New math font setup 535
`\everymath`: New math font setup . 535
`\frozen@everydisplay`: New math font setup 535
`\frozen@everymath`: New math font setup 535
`\math@version`: New math font setup 534
1994-01-17 ltfssini.dtx v2.1e
`\not@math@alphabet`: Message changed 655
1994-01-17 ltfsstrc.dtx v2.3a
General: New math font setup 572
`\check@mathfonts`: New math font setup 582
`\glb@currsize`: New math font setup 580
`\restglb@settings`: New math font setup 583
1994-01-18 ltbibl.dtx LaTeX2e
`\bibliography`: Use `\@input@` so include files are listed. 939
1994-01-18 ltclass.dtx v0.2t
`\@ifclassloaded`: Fix typo `\@pkgetension` 983
1994-01-18 ltfilehook.dtx v0.9p
`\unqu@tefilef@und`: New Definition 1046
1994-01-18 ltfiles.dtx v0.9p
`\@iffileonpath`: Macro added 459
`\@input`: do not use a different definition for `\input@path` 461
`\@input@`: Macro added 461
`\IfFileExists@`: New Definition . . 457
`\includeonly`: Use `\@input@` so include files are listed. 450

1994-01-18 ltfssini.dtx v2.1f
\not@math@alphabet: Message corrected 655
1994-01-18 ltmiscen.dtx v0.9p
\@verbatim: Add \global\@inlabelfalse 778
Only add \penalty if in hmode . 778
1994-01-19 fontdef.dtx v2.1e
General: Added missing setting for symbols in bold version. 669
1994-01-19 ltdirchk.dtx v0.2e
\IfFileExists: name changed from \test 9
\input@path: No longer check that an empty group is in the path 10
\strip@prefix: name changed from \strip@meaning, to match NFSS. 5
1994-01-19 ltmath.dtx v1.0n classes
\mathindent: Deferred setting of \mathindent 798
1994-01-20 ltdirchk.dtx v0.2f
General: \@copytexsys and the texsys.new file removed 8
Modify all of ltxcheck 13
\IfFileExists: \@copytexsys removed 10
1994-01-21 ltclass.dtx v0.2u
\documentstyle: compatibility file now latex209.def. 995
1994-01-21 ltdirchk.dtx v0.2g
General: Improve documentation, reorganize docstrip module 1
\filename@parse: Minor changes, and add Mac version (:) 11
\today: Name changed from \stamp, to save memory 9
1994-01-21 ltfloat.dtx LaTeX2e
\@xfloat: Added missing percent characters. 915
1994-01-21 ltmiscen.dtx v0.9s
\verbatim@font: Removed unnecessary category code hackery. 779
1994-01-24 ltdirchk.dtx v0.2h
\IfFileExists: Stop testing once texsys.aux has been found 9
1994-01-24 ltpage.dtx LaTeX2e
\pagestyle: (DPC) Complain if pagestyle is undefined. 969
1994-01-25 ltdirchk.dtx v0.2i
General: Protect against looping on \@@input and \@@end. 3
1994-01-25 ltfssbas.dtx v2.1b
\math@version: Corrections for math setup 534
1994-01-25 ltmath.dtx LaTeX2e
\bordermatrix: Removed \p@renwd. 790
1994-01-26 ltfsstrc.dtx v2.3c
\check@mathfonts: Correct trace info placement 582
\restglb@settings: Correct trace info placement 583
1994-01-27 ltfntcmd.dtx v3.1a
\nocorrlist: Only ., used as default for cm fonts 693
1994-01-29 ltclass.dtx v0.2v
\@@unprocessedoptions: Macro added. 1009
\load@onefile@withoptions: All options raise error if no \ProcessOptions appears 1005
1994-01-31 ltdefns.dtx v0.2w
\g@addto@macro: Use toks register to avoid 'hash' problems 112
1994-01-31 ltfiles.dtx v0.9t
\document: set \@normalsize or \normalsize if necessary 445
1994-01-31 ltfntcmd.dtx v3.1b
General: \@normalsize no longer defined 686
1994-02-01 ltpage.dtx LaTeX2e
\pagestyle: (DPC) Modify to get nicer error message 969
\thispagestyle: (DPC) Modify to get nicer error message 970
1994-02-02 ltclass.dtx v0.2x
\load@onefile@withoptions: Only run the hook and options check if the file was loaded. 1005
1994-02-03 ltoutput.dtx v1.0k
\@makespecialcolbox: correct mistakes in the documentation . 1113
1994-02-07 ltclass.dtx v0.2y
\@fileswithoptions: Run \@compatibility on the first class to start (not the first to finish) . 999
\@ifclasswith: Add extra ,s so 'two' is not matched with 'twocolumn' 985
\ProcessOptions*: Add extra ,s so 'two' is not matched with 'twocolumn' 992
1994-02-07 ltfssbas.dtx v2.1c
\DeclareFontEncoding: revert catcode settings earlier 527
\DeclareFontShape@: revert catcode settings earlier 524

1994-02-08 ltoutput.dtx v1.0k
\\@makespecialcolbox: boxmaxdepth setting added 1113
boxmaxdepth setting removed . 1112
General: Documentation and tasks tidied. 1087
1994-02-10 ltclass.dtx v0.2z
\\@documentclasshook: Changed the name from \\@compatibility to \\@documentclasshook, and added the check for whether \\@normalsize has been defined. ASAJ. 978
\\@fileswithoptions: Renamed \\@compatibility to \\@documentclasshook. ASAJ. . . 999
1994-02-10 ltfssbas.dtx v2.1d
\\addto@hook: Made \\addto@hook long. 549
1994-02-10 ltfsscmp.dtx v2.1d
\\scan@@fontshape: scan away stuff after pt 598
1994-02-22 ltfssini.dtx v2.1g
General: Correct error message 659
1994-02-24 ltfssbas.dtx v2.1e
\\DeclareFontShape: Separate restoration of catcodes for fd cmds 524
\\define@newfont: Separate restoration of catcodes for fd cmds 537
\\nfss@catcodes: Separate restoration of catcodes for fd cmds 538
1994-02-25 ltdirchk.dtx v0.2j
General: Remove need for drv file 1
1994-03-01 ltdirchk.dtx v0.2k
General: Add unstripped module, so that dircheck.dtx may be used with initex 1
1994-03-02 ltboxes.dtx v0.1e
General: Add 2ekernel module 819
Remove need for drv file 819
1994-03-02 ltclass.dtx v0.3a
General: Remove need for driver file 978
1994-03-03 ltboxes.dtx v0.1f
\\@irsbox: Replaced a missing \\else 835
1994-03-04 ltfloat.dtx v1.0a
General: Initial version, split from latex.dtx 911
1994-03-04 ltsect.dtx v1.0a
General: Initial version, split from latex.dtx 898
1994-03-04 lttab.dtx v1.0a
General: Initial version, split from latex.dtx 837
1994-03-04 ltvers.dtx v1.0a
General: Initial version, split from latex.dtx 37
1994-03-07 ltboxes.dtx v0.1a
\\@mpfootnotetext: Extra group for color 832
1994-03-07 ltboxes.dtx v1.0a
General: Unify format with other Kernel files 819
1994-03-07 ltdefns.dtx v1.0a
\\@@italiccorr: Macro added 81
1994-03-07 ltfiles.dtx v1.0a
General: Initial version, split from latex.dtx 442
Long lines wrapped to 72 columns 442
1994-03-07 ltfinal.dtx v0.1a
General: Add code from the old dump.dtx 1190
Initial version, split from latex.dtx 1174
move code here from lhyphen.dtx 1180
Remove oldcomments environment 1174
use \\InputIfFileExists not \\IfFileExists 1180
1994-03-07 ltfloat.dtx v1.0a
\\@endfloatbox: (DPC) Extra group for colour 920
\\@footnotetext: (DPC) Extra group for colour 930
\\@xfloat: (DPC) Extra group for colour 916
1994-03-07 lthyphen.dtx v0.1c
General: move the 2ekernel code to ltfinal.dtx 1172
1994-03-07 ltlength.dtx v1.0a
\\@settodim: (DPC) Extra group for colour 521
1994-03-07 ltlists.dtx v1.0a
General: Initial version, split from latex.dtx 802
Long lines wrapped to 72 columns 802
1994-03-07 ltpage.dtx v1.0a
General: Initial version, split from ltherest.dtx 969
1994-03-07 ltpictur.dtx v0.1a
General: Initial version, split from latex.dtx 862
Long lines wrapped to 72 columns 862
1994-03-07 ltsect.dtx v1.0a
\\@hangfrom: (DPC)Extra groups for colour 905

1994-03-07 lttab.dtx v1.0a
General: Long lines wrapped to 72 columns 837
1994-03-08 ltclass.dtx v0.3b
General: Modify driver code into 'new style' 978
1994-03-08 ltdirchk.dtx v1.0a
General: Reorganize driver module into 'new style' 1
1994-03-08 ltplain.dtx v1.0a
General: Remove need for a driver file. 14
1994-03-10 ltfssbas.dtx v2.2f
`\math@egroup`: Changed `\begingroup`/`\endgroup` to `\bgroup`/`\egroup`. 546
1994-03-11 ltfssdcl.dtx v2.1b
`\DeclareSymbolFontAlphabet@`: Added check against use of alphabet switch outside of math mode. 632
`\SetMathAlphabet@`: Changed parameter template in temporary macro to catch check add below. 621
1994-03-12 ltclass.dtx v0.3c
General: Change name from docclass to ltclass 978
`\ProvidesFile`: Add `\wlog` 989
`\ProvidesPackage`: Add `\wlog` 987
use `\@gtempa` 987
1994-03-12 ltdefns.dtx v1.0b
`\@reargdef`: New defn, in terms of `\@yargdef` 85
`\@yargd@f`: Name changed from `\XXX@argdef` 85
1994-03-12 ltdirchk.dtx v1.0b
General: Change name from dircheck.dtx 1
Minor edits to the typeouts in ltxcheck 1
1994-03-12 ltfloat.dtx v1.0b
`\@savemarbox`: (DPC) Extra group for colour 924
`\@xympar`: (DPC) Extra bgroup for colour 925
1994-03-12 ltplain.dtx v1.0b
General: Name changed from lplain. The end of an era 14
1994-03-12 ltplain.dtx v1.0e
General: Replaced remaining width, height, depth by LaTeX macro names to save tokens. 14
1994-03-13 ltcntrl.dtx v1.0c
`\@tfor`: (DPC) Add `\@tf@r` so a single group is correctly treated. 387
1994-03-13 ltfilehook.dtx v0.3b
`\unqu@tefilef@und`: Use new cmd `\@addtofilelist` 1046
1994-03-13 ltfiles.dtx LaTeX2e
`\@addtofilelist`: Macro added . . . 463
`\listfiles`: Reset `\@addtofilelist` at begin document 464
1994-03-13 ltfssbas.dtx v2.1g
General: add 2ekernel module to omit repeated code 523
1994-03-13 ltfssdcl.dtx v2.1c
General: add 2ekernel module to omit repeated code 602
1994-03-14 ltboxes.dtx v1.0b
`\@isavebox`: Use `\color@setgroup` . 824
`\@isavepicbox`: Use `\color@setgroup` 824
`\color@begingroup`: macro added for color support 822
`\color@endgroup`: macro added for color support 823
`\lrbox`: Use `\color@setgroup` 824
`\sbox`: Use `\color@setgroup` 824
1994-03-14 ltfloat.dtx 1.0c
`\@xympar`: (DPC) Use `\color@begingroup` 925
1994-03-14 ltfloat.dtx v1.0c
`\@endfloatbox`: (DPC) Use `\color@endgroup` 920
`\@footnotetext`: (DPC) Use `\color@begingroup`, add `\endgraf` 930
`\@savemarbox`: (DPC) Use `\color@begingroup` 924
`\@xfloat`: (DPC) Use `\color@begingroup` 916
1994-03-15 ltfiles.dtx LaTeX2e
`\@missingfileerror`: Quit on x or X just like a real error 461
1994-03-15 ltfntcmd.dtx v3.2a
General: Adapted to mass formatting 686
Changed `\/` to `\@@italiccorr` . . 686
Removed `\@renewfontswitch` . . . 686
Removed defs of short-forms and all sizes except `\normalize` 686
1994-03-15 ltoutput.dtx v1.0l
`\@addtocurcol`: Changed `\addvspace` to `\vskip` 1129, 1132
`\@combinedblfloats`: Removed boxmaxdepth setting. 1120
`\@makecol`: `\maxdepth` changed to `\@maxdepth` 1111
Removed boxmaxdepth setting. 1112

\@makespecialcolbox: Removed boxmaxdepth setting. 1113
\@topnewpage: Corrected and amended warning message 1103
Warning added: it should be improved 1104
General: Added some warnings when page gets full of top floats. . . . 1087
Driver added and further tidying. 1087
Removed duplicated code and corrected docstrip options. 1087
Some boxmaxdepth settings removed. 1087
1994-03-16 ltclass.dtx v0.3f
General: Add pkgindoc package . . 1026
1994-03-16 ltfiles.dtx LaTeX2e
\listfiles: Move this code directly into \document 464
1994-03-16 ltfiles.dtx v1.0c
\document: (DPC) directly add file list settings 446
1994-03-16 ltmiscen.dtx v1.0b
\@verbatim: Remove \global\@inlabelfalse again. . 778
1994-03-28 ltalloc.dtx v1.0d
General: Redefinition of 'new' allocations removed. 382
1994-03-28 ltdirchk.dtx v1.0d
General: Improve documentation 1
1994-03-28 lterror.dtx v1.0d
\@invalidchar: (DPC) Comment out (use catcode15 instead) 398
General: Remove test for \inputlineno undefined. 394
1994-03-28 ltfiles.dtx v1.0d
\document: (DPC) Use \normalsize not \@normalsize 445
(DPC) remove \@normalsize check 445
1994-03-28 ltfloat.dtx v1.0b
\@caption: Use \normalsize not \@normalsize 914
General: Split further from ltherest.dtx 911
1994-03-28 ltlists.dtx v1.0b
General: Improve documentation . . 801
1994-03-28 ltmiscen.dtx v1.0c
General: Improve Documentation . . 760
1994-03-28 ltplain.dtx v1.0c
\newlanguage: Remove some \outer declarations. 17
1994-03-28 ltsect.dtx v1.0b
General: Split further from ltherest.dtx 898
1994-03-28 lttab.dtx v1.0b
General: Improve documentation . . 837
1994-03-28 ltthm.dtx v1.0a
General: Initial version, split from latex.dtx 894
1994-03-29 ltcounts.dtx v1.0c
General: Create file from parts of ltmiscen and ltherest. 512
1994-03-29 ltlength.dtx v1.0c
General: Create file ltcntlen from parts of ltmiscen and ltherest. 521
1994-03-29 ltmiscen.dtx v1.0d
General: Remove counter macros to ltcntlen 760
1994-03-29 ltpageno.dtx v1.0c
General: Create file ltcntlen from parts of ltmiscen and ltherest. 737
1994-03-29 ltxref.dtx v1.0c
General: Create file ltcntlen from parts of ltmiscen and ltherest. 738
1994-03-31 ltbibl.dtx v1.0a
General: Initial version of ltidxbib.dtx, split from ltherest.dtx 937
1994-03-31 ltidxglo.dtx v1.0a
General: Initial version of ltidxbib.dtx, split from ltherest.dtx 934
1994-04-09 ltcounts.dtx v1.0d
\@newctr: \@nocnterr now has counter name argument 513
\addtocounter: \@nocnterr now has counter name argument 513
\setcounter: \@nocnterr now has counter name argument 513
\stepcounter: Use \addtocounter to have name checked 514
1994-04-09 ltthm.dtx v1.0b
\@othm: Use standard counter error message (FMi) 896
1994-04-11 ltclass.dtx v0.3g
\endfilecontents: Add star form, don't write \endinput at the end of the file. 1010
\ProvidesFile: Protect against weird catcodes. 989
1994-04-11 ltfssbas.dtx v2.1h
General: Added \defaultscriptratio and \defaultscriptscriptratio. ASAJ. 523
\defaultscriptratio: Macro added 547
\defaultscriptscriptratio: Macro added 547
1994-04-12 ltboxes.dtx v1.0c
General: Remove \@acci, now defined in ltplain.dtx 829

Remove \@dischyph, now defined in ltinit.dtx 829
1994-04-12 ltdefns.dtx v1.0g
\@dischyph: Define \@dischyph, was previously in ltboxes.dtx 110
1994-04-12 ltplain.dtx v1.0d
General: Define \@acci 32
1994-04-12 ltvers.dtx v1.0b
General: Have version info generated automatically. 37
1994-04-14 ltfntcmd.dtx v3.2b
General: Macros renamed to non-private forms, JB 686
\DeclareOldFontCommand: Renamed from \@newfontswitch 693
1994-04-15 ltboxes.dtx v1.0d
\@isavebox: Added missing percent character. 824
1994-04-17 ltcounts.dtx v1.0e
\@newctr: Use \@nocounterr instead of \@nocnterr 513
\addtocounter: Use \@nocounterr instead of \@nocnterr 513
\setcounter: Use \@nocounterr instead of \@nocnterr 513
1994-04-17 lterror.dtx v1.0h
\@nocounterr: New name for error message, old error message (without arg) kept 395
1994-04-17 ltthm.dtx v1.0c
\@othm: Use new std counter error message (FMi) 896
\@thm: Use new std counter error message (FMi) 896
1994-04-18 ltfinal.dtx v0.1b
General: Initialise \textheight, \textwidth and page style . . . 1177
1994-04-18 ltfloat.dtx v1.0d
\@footnotetext: (DPC) Remove Colour support 930
\@savemarbox: (DPC) Remove Colour support 924
1994-04-18 ltfssbas.dtx v2.1i
General: Macro \no@alphabet@help removed again 523
\calculate@math@sizes: Changed message to log only 546
\no@alphabet@error: Use std LaTeX error macro 523
1994-04-18 ltfssdcl.dtx ???
\DeclareMathAlphabet: Pass correct arg (2 not 3) 619
1994-04-18 ltfssdcl.dtx v2.1d
General: Removed surplus \no@alphabet@error (see fam.dtx) 602
1994-04-18 ltfsstrc.dtx v2.3d
General: Changed to new error/warning scheme 572
\font@submax: Changed dimen to macro 591
\fontsubfuzz: Changed dimen to macro 591
\subst@size: \font@submax and \fontsubfuzz now macros 592
1994-04-19 ltpage.dtx v1.0b
General: Improve documentation . . 969
1994-04-20 ltfntcmd.dtx v3.3a
General: Documentation up-dated . 686
New implementation of \nocorr . 686
\check@nocorr@: Macros added . . . 690
\maybe@ic@: \nocorr etc removed from list of tokens to check, leaving only punctuation characters 691
1994-04-20 ltmiscen.dtx v1.0e
\@enddocument@kernel@warnings: Changed logic for producing warning messages 763
1994-04-21 ltboxes.dtx v1.0e
\@iiiminipage: Extra \bgroup for color 831
\@mpfootnotetext: Extra \endgraf for color 832
\endminipage: Extra \egroup for color 832
1994-04-21 ltfinal.dtx v0.1c
General: Added comments, set the catcodes of 128–255. 1174
1994-04-22 ltfssini.dtx v2.1g
\not@math@alphabet: Message changed again 655
1994-04-23 ltfinal.dtx v0.1d
General: Check that \font@submax is still zero 1174
1994-04-24 ltoutput.dtx v1.0m
\@resethfps: Number 2 changed to \tw@ 1150
Warning changed 1150
\@specialoutput: Message changed to give more info and 'top' removed 1105
\@topnewpage: Message changed to give more info 1104
Warning message removed as it will be generated later 1103
General: Changed \@normalsize to \normalsize. 1087

Corrected unverbed commands in documentation. 1087
Removed some long lines and other aesthetic changes. 1087
Warning messages changed/corrected. 1087
1994-04-24 ltpictur.dtx v0.1b
General: Removed surplus spaces after `\hbox to` in several cases 862
1994-04-25 ltclass.dtx v0.3h
General: Removed spurious extra ‘.’s at the end of error messages 978
1994-04-25 ltfloat.dtx v1.0e
`\@largefloatcheck`: Changed warning message to give more info 920
Command added 920
General: Changed warning messages 911
Removed obsolete tracing code . . 911
1994-04-27 ltfsstrc.dtx v2.3e
General: Corrected item that was forgotten in last change. 572
1994-04-28 lterror.dtx v1.0j
`\@inmatherr`: Macro added 398
1994-04-28 lterror.dtx v1.1c
`\@inmatherr`: Replaced `\noexpand` with `\protect`. 398
1994-04-28 ltfssdcl.dtx v2.1e
General: Removed all `\uppercase` in hex num parsing macros 602
1994-04-28 ltlists.dtx v1.0c
`\item`: Replaced `\@ltxnomath` by `\@inmatherr` 813
1994-04-28 ltpictur.dtx v0.1c
`\@multiput`: (DPC) Macro added . . 866
General: bezier curves added 889
`\multiput`: (DPC) Ignore spaces between)(. 865
`\picture`: (DPC) Ignore spaces before (. 864
1994-04-28 ltplain.dtx v1.0g
General: Turn off overfull box tracing in log 25
1994-04-29 ltclass.dtx v1.0a
General: Change version number to 1 (no other change) 978
1994-04-29 ltmiscen.dtx v1.0f
`\@verbatim`: `\leavevmode` added . . 778
Change to `\everypar` added 778
1994-04-29 ltoutenc.dtx 1.4a
General: Removed `\EncodingSpecific`. Renamed all the commands. Added `\DeclareTextGlyph` and `\UndeclareTextCommand`. 470
Removed Rokicki’s OT1 variant encoding. Moved the driver to the top. 469
1994-04-30 ltfntcmd.dtx v3.3b
General: Documentation up-dated and tidied 686
Prefix frag@ changed to frag in `\@protecteddef` 686
Title changed 686
Warning changed to info message in `\@protecteddef` 686
1994-04-30 ltoutput.dtx v1.0n
`\@activechar@info`: `\@activechar@warning` changed to `\@activechar@info` 1114
`\@combinedblfloats`: Removed rule in topnewpage case 1120
`\@emptycol`: Empty column action added: `\@emptycol` 1102
`\@flsetnum`: Rogue space removed 1151
`\@specialoutput`: Cut-off point changed to 2`\baselineskip` . . . 1105
Empty column action added: `\@emptycol` 1105
Extra empty column added for twocolumn case 1105
Extra empty column added for twocolumn case (wrong, see below) 1105
`\@topnewpage`: Added setting of `\col@number` 1102, 1103
Cut-off point changed to 3`\baselineskip` 1104
Empty column action added: `\@emptycol` 1104
Message changed for Frank 1104
General: `\@activechar@warning` changed to an info message. . . . 1087
Added `\col@number`. 1087
Documentation tidied. 1087
Empty column action added. . . 1087
Fixed bug from `\dblfigrule` with `\@topnewpage`. 1087
Full of floats action improved. . . 1087
`\col@number`: Added `\col@number` 1099
`\onecolumn`: Added setting of `\col@number` 1101
1994-05-01 lterror.dtx v1.0k
`\@latexerr`: (CAR) Added draft `\@latexinfo`. 394
1994-05-01 ltoutenc.dtx 1.4a
General: Added the `\a` command. . . 478
Added the `\SaveAtCatcode` and `\RestoreAtCatcode` commands. . 482

Removed the uc/lc table settings, since the T1 uc/lc table is now the default. 490
Rewrote for the new syntax. . 482, 484
1994-05-01 ltoutenc.dtx v1.4a
General: Removed Rokicki's encoding. 466
Renamed the commands, removed the `\EncodingSpecific` command. Turned all slots into decimal. Added `\a`. 466
1994-05-02 ltcntrl.dtx v1.0l
`\@break@tfor`: Macro added (from ltfiles.dtx) 387
1994-05-02 ltdefns.dtx v1.1f
`\renewcommand`: Removed surplus `\space` in error 86
`\renewenvironment`: Removed surplus `\space` in error 87
1994-05-02 ltfiles.dtx v1.0f
`\@iffileonpath`: `\@break@loop` renamed to `\@break@tfor` 459
`\@obsoletefile`: Make `\@onlypreamble` 463
1994-05-02 ltfinal.dtx v0.1e
General: Added setting the 'letter' catcodes. 1186
Added setting the 'other' catcodes. 1186
Added setting the special catcodes. 1186
Made slot 127 illegal 1186
Set all the catcodes 1174
1994-05-02 ltfinal.dtx v0.1f
General: Set the catcode of control-J. 1186
1994-05-02 ltmiscen.dtx v1.0g
General: Changed 91 to 1991 and moved some bits 760
1994-05-02 ltoutput.dtx v1.0o
`\@resethfps`: Code shortened 1150
General: Code of `\@resethfps` shortened. 1087
1994-05-03 ltbibl.dtx v1.0b
`\nocite`: Make `\nocite` issue a warning for an undefined citation key. 940
1994-05-03 ltfinal.dtx v0.1f
General: Set the catcode of control-J to be 'other', for use in messages. 1174
1994-05-03 ltfloat.dtx v1.0f
General: (CAR) Added `\@largefloatcheck` 911
Removed unnecessary braces from arguments of `\@ifnextchar` 911
`\end@dblfloat`: `\@largefloatcheck` added 919
`\end@float`: (CAR) Added `\@largefloatcheck` 918
1994-05-03 ltfssdcl.dtx v2.1f
General: Renamed `\@@DeclareMathDelimiter` to `\@DeclareMathDelimiter` 602
1994-05-03 ltlists.dtx v1.0d
`\@item`: `\hskip` changed to `\kern` . . 814
`\item`: Removed superfluous braces 813
1994-05-03 ltmiscen.dtx v1.0h
`\@centercr`: `\@badcrerr` replaced by `\@nolnerr` 774
1994-05-03 lttab.dtx v1.0d
`\@endpbox`: Use `\@finalstrut` based on depth of `\@arstrutbox` 860
1994-05-04 ltclass.dtx v1.0b
`\NeedsTeXFormat`: Changed wording of the warning 998
1994-05-04 lterror.dtx v1.0m
`\@badcrerr`: Error message removed 397
1994-05-05 ltbibl.dtx v1.0c
`\@citex`: Set switch for warning and end of run. 939
`\nocite`: Do not write page number in `\nocite` warning message. 940
Set switch for warning and end of run. 940
1994-05-05 ltfinal.dtx v0.1g
General: Added empty errhelp. . . . 1174
`\errhelp`: Set error help empty. . . 1191
1994-05-05 ltfntcmd.dtx v3.3c
`\@@math@egroup`: Corrected `\@fontswitch` and added saved versions 693
General: Corrected `\@fontswitch` . . 686
1994-05-05 ltmiscen.dtx v1.0i
General: Removed braces from ifnextchar and ifstar arguments . 760
1994-05-07 lttab.dtx v1.0c
`\@maxtab`: Changed `\@firsttab` to `\chardef` 841
Changed `\@maxtab` to `\chardef` . 841
General: Removed definition of `\+` . 837
Removed surplus braces from `\@ifnextchar` constructs 837
1994-05-08 ltfntcmd.dtx v3.3d
General: Removed `\@undefinedfonterror` 686
`\normalsize`: Removed `\@undefinedfonterror` 694

1994-05-09 ltfntcmd.dtx v3.3f
General: Replaced all \next by \@let@token and undo change 3.3e, whatever that was. 686
1994-05-10 ltdefns.dtx v1.0n
General: (ASAJ) Added \DeclareProtectedCommand. 80
Added \DeclareProtectedCommand 90
Removed braces around \@ifundefined argument. ASAJ. 86
\makeatother: Added \makeatletter and \makeatother ASAJ. 109
1994-05-10 lterror.dtx v1.0n
\@latexerr: (ASAJ) Added extra blank lines to \@latexerr. 394
1994-05-10 ltmiscen.dtx v1.0j
\@@sverb: Slight change in error message text. 781
1994-05-11 ltboxes.dtx v1.0f
\@begin@tempboxa: Use new \color@setgroup concept. 820
\@iiiminipage: Use new \color@setgroup concept. 831
\@mpfootnotetext: Use new \color@setgroup concept. 832
Use new \normalcolor and \@finalstrut. 832
General: Superfluous braces removed from several commands 819
\color@setgroup: macro added for color support 822
\endminipage: Use new \color@setgroup concept. 832
1994-05-11 ltclass.dtx v1.0c
\endfilecontents: Add checks for form feed and tab 1010
1994-05-11 ltdirchk.dtx v1.0e
General: Add \ProvidesFile as used in fd files. 4
1994-05-11 lterror.dtx v1.0o
\@latexerr: (ASAJ) Removed one of the extra blank lines to \@latexerr. 394
1994-05-11 ltlogos.dtx v1.0o
\LaTeX: Use \DeclareProtectedCommand. ASAJ. 441
\LaTeXe: Use \DeclareProtectedCommand. ASAJ. 441
1994-05-11 ltoutenc.dtx 1.5a
General: Made T1 and OT1 generate packages rather than def files. Renamed the 'package' module to 'teststy'. 469
1994-05-11 ltoutenc.dtx v1.5a
General: Reimplemented \DeclareTextCommand using \@changed@cmd and \DeclareProtectedCommand. . . . 470
Renamed the commands again. Made the encoding part of the command syntax. Added the \DeclareTextCommand interface. Used \DeclareProtectedCommand. . . . 466
\DeclareTextAccent: Reimplemented using \DeclareTextCommand. . . . 472
1994-05-11 ltspace.dtx v1.0o
\hspace: Use \DeclareRobustCommand. ASAJ. 439
1994-05-12 ltboxes.dtx v1.0g
\@finalstrut: macro added 835
\fbox: New definition, merged with \framebox 825
\framebox: Merged \fbox and \framebox 826
\normalcolor: macro added for color support 822
1994-05-12 ltdefns.dtx v1.0p
General: (ASAJ) Fixed a bug with \relax which was using \@gobble before defining it. 80
Fixed a bug with \relax which was using \@gobble before defining it. 90
1994-05-12 ltfssbas.dtx v2.1j
General: New baselinestretch concept 523
Replaced hand-protected commands by \DeclareRobustCommand defs 523
\f@linespread: New macro 533
\fontencoding: Use \DeclareRobustCommand. 531
\fontfamily: Use \DeclareRobustCommand. 532
\fontseries: Use \DeclareRobustCommand. 532
\fontshape: Use \DeclareRobustCommand. 532
\fontsize: Redefined to use \set@fontsize 533
\linespread: New macro 533
\mathversion: Use \DeclareRobustCommand. 534
1994-05-12 ltfssdcl.dtx v2.1g
General: Allow \relax as undefined command 602

Allow \relax'ed cmds to be declared 602
1994-05-12 ltfssini.dtx v2.1i
General: Moved \fontencoding to fam.dtx 634
Moved \fontfamily to fam.dtx . 634
Moved \fontseries to fam.dtx . 634
Moved \fontshape to fam.dtx . . 634
Moved \fontsize to fam.dtx . . . 634
Moved \mathversion to fam.dtx . 634
Moved \selectfont to tracefnt.dtx 634
1994-05-12 ltfsstrc.dtx v2.3f
\selectfont: Use \DeclareRobustCommand 576
1994-05-12 ltoutenc.dtx 1.5a
General: Removed the \SaveAtCatcode and \RestoreAtCatcode commands. . 482
Rewrote for the new syntax. . 482, 484
1994-05-12 ltoutput.dtx v1.0p
\@writesetup: \normalcoloradded 1114
General: \normalcoloradded in various places (DPC). 1087
1994-05-13 ltboxes.dtx v1.0h
\@arrayparboxrestore: New accent system, use \let not \def 830
1994-05-13 ltcounts.dtx v1.0f
General: Removed \@Ialph 518
Removed \@ialph 518
1994-05-13 ltdefns.dtx v1.0q
General: (ASAJ) Renamed \DeclareProtectedCommand to \DeclareRobustCommand. Removed \@if@short@command. . . 80
(ASAJ) Replaces \space by ‘ ’ in \csname. 80
Renamed \DeclareProtectedCommand to \DeclareRobustCommand. Removed \@if@short@command. Moved to after the definition of \@gobble. 90
1994-05-13 ltdefns.dtx v1.0r
General: (ASAJ) Added logging message to \DeclareProtectedCommand. 80
Added logging message to \DeclareProtectedCommand. 90
1994-05-13 ltdefns.dtx v1.0s
General: (ASAJ) Added \@backslashchar. 80
(ASAJ) Coded \@ifdefinable more efficiently. 80
Coded more efficiently, thanks to FMi. 86
1994-05-13 ltfiles.dtx LaTeX2e
\listfiles: Stop \listfiles being run twice 463
1994-05-13 ltfiles.dtx v1.0g
\document: Added execution of \every@size 445
1994-05-13 ltfinal.dtx v0.1h
General: Added package ot1enc, and defined \@acci, \@accii and \@acciii. 1174
1994-05-13 ltfinal.dtx v1.0h
General: Added output enc stuff . . 1190
1994-05-13 ltfloat.dtx v1.0g
\@footnotetext: (DPC) Add new style colour support: \normalcolor 930
(DPC) Use \@finalstrut 930
\@xfloat: (DPC) Use \normalcolor 916
1994-05-13 ltfntcmd.dtx v3.3g
General: Replaced \@protecteddef by \DeclareRobustCommand 686
1994-05-13 ltfssbas.dtx v2.1k
General: Remove File identification ‘typeout’ 523
1994-05-13 ltfssbas.dtx v2.1l
\DeclareFontEncoding: Init encoding change command 527
\define@newfont: Use \@input@ for fd files 537
1994-05-13 ltfssdcl.dtx v2.1h
General: Removed file identification typeout 602
1994-05-13 ltfssini.dtx v2.1j
General: Removed file identification typeout 634
1994-05-13 ltfsstrc.dtx v2.3g
General: Removed typeouts as \ProvidesPackage writes to log. 572
1994-05-13 ltoutenc.dtx v1.5b
General: Added \{, \} and \$. . . . 466
Renamed \DeclareProtectedCommand to \DeclareRobustCommand. 466
Replaces \space by ‘ ’ in \csname. 466
1994-05-13 ltpictur.dtx v0.1d
General: Removed surplus braces from \@if.. constructions 862
1994-05-13 lttab.dtx v1.0d
\@contfield: Colour support 843
\@startfield: Colour support 843
\@stopfield: Colour support 843
\a: moved to ltoutenc 841

1994-05-14 fontdef.dtx v2.1f
General: Removed .def files. 664
1994-05-14 ltfssbas.dtx v2.1m
`\enc@update`: Macro added 532
1994-05-14 ltfssbas.dtx v2.1n
General: Set defaults for all `\f@...` 533
`\DeclareErrorFont`: Don't set `\f@encoding` 541
`\DeclareFontEncoding`: Log if encoding is redeclared 527
Only init enc change cmd when new encoding 527
1994-05-14 ltfssini.dtx v2.1k
General: Init error font just before checking for fontdef.cfg 659
`\reset@font`: Remove surplus braces 657
1994-05-14 ltfsstrc.dtx v2.3h
`\selectfont`: Added `\enc@update` . 578
1994-05-14 ltoutenc.dtx 1.5d
General: Moved the driver to the top. 469
1994-05-14 ltoutenc.dtx v1.5c
General: Added the fontenc package 509
Added the fontenc package. 466
Fixed a bug which caused an infinite loop if `\f@encoding` was incorrectly set. 466, 470
Moved fontsmpl to its own dtx file. 466
1994-05-14 ltoutenc.dtx v1.5d
General: Rewrote `\DeclareTextCommand` to define its argument to use the current encoding by default, rather than the encoding provided to `\DeclareTextCommand`. . . . 466, 470
Tidied up the documentation. . . . 466
1994-05-14 ltoutenc.dtx v1.5e
General: Replaced `\ENC@cmd` by `\ENC-cmd`. 466
1994-05-15 ltfssbas.dtx v2.1o
General: encoding cmds changed to enc-cmd 523
1994-05-16 fontdef.dtx v2.1g
General: Removed `\DeclareFontEncoding` for ot1 and t1 and input .def files instead . . 664
1994-05-16 ltalloc.dtx v1.1a
General: (ASAJ) Split from ltinit.dtx. 382
1994-05-16 ltcntrl.dtx v1.0a
General: (ASAJ) Split from ltinit.dtx. 384
1994-05-16 ltdefns.dtx v1.1a
General: (ASAJ) Split from ltinit.dtx. 80
1994-05-16 lterror.dtx v1.1a
General: (ASAJ) Completely new error interface. 388
(ASAJ) Split from ltinit.dtx. . . . 388
1994-05-16 ltfinal.dtx v1.0i
General: moved output enc stuff to lfonts 1190
1994-05-16 ltfssbas.dtx v2.1p
`\fontsize`: Pass `\baselinstretch` not `\f@linespread` 533
`\linespread`: Remove surplus braces 533
1994-05-16 ltfssini.dtx v2.1m
`\@acciii`: Define saved versions of accents 660
1994-05-16 ltlogos.dtx v1.1a
General: (ASAJ) Split from ltinit.dtx. 441
1994-05-16 ltmath.dtx v1.0k
`\ensuremath`: Use `\DeclareRobustCommand` and add extra braces in math mode 797
1994-05-16 ltoutenc.dtx 1.5h
General: `\pounds` was still using u rather than ui shape. 482
1994-05-16 ltoutenc.dtx v1.5f
General: enc files now have uc encoding name parts (FMi) 466
Revert code so that the encoding given is used in `\DeclareTextCommand` (FMi) . . . 466
1994-05-16 ltoutenc.dtx v1.5g
General: Made fontenc.sty use the new mixed-case encoding files. 466
Removed the lowercasing of the filename. 509
1994-05-16 ltoutenc.dtx v1.5h
General: Added `\NG`, `\ng`, `\TH`, `\th`, `\DH`, `\dh`, `\DJ` and `\dj`. 466
Added `\r` (ring accent) and `\k` (ogonek) accents. 466
Fixed a bug with `\pounds`. 466
Removed `\P` from the OT1 definitions file. 466
1994-05-16 ltoutenc.dtx v1.5i
General: Fixed a bug with `\d`. . . . 466
1994-05-16 ltoutput.dtx v1.0q
`\@writesetup`: Changed setting of accents (FMi): with the new encoding setup they can use `\let`. It could also use the new internal commands? 1115
General: Changed setting of accents (FMi). 1087
1994-05-16 ltpar.dtx v1.1a
General: (ASAJ) Split from ltinit.dtx. 399
1994-05-16 ltplain.dtx v1.0h
General: Comment out encoding specific commands 31

Remove \@acci and friends again . 32
Remove unnecessary def for \item 31
\loop: Use Kabelschacht method . . . 29
\m@th: Remove unnecessary space . . . 31
1994-05-16 ltspace.dtx v1.1a
General: (ASAJ) Split from ltinit.dtx. 421
1994-05-17 ltclass.dtx v1.0e
\@use@ption: Execute option after removing from list, not before . . 994
1994-05-17 ltdefns.dtx 1.1b
General: (ASAJ) Added the \@protect@... commands. 90
1994-05-17 ltdefns.dtx v1.1b
General: (ASAJ) Added definitions for protect. 80
(ASAJ) Removed warnings and logging to lterror.dtx. 80
Added the discussion of protected commands, defined the values that \protect should have. 90
1994-05-17 ltdefns.dtx v1.1c
General: (ASAJ) Redid definitions for protect. 80
1994-05-17 lterror.dtx v1.1b
General: (ASAJ) Moved error stuff from ltdefns.dtx. 388
1994-05-17 ltfssini.dtx v2.1n
\copyright: Really add extra braces 657
\nfss@text: Added braces to allow use in subscripts 657
1994-05-17 ltmath.dtx v1.0i
General: Replaced \let by \gdef, for indirect definition. 792
1994-05-17 ltoutenc.dtx v1.5j
General: Added braces to \pounds so it works as a subscript. 466
1994-05-18 ltdefns.dtx 1.1c
General: (ASAJ) Renamed the commands, and removed one which is no longer needed. 90
1994-05-18 ltdefns.dtx v1.1c
General: Redid the discussion and definitions, in line with the proposed new setting of \protect in the output routine. 90
1994-05-18 ltfinal.dtx v0.1j
General: Corrected the lccode for d-bar. 1174
1994-05-18 ltlogos.dtx v1.1b
General: (ASAJ) Added the TEX logo. 441
(ASAJ) Made the LATEX 2ε logo use the text font '2' rather than the math font '2'. 441
1994-05-18 ltoutenc.dtx v1.5k
General: Made dotted-i produce 'i'. . 466
Removed braces from \pounds and \dollar. 466
Replaced \defaultencoding with \encodingdefault. 466
1994-05-19 ltbibl.dtx v1.1a
General: Initial version of ltbibl.dtx, split from ltidxbib.dtx 937
1994-05-19 ltcounts.dtx v1.1a
General: Extracted file from ltcntlen. 512
1994-05-19 ltdefns.dtx v1.1d
General: (RmS) Added definitions for \@namedef and \@nameuse again. . 80
1994-05-19 ltfinal.dtx v0.1k
General: Removed \makeat... . . . 1174
1994-05-19 ltidxglo.dtx v1.1a
General: Initial version of ltidxglo.dtx, split from ltidxbib.dtx 934
1994-05-19 ltlength.dtx v1.1a
General: Extract file ltlength from ltcntlen. 521
1994-05-19 ltpageno.dtx v1.1a
General: Extract file ltpageno from ltcntlen. 737
1994-05-19 ltplain.dtx v0.1k ltfinal
\showoutput: used \maxdimen not 99999 33
\showoverfull: used \@ne not 1 33
1994-05-19 ltxref.dtx v1.1a
General: Extract file ltxref from ltcntlen. 738
1994-05-20 ltdefns.dtx v1.1e
General: Changed command name from \@checkcommand to \CheckCommand. 80
\CheckCommand: Changed name from \@checkcommand to \CheckCommand. 88
1994-05-20 lterror.dtx v1.1c
General: (ASAJ) Added \@latex@info@no@line. 388
(ASAJ) Added missing full stops. 388
(ASAJ) Fixed a bug with \@inmatherr. 388
1994-05-20 ltfinal.dtx v0.1l
General: Use new font warning commands 1181
1994-05-20 ltfloat.dtx v1.0h
\@endfloatbox: Restore outer value of @nobreak switch. 920
\outer@nobreak: Macro added: default is to do nothing. 920

1994-05-20 ltfntcmd.dtx v3.3h
General: Use new error commands . 686
1994-05-20 ltfssbas.dtx v2.1q
General: Use new error commands . 523
1994-05-20 ltfsstrc.dtx v2.3i
General: Use new error command names 572
1994-05-20 ltmiscen.dtx v1.0l
`\@writefile`: Added correct setting of `\protect`. 767
1994-05-20 ltmiscen.dtx v1.0m
General: Use new warning commands 760
1994-05-20 ltoutput.dtx v1.0s
`\@writesetup`: Added setting of `\protect` during `\shipout`. . . . 1114
General: Added setting of `\protect` during `\shipout`. 1087
1994-05-20 ltpage.dtx v1.0d
`\markright`: Changed setting for `\protect`. 970
1994-05-20 ltsect.dtx v1.0c
General: Correct setting of `\protect`. 908
`\addcontentsline`: Correct setting of `\protect`. 907
1994-05-21 ltbibl.dtx v1.1b
General: Use new warning commands 937
1994-05-21 lterror.dtx v1.1d
General: (ASAJ) Made the error commands robust. 388
1994-05-21 ltfiles.dtx v1.0h
General: Use new error commands . 442
1994-05-21 ltlists.dtx v1.0f
General: Use new error commands . 801
1994-05-21 ltmiscen.dtx v1.0n
General: Use new error commands . 760
1994-05-21 ltsect.dtx v1.0d
General: Use new error commands . 898
1994-05-21 lttab.dtx v1.0f
General: Use new error commands . 837
1994-05-21 ltxref.dtx v1.1b
General: Use new warning commands 738
`\newlabel`: Use new warning commands 741
1994-05-22 ltclass.dtx v1.0f
General: Use new warning and error commands 974
1994-05-22 ltdefns.dtx v1.1f
General: Use new warning and error cmds 80
1994-05-22 lterror.dtx v1.1e
General: (ASAJ) Replaced bgroup by begingroup in error messages, to stop extra mathords creeping into math mode. 388
1994-05-22 lterror.dtx v1.2a
General: (ASAJ) Made `\GenericError`, `\GenericWarning` and `\GenericInfo` robust. 388
(ASAJ) Replaced `\\` and tilde by `\MessageBreak` and `\space`. 388
(ASAJ) Replaced `\@generic@message` and `\@generic@error` by `\GenericError`, `\GenericWarning` and `\GenericInfo`. 388
(ASAJ) Replaces `\string` by `\protect` in some messages. . . . 388
1994-05-22 lterror.dtx v1.2d
`\GenericError`: (DPC) Alternative version added for old TeXs 389
(DPC) New version using long command name. 389
1994-05-22 ltfloat.dtx v1.0i
General: Use new warning commands 911
1994-05-22 ltoutput.dtx v1.0t
General: Changed warnings and infos to new commands. 1087
1994-05-22 ltpictur.dtx v0.1e
General: Use new warning cmds . . . 862
1994-05-23 ltclass.dtx v1.0h
`\NeedsTeXFormat`: Don't stop completely when format is wrong 998
`\usepackage`: Remove argument if possible 997
1994-05-23 ltdirchk.dtx v1.0f
General: Document `\@TeXversion` . . . 1
1994-05-23 ltfsstrc.dtx v2.3j
General: Removed def of `\f@warn@break` 591
1994-05-23 ltoutput.dtx v1.0u
`\@activechar@info`: Added `\MessageBreak` 1114
`\@writesetup`: Changed resetting of `\protect` after shipout to use `\aftergroup` 1114
General: Added `\MessageBreak`. . . 1087
Changed resetting of `\protect` after shipout. 1087
1994-05-24 lterror.dtx v1.2e
`\@latex@info@no@line`: Macro added 392
1994-05-24 lterror.dtx v1.2f
General: (DPC) wrap long lines . . . 388
1994-05-24 ltfntcmd.dtx v3.3i
General: Tidying and typos fixed . . 686
1994-05-24 ltmiscen.dtx v1.0q
`\@currenvline`: Use `\@empty` as outer default 773

1994-05-25 ltdirchk.dtx v1.0g
\filename@parse: Mac parser had " typo for : 12
1994-05-25 ltfntcmd.dtx v3.3j
General: Insertion of \aftergroups to implement \nocorr moved to the end of the group 686
\check@icr: Macros added 690
\check@nocorr@: Insertion of \aftergroups moved and defaults set up for efficiency 690
\DeclareTextFontCommand: \expandafter inserted 688
Insertion of \aftergroups moved 688
1994-05-25 ltoutput.dtx v1.0v
General: Extra documentation. . . . 1087
1994-05-25 ltsect.dtx v1.0e
\@dottedtocline: Put braces around argument 4 (the actual toc entry) to avoid font (and possibly other) changes leaking out to the leaders. 909
1994-05-25 ltthm.dtx v1.0c
General: Modify documentation . . . 894
1994-05-25 ltvers.dtx v1.0d
General: Remove PRELIMINARY TEST RELEASE from startup banner (spring is here) 37
1994-05-25 ltxref.dtx v1.1c
General: Modify documentation . . . 738
1994-05-26 ltfiles.dtx LaTeX2e
\@missingfileerror: Modify message format 461
1994-05-26 ltlogos.dtx v1.1c
General: Remove \SLiTeX logo 441
1994-05-26 ltmiscen.dtx v1.0r
General: \literal removed 784
1994-05-26 ltplain.dtx v1.1m
\iterate: (CAR) added \long 29
\underbar: (CAR/FMi) changed to use box \tw@ 31
1994-05-26 ltplain.dtx v1.1p
\underbar: (DPC) changed to use \sbox 31
1994-05-29 ltfssdcl.dtx v2.1j
General: Use new error commands . 602
1994-05-31 ltfinal.dtx v1.0n
General: Renamed lthyphen.* to lthyphen.*. 1174
1994-06-01 ltboxes.dtx v1.0i
\@frameb@x: Macro added. 827
\@iframebox: New version, so \width is correct in \framebox 826
\fbox: New version, using \@frameb@x 825
\framebox: New version, so \width is correct in \framebox 826
1994-06-01 ltlogos.dtx v1.1d
\LaTeX: Add \m@th to force math size calculations 441
1994-06-01 ltoutput.dtx v1.0w
General: Tidied up typesetting. . . 1087
1994-06-08 ltfinal.dtx v1.0m
General: Add patch file system . . . 1190
1994-06-09 ltfinal.dtx v1.0n
General: For TEX2, do not set codes for higher half of character table. 1179, 1187
1994-06-09 ltfntcmd.dtx v3.3k
General: Tidying and typos fixed in documentation 686
1994-06-18 ltfntcmd.dtx v3.3l
General: Added check for empty text 686
\check@nocorr@: Added check for empty text 690
1994-06-22 ltfntcmd.dtx v3.3m
General: Removed space from \nfss@text 686
Renamed \check@nocorr 686
\check@nocorr@: Renamed \check@nocorr to \text@command to improve \long error message . 690
\DeclareTextFontCommand: Removed space from \nfss@text 688
1994-06-22 ltmath.dtx v1.2t classes
\mathindent: Set \mathindent at the end of the class instead of at begin document 798
1994-07-20 ltlogos.dtx v1.1e
\LaTeX: Save a few tokens 441
\LaTeXe: Save a few tokens 441
1994-07-20 ltpage.dtx v1.0h
\sloppy: Save a few tokens 972
1994-09-16 ltfssbas.dtx v2.1s
\nfss@catcodes: Reset [and] as well, just in case 538
1994-10-07 ltoutenc.dtx v1.5l
General: Moved the ogonek accent. . 466
1994-10-11 ltdirchk.dtx v1.0h
\@TeXversion: Check for TeX3.14 . . 13
General: Modify all of ltxcheck again 13
1994-10-12 ltsect.dtx v1.0f
General: Doc. typos 898
1994-10-14 fontdef.dtx v2.2a
General: New coding 662
1994-10-14 ltfssini.dtx v2.2a
General: New coding for cfg files . . . 634
1994-10-14 ltmiscen.dtx v1.0s
General: Move math to other file . . 760

1994-10-14 ltplain.dtx v1.1a
General: Moved code to other files. . . 14
1994-10-15 ltfssbas.dtx v2.1t
\extract@alph@from@version: Warn if math alpha is used outside math 546
1994-10-18 ltboxes.dtx v1.0j
\@frameb@x: \leavevmode added . . 827
\@iframebox: \leavevmode moved to \@frameb@x 826
\@parboxto: Macro added to remove misuse of \@empty 828
General: stuff from ltpatch done . . . 819
\fbox: \long added 825
\mbox: \long added 820
\sbox: \long added 824
1994-10-18 ltclass.dtx v1.0j
General: Move \listfiles to ltfiles.dtx 974
1994-10-18 ltdefns.dtx v1.2a
\@star@or@long: macro added 83
General: Add extra test for \endgraf 80
Add star-forms for all commands . 80
\renew@environment: reset end command 87
1994-10-18 ltfiles.dtx v1.0i
\listfiles: code moved here from ltclass 463
1994-10-18 ltoutenc.dtx v1.5l
General: Added new definitions of \patterns and \hyphenation. . . 478
1994-10-18 ltoutenc.dtx v1.5m
General: Added new definitions of \patterns and \hyphenation. . . 466
1994-10-18 ltsect.dtx v1.0g
\@dottedtocline: Added \normalcolor for page number . 909
General: Added \normalcolor 898
1994-10-19 ltfssbas.dtx v2.1t
\DeclareFontEncoding: Add missing \relax. 527
1994-10-23 ltfsstrc.dtx v23.k
\every@math@size: Renamed to \every@math@size 581
1994-10-23 ltmath.dtx v1.0l
\@eqnnum: Added \normalcolor since \eqno introduces a subgroup of the displayed math group 794
\ensuremath: Remove extra braces: but see p 168 of Leslie's book . . 797
1994-10-24 ltboxes.dtx v1.0k
\fbox: Inner braces added (to fix latex/1061) 825
1994-10-25 fontdef.dtx v2.2c
General: Added OMSenc.def 664
1994-10-25 ltboxes.dtx v1.0l
\@isavepicbox: missing percent (moved from ltpatch) 824
1994-10-25 ltdefns.dtx v1.2b
General: Documentation improvements 80
1994-10-25 ltoutenc.dtx 1.6a
General: Added \textdollar, \textlbrace, \textrbrace, \textsterling, \textunderline. 484
Removed \textlbrace, \textrbrace, \textunderline to give them their proper names. . . 484
1994-10-25 ltoutenc.dtx v1.6a
General: Added \ProvideTextCommand, \UseTextSymbol, \UseTextAccent, \DeclareTextSymbolDefault, \DeclareTextAccentDefault, \DeclareTextCommandDefault, and \ProvideTextCommandDefault. . 466
Added the \Provide commands, and the default definitions. 470
Added the defaults. 478
Added the files OT1enc.def, T1enc.def and OMSenc.def. 478
Added the OMS encoding. 490
1994-10-27 ltoutenc.dtx 1.6b
General: Added \textasciicircum \textasciitilde \textbackslash \textbar \textbraceleft \textbraceright \textcompwordmark \textemdash \textendash \textexclamdown \textgreater \texthyphenchar \texthyphen \textless \textquestiondown \textquotedblleft \textquotedblright \textquotedbl \textquoteleft \textquoteright \textunderscore \textvisiblespace 484
Added: \textemdash \textendash \textexclamdown \texthyphenchar \texthyphen \textquestiondown \textquotedblleft \textquotedblright \textquoteleft \textquoteright 482

1994-10-27 ltoutenc.dtx v1.5d
General: Rewrote \DeclareTextSymbol to define its argument to use the current encoding by default, to fit with \DeclareTextCommand. 470
1994-10-27 ltoutenc.dtx v1.6b
General: Added \textbackslash. . . 490
Added more defaults for OT1. . . 478
Removed the enc.def files 466
Removed the files OT1enc.def, T1enc.def and OMSenc.def. 478
Renamed \textlbrace to \textbraceleft and \textrbrace to \textbraceright. 490
1994-10-29 ltmath.dtx 1.0m
General: ASAJ: Added \DeclareMathOperator. 785
ASAJ: Tidied up documentation. 793
1994-10-29 ltmath.dtx v1.0m
General: ASAJ: Added \mathellipsis, \mathdollar and \mathsterling. 792
ASAJ: Removed \dag, \ddag. . . . 792
ASAJ: Renamed \S and \P to \mathsection and \mathparagraph and made them \mathchardefs. 792
1994-10-29 ltoutenc.dtx v1.6c
General: Added commands like \dots for use in text and math. 478
Renamed \P, \S, \dag and \ddag to \textparagraph, \textsection, \textdagger and \textdaggerdbl. 466
1994-10-30 ltdefns.dtx v1.2c
\@onelevel@sanitize: Macro added 109
General: (CAR)\@onelevel@sanitize added 80
1994-10-30 ltdefns.dtx v1.2f
General: (DPC)\newwrite's moved to ltfiles 80
1994-10-30 ltmath.dtx v1.0n
General: ASAJ: Moved the new commands to ltoutenc. 792
1994-10-30 ltoutenc.dtx v1.6d
General: Added \DeclareTextCompositeCommand. 466
Added \textcircled. . 466, 480, 490
Added \t. 480
Added math commands. 466
Added OML encoding. 466, 480
Added the OML encoding. 491
Made \textless and \textgreater come from OML. 480
Moved math commands here from ltmath. 482
Removed \textregistered. 480
Rewrote \copyright to use \textcircled. 480
1994-10-31 fontdef.dtx v2.2d
General: Added OMLenc.def 664
1994-10-31 fontdef.dtx v2.2e
General: ... and moved further down 664
1994-10-31 ltfloat.dtx v1.1a
\@dblfloat: Major changes since two-column and one-column cases merged 914
\@dblflset: Macro added 914
Major changes to parameter parsing, setting of local variables, etc; two-column and one-column cases merged; space hacks moved 914
\@endfloatbox: (DPC/CAR) Extra box added to remove colour resetting from vmode 920
\@floatboxreset: Macro added . . . 918
\@footnotetext: (DPC/CAR) Move colour setting to output routine . 930
\@savemarbox: (DPC/CAR) Extra box added for colour 924
\@setfps: Macro added 915
\@xdblfloat: Macros removed: \@dbflt, \@xdblfloat 920
\@xfloat: (DPC/CAR) Extra box added to remove colour resetting from vmode 916
Major changes, removing setting of local variables, space hacks etc; two-column and one-column cases merged 915
Reset hook added 916
\@xympar: (DPC/CAR) Extra box added since needed for floats . . . 925
\fps@dbl: Macro added 915
1994-10-31 ltoutput.dtx v1.1a
\@makecol: (DPC/CAR) Colour resetting moved to here 1111
\@topnewpage: (DPC/CAR) Extra box added to remove colour resetting from vmode 1103
(DPC/CAR) Use \color@begingroup for colour . 1103
(DPC/CAR) Use \normalcolor 1103
1994-11-02 ltoutenc.dtx v1.6d
General: Wrapped lines longer than 70 characters. 466

1994-11-03 ltclass.dtx v1.0k
General: Move \@missingfileerror to ltfiles 978
1994-11-03 ltdirchk.dtx v1.0i
General: Generate an error if latex.ltx not used with clean initex 1
1994-11-03 ltfiles.dtx v1.0j
\@missingfileerror: Move here from ltclass 461
1994-11-04 ltboxes.dtx v1.0m
\@mpfootnotetext: Added \protected@edef. ASAJ. 832
1994-11-04 ltdefns.dtx v1.2e
General: Added \set@display@protect to \typeout. ASAJ. 80
Added commands for setting and restoring \protect. ASAJ. 92
Rewrote protected short commands using \x@protect. ASAJ. 90
1994-11-04 lterror.dtx v1.2g
General: Added \set@display@protect to \Generic* commands. ASAJ. . . 388
1994-11-04 ltfiles.dtx v1.0k
\nofiles: Added setting of \protected@write, \makeindex and \makeglossary to \nofiles. ASAJ. 449
\protected@write: Macro added ASAJ. 449
1994-11-04 ltfloat.dtx v1.1b
\@footnotetext: (ASAJ) Added \protected@edef. 930
\footnotemark: Added \protected@xdef to \footnotemark. 931
1994-11-04 ltidxglo.dtx v1.1b
\@wrglossary: Added \protected@write to \@wrglossary. 936
\@wrindex: Added \protected@write to \@wrindex. 935
General: Removed \if@filesw from \makeindex. 934
\makeglossary: Removed \if@filesw from \makeglossary. 935
1994-11-04 ltmiscen.dtx v1.0t
\@writefile: Removed setting of \protect. ASAJ. 767
1994-11-04 ltoutenc.dtx v1.6f
General: Added _ 481
Added \mathunderscore. 482
1994-11-04 ltpage.dtx v1.0e
\markright: Added \@unexpandable@protect. ASAJ. 970
1994-11-04 ltsect.dtx 1.0h
\@sect: (ASAJ) Added \protected@edef. 902
General: (ASAJ) Added \protected@xdef to \thanks. . . 898
1994-11-04 ltsect.dtx v1.0h
General: Added \protected@write to \addtocontents. ASAJ. 908
\addcontentsline: Added \protected@write to \addcontentsline. ASAJ. . . . 907
1994-11-04 lttab.dtx v1.0h
\@mkpream: (ASAJ) Added \@unexpandable@protect to \@mkpream. 856
\multicolumn: (ASAJ) added \set@typeset@protect. 852
1994-11-04 ltxref.dtx v1.1d
\label: (ASAJ)Added \protected@edef 742
(ASAJ)Added \protected@write 742
1994-11-05 ltboxes.dtx v1.0n
\@mpfootnotetext: Color resetting for footnotes moved to endminipage: as for main page. 832
\color@endbox: macro added for color support 822
\color@hbox: macro added for color support 822
\endminipage: Color resetting for footnotes moved to here: as for main page. 832
1994-11-05 ltboxes.dtx v1.0o
\@mpfootnotetext: Color groups restored here. 832
1994-11-05 ltfloat.dtx v1.1c
\@dblflset: Add compatibility with old version of \@xfloat. 914
\@endfloatbox: Use new \color@hbox concept. 920
\@footnotetext: Removed \normalcolor (again) 930
\@savemarbox: Use new \color@hbox concept. 924
\@setfps: Add compatibility with old version of \@xfloat. 915
\@xfloat: Add compatibility with old version of \@xfloat: but the arguments, provided at exorbitant cost, are now completely ignored 915
Use new \color@hbox concept. . . 916

\@xympar: Use new \color@hbox concept. 925
1994-11-05 ltoutenc.dtx v1.6g
General: Added setting of \@typeset@protect to \patterns and \hyphenation. 478
1994-11-05 ltoutput.dtx v1.1b
\@topnewpage: Use new \color@hbox concept. 1103
\@writesetup: Change protect settings for new-style, protect-free aux-files. 1114
Use new \color@hbox concept. . 1114
1994-11-05 ltoutput.dtx v1.1c
\@begindvi: Added macro 1118
\@begindvibox: Added macro . . . 1099
\@writesetup: Add new \AtBeginDvi concept 1114
\AtBeginDvi: Added macro 1099
1994-11-06 ltfssbas.dtx v2.1u
\cf@encoding: New macro 533
\DeclareFixedFont: Renamed \every@size to \every@math@size. 525
1994-11-06 ltfssini.dtx v2.2b
\@setsize: Use \@typeset@protect 656
1994-11-06 ltfsstrc.dtx v2.3k
\glb@currsize: New implementation 580
\try@simples: New implementation 591
\try@size@substitution: New implementation 591
\tryis@simple: New implementation 592
1994-11-07 fontdef.dtx v2.2f
General: (DPC) Add \DeclareMathSizes declarations 669
(DPC) Updated to use \ProvidesFile 664
1994-11-07 ltfiles.dtx v1.0m
\document: Renamed \every@size to \every@math@size. 445
1994-11-07 ltplain.dtx v1.0l
\@unused: move here from ltdefns, remove duplicate \@mainaux 23
1994-11-07 preload.dtx v2.1e
General: (DPC) Updated to use \ProvidesFile 683
1994-11-09 ltboxes.dtx v1.0p
\@finalstrut: Revert \finalstrut to 2.09 equivalent (from ltpatch) . . 835
General: more color changes... . . . 819
1994-11-09 ltfssbas.dtx v2.1v
\@vpt: (DPC) macros added, from setsize.dtx 549
(DPC) reduce save stack usage latex/1742 549
1994-11-10 ltbibl.dtx v1.1c
General: Fix \nocite{*} 937
\nocite: Fix \nocite{*} 940
1994-11-10 ltmath.dtx v1.2v classes
eqnarray: Added value of \parskip to \abovedisplayskip to compensate for negative \topsep 800
1994-11-10 ltoutput.dtx v1.1e
\@writesetup: Modify \protect setting 1114
1994-11-10 ltplain.dtx v1.1b
General: (CAR) added patch to \loop. 14
\iterate: (CAR) added extra \relax 29
1994-11-11 ltspace.dtx v1.2a
\\: (DPC) Make robust 425
1994-11-12 ltfntcmd.dtx v3.3o
\normalsize: Added \MessageBreak 694
1994-11-12 ltlists.dtx v1.2b ltspace
\endtrivlist: Changed order of tests to make \@noitemerror correct: end of an era. 810
1994-11-12 ltmiscen.dtx v1.0u
center: Changed end macro to \def: safer and consistent 775
flushleft: Changed end macro to \def: safer and consistent 776
flushright: Changed end macro to \def: safer and consistent 777
1994-11-12 ltplain.dtx v1.1c
General: Comment out more encoding specific commands 31
1994-11-12 ltspace.dtx v1.2b
\addpenalty: Corrected error message 433
\addvspace: Corrected error message 432
1994-11-13 ltspace.dtx v1.2c
\addpenalty: Recorrected error message 433
\addvspace: Recorrected error message 432
1994-11-14 ltoutput.dtx v1.1f
\@begindvi: Use normal box register: why a box? 1118
\@begindvibox: Use normal box register: why a box? 1099
\@writesetup: Modify new \AtBeginDvi concept 1114
General: Removed old definition of \@testfp. 1087
1994-11-14 ltspace.dtx v1.2d
\\: (DPC) Macro modified 425

1994-11-14 lttab.dtx v1.0i
\tabularnewline: (DPC) Macro added 851
1994-11-16 fontdef.dtx v2.2h
General: (DPC) Removed \{ and \} 664
1994-11-17 ltboxes.dtx v1.0q
General: \@tempa to \reserved@a . . 819
1994-11-17 ltclass.dtx v1.0l
General: \@tempa to \reserved@a . . 974
1994-11-17 ltcntrl.dtx v1.0b
General: \@tempa to \reserved@a . . 384
1994-11-17 ltdefns.dtx v1.0g
General: \@tempa to \reserved@a . . . 80
1994-11-17 ltdirchk.dtx v1.0j
General: \@tempa to \reserved@a 1
1994-11-17 lterror.dtx v1.2h
General: \@tempa to \reserved@a . . 388
1994-11-17 ltfiles.dtx v1.0n
General: \@tempa to \reserved@a . . 442
1994-11-17 ltfinal.dtx v1.0o
General: \@tempa to \reserved@a . 1174
1994-11-17 ltfloat.dtx v1.1e
General: \@tempa to \reserved@a . . 911
1994-11-17 ltfntcmd.dtx v3.3p
General: \@tempa to \reserved@a . . 686
1994-11-17 ltfssbas.dtx v2.1w
General: \@tempa to \reserved@a . . 523
1994-11-17 ltfssdcl.dtx v2.1m
General: \@tempa to \reserved@a . . 602
1994-11-17 ltfsstrc.dtx v2.3l
General: \@tempa to \reserved@a . . 572
1994-11-17 ltmath.dtx v1.0o
General: \@tempa to \reserved@a . . 785
1994-11-17 ltmiscen.dtx v1.0v
General: \@tempa to \reserved@a . . 760
1994-11-17 ltoutenc.dtx v1.6h
General: (DPC) \@tempa to \reserved@a 466
1994-11-17 ltoutput.dtx v1.1h
General: \@tempa to \reserved@a. 1087
1994-11-17 ltpictur.dtx v1.0f
General: \@tempa to \reserved@a . . 862
1994-11-17 ltsect.dtx v1.0i
General: \@tempa to \reserved@a . . 898
1994-11-17 lttab.dtx v1.0j
General: \@tempa to \reserved@a . . 837
1994-11-18 ltboxes.dtx v1.0r
\color@vbox: macro added for color support 822
1994-11-18 ltfinal.dtx v1.0n
General: re-allow slots 127–255 . . . 1186
1994-11-18 ltfssbas.dtx v2.1x
General: (DPC) use \reserved@f not \next 523
1994-11-18 ltfssdcl.dtx v2.1m
\DeclareMathDelimiter: (DPC) \expandafter instead of \next . 626
1994-11-18 ltfsstrc.dtx v2.3m
General: \next to \reserved@f . . . 572
1994-11-18 ltmath.dtx v1.0p
\phantom: (DPC) colour support . . 787
(DPC) use \expandafter instead of \next 787
\prime@s: (DPC) use \@let@token instead of \next and \expandafter instead of \nxt . . 792
\smash: (DPC) colour support 788
(DPC) use \expandafter instead of \next 788
1994-11-21 ltfloat.dtx v1.1f
\@endfloatbox: Added reset of minipage flag 920
Corrected position of \outer@nobreak 920
\@marginparreset: Macro added . . 924
\@savemarbox: Added \@setminipage etc 924
Added resetting of size and font . 924
Changed to \color@vbox 924
Use \@setnobreak etc 924
\@setminipage: Macro added 918
\@setnobreak: Macro added 918
\@xfloat: Added \@setminipage . . 916
Added resetting of size and font . 916
Changed to \color@vbox so that large floats overflow at the bottom 916
Missing percents reinserted after 4, 8: these are not numbers. 915
Use \@setnobreak 916
\@xympar: Changed to \color@vbox 925
1994-11-21 ltoutput.dtx v1.1i
\@addtocurcol: Added \if@nobreak test before float box 1129, 1132
\@specialoutput: Added \if@nobreak test 1107
\@topnewpage: Changed to \color@vbox 1103
1994-11-22 ltfssdcl.dtx v2.1o
General: wrap long lines 602
1994-11-22 ltoutenc.dtx v1.6i
General: Corrected \dots so that there's no kerning in monowidth fonts. 466
Corrected typo with \mathunderscore. 466
Fixed empty accents. Again. . . . 466

1994-11-24 ltdefns.dtx v1.2h
\@newenv: Added test for \endgraf . . 87
1994-11-25 ltplain.dtx v1.1f
General: (DPC) Comment out lots of obsolete code 14
1994-11-26 ltfloat.dtx v1.1b
\footnote: (ASAJ) Added \protected@xdef. 930
1994-11-28 ltcntrl.dtx v1.0c
General: Documentation improvements 384
1994-11-30 ltfiles.dtx v1.0o
\@dofilelist: Macro added 465
\listfiles: Use \@dofilelist . . . 463
\nofiles: There is no \@gobblethree... 449
1994-11-30 ltfssbas.dtx v2.1y
\fontshape: Use \@current@cmd in \@@enc@update. ASAJ. 532
1994-11-30 ltmath.dtx 1.0q
General: ASAJ: \DeclareMathOperator moved to AMSLaTeX. 785
1994-11-30 ltmiscen.dtx v1.0w
\@enddocument@kernel@warnings: (DPC) Do warnings even for \nofiles 763
\enddocument: (DPC) Use \@dofilelist 762
1994-11-30 ltoutenc.dtx 1.7a
General: Redefined \a for the new scheme. 478
1994-11-30 ltoutenc.dtx v1.6g
General: Removed new definitions of \patterns and \hyphenation, since encoding-specific commands now expand in the mouth. 478
1994-11-30 ltoutenc.dtx v1.7a
General: Added new code for encoding-specific commands. These now expand in the mouth, which means that ligaturing and kerning can happen. 466
Always load the enc.def file, so that the default encoding for the commands will change. 509
Redefined \@changed@cmd to expand in the mouth. 470
Removed \@changed@x@mouth since \@changed@x now expands in the mouth. 470
Rewrote \@text@composite so it allows an empty argument, or an argument containing lots of commands. 472
1994-12-01 ltfinal.dtx v1.0p
General: Renamed lthyphen.* to hyphen.*. 1174
1994-12-01 lthyphen.dtx v1.0g
General: Rename lthyphen.ltx/cfg to hyphen.ltx/cfg 1172
1994-12-01 ltplain.dtx v1.1g
General: (DPC) More doc changes . . 14
1994-12-02 fontdef.dtx v2.2i
General: Commented out \ldots. ASAJ. 662
1994-12-02 ltfssini.dtx v2.2c
\copyright: \copyright is now in ltoutenc. ASAJ 657
1994-12-02 ltlists.dtx v1.0e
\@trivlist: RmS: Added check for looping 809
1994-12-02 ltoutenc.dtx 1.7b
General: Redefined \a properly. . . . 478
1994-12-02 ltoutenc.dtx v1.7b
General: Fixed a bug with \a. 466
1994-12-04 lthyphen.dtx v1.0h
General: Documentation edits for /1989 1172
1994-12-05 ltoutenc.dtx v1.7c
General: Added braces to \textcircled. 466
1994-12-06 ltfssbas.dtx v2.1z
\DeclareFontEncoding: use \nfss@catcodes 527
\nfss@catcodes: Added tab char as well 538
1994-12-08 ltoutenc.dtx v1.7d
General: Added \null and \sh@ft to \b and \d. 466
1994-12-08 lttab.dtx v1.0k
\@array: Add \tabularnewline . . . 850
\tabularnewline: (DPC) Made it \relax 851
1994-12-09 ltbibl.dtx v1.1d
\bibliographystyle: (DPC) Allow use in preamble. 940
1994-12-10 ltfloat.dtx v1.1g
\@dblfloat: Old version reinstated temporarily 914
\@dblflset: Macro removed temporarily 914
Old version reinstated temporarily 914
\@setfps: Macro removed temporarily 915
\@xdblfloat: Macros reinserted temporarily 920

\@xfloat: Old version reinstated temporarily 915
Sanitization added temporarily . . 915
General: Some temps reinserted temporarily 911
\fps@dbl: Macro removed temporarily 915
1994-12-10 ltfntcmd.dtx v3.3q
\@@math@egroup: Don't read arguments 693
\check@nocorr@: Use \space command for comparison 690
1994-12-10 ltfssdcl.dtx v2.1p
\document@select@group: Surround with braces (add fourth arg) . . . 608
\select@group: Surround with braces (add fourth arg) 605
1994-12-10 ltoutenc.dtx v1.7e
General: Added documentation for the OML encoding. 466
Replaced width with \@width and ditto height in vrules. 466
1994-12-14 ltoutenc.dtx v1.7f
General: Added braces to \copyright so it works unbraced in subscripts. 466
Added check for math mode in \@changed@cmd. 466
Commented out \textasciicircum, \textasciitilde, \textbackslash, \textbar, \textgreater, \texthyphenchar, \texthyphen and \textless to save memory. 466
1995-01-12 ltmath.dtx v1.2y classes
\@eqnnum: Added \normalcolor . . . 798
1995-03-03 ltoutenc.dtx 1.7g
General: Corrected an error in documentation referring to the tabular rather than the tabbing environment. 478
1995-04-02 ltfntcmd.dtx v3.3r
\@@math@egroup: Read them again to be able to add \relax. 693
1995-04-02 ltfssdcl.dtx v2.1q
\document@select@group: fix problem for pr/1275 608
\select@group: fix problem for pr/1275 605
\set@mathdelimiter: fix pr/1329 . . 628
1995-04-02 ltfssini.dtx v2.2d
\not@math@alphabet: add \noexpand to second part of message 655
1995-04-21 ltclass.dtx v1.0m
\DeclareOption*: Made long /1498 991
\endfilecontents: Close input check stream: latex/1487 1010
1995-04-21 ltfinal.dtx v1.0q
General: Allow initial patch level 0 1190
1995-04-21 ltoutenc.dtx v1.7h
General: Added \null \k latex/1274 466
1995-04-22 ltfiles.dtx v1.0p
\includeonly: Allow blanks in argument 450
1995-04-22 ltmiscen.dtx v1.0x
General: Removed extra def of \@gobble 760
1995-04-23 ltsect.dtx v1.0j
\addcontentsline: Use \contentsline internally. 907
1995-04-24 ltbibl.dtx v1.1e
\@citex: Add \mbox to undefined case: latex/1239. 939
1995-04-24 ltbibl.dtx v1.1f
\bibcite: Make \@onlypreamble /1388. 938
1995-04-24 ltcntrl.dtx v1.0d
\@for: Don't expand second argument with \edef: /1317 (DPC) 387
1995-04-24 ltoutput.dtx v1.1j
\fl@tracemessage: Do not add to kernel unless 'trace' specified . . 1146
1995-04-24 ltoutput.dtx v1.1l
\@begindvibox: Add \vbox latex/1392 1099
\@writesetup: Reset \\ latex/1451 (DPC) 1115
1995-04-24 ltpage.dtx v1.0f
\fussy: reset \emergencystretch latex/1344 973
1995-04-24 ltplain.dtx v1.1h
\newlanguage: Remove remaining \outer declarations. 17
1995-04-24 ltxref.dtx v1.1e
\newlabel: Make \@onlypreamble for /1388. 741
1995-04-25 ltdefns.dtx v1.2i
\@check@c: Make \long for latex/1346 88
\new@environment: Parse arguments slowly but safely /1507 86
1995-04-25 ltfiles.dtx v1.0q
\document: Removed execution of \every@size latex/1407 445
1995-04-25 ltsect.dtx v1.0k
\@dottedtocline: Added \hbox around dots. 909

1995-04-27 ltboxes.dtx v1.0s
\@frameb@x: Move \leavevmode for graphics/1512 827
\@iframebox: Move \leavevmode for graphics/1512 826
\@iirsbox: Move \leavevmode for graphics/1512 835
\@irsbox: Move \leavevmode for graphics/1512 835
\fbox: Move \leavevmode for graphics/1512 825
\raisebox: Move \leavevmode for graphics/1512 834
1995-04-27 ltfiles.dtx v1.0r
\document: Added \global to support groups in hook 446
1995-04-27 ltmiscen.dtx v1.0y
\enddocument: \@checkend moved after hook 761
1995-04-27 ltplain.dtx v1.1i
General: Move \hang and \textindent to latex209.def 31
1995-04-29 ltcntrl.dtx v1.0e
General: Moved init of \protect to ltdefns.dtx 387
Removed unused defs for \@setprotect and \@resetprotect 387
1995-04-29 ltdefns.dtx v1.2j
\protect: Init \protect here 92
1995-04-29 ltpar.dtx v1.1b
General: (TO) Comments clean-up. 399
1995-05-02 ltsect.dtx v1.0l
\@dottedtocline: Don't reset to \rmfamily 909
1995-05-03 ltsect.dtx v1.0m
General: TO: Promoted documentation to doc.sty standard 898
1995-05-06 ltsect.dtx 1.0n
\@seccntformat: Use \quad instead of \hskip 904
\@sect: Added \relax after \@seccntformat just in case . . . 902
1995-05-07 ltboxes.dtx v1.0t
General: Use \hb@xt@ 819
1995-05-07 ltdefns.dtx v1.2k
\hb@xt@: Macro added 81
1995-05-07 ltmath.dtx v1.0r
General: Use \hb@xt@ 785
1995-05-07 ltoutput.dtx v1.1m
General: Use \hb@xt@. 1087
1995-05-07 ltpictur.dtx v1.0g
General: Use \hb@xt@ 862
1995-05-07 ltplain.dtx v1.1j
General: Use \hb@xt@ 14
1995-05-07 ltsect.dtx v1.0o
General: Use \hb@xt@ 898
1995-05-07 lttab.dtx v1.0l
General: Use \hb@xt@ 837
1995-05-08 ltbibl.dtx v1.1g
\@citex: Use \@firstofone 939
\bibitem: Removed unnecessary braces 938
\nocite: Use \@firstofone 940
1995-05-08 ltdefns.dtx v1.2k
\typein: Use \@firstofone 82
1995-05-08 ltdefns.dtx v1.2l
\typein: Remove unnecessary braces 82
Replace \def by \let 82
1995-05-08 ltfsstrc.dtx v2.3n
\ifnot@nil: Use \@firstofone . . . 585
1995-05-11 fontdef.dtx v2.2j
General: Updates to some plain macros 662
1995-05-12 ltclass.dtx v1.0n
\DeclareOption*: Use \toks@ to remove need to double hash /1557 991
1995-05-12 ltfloat.dtx v1.1h
\@footnotemark: Add \nobreak to allow hyphenation. latex/1605 . . 932
1995-05-12 ltpictur.dtx v1.0h
\pictur@: Macro added for latex/1355 864
1995-05-12 ltvers.dtx v1.0e
General: Add autoload docstrip guards 37
Check for format older than 1 year 37
1995-05-13 ltfsstrc.dtx v2.3o
General: Use single hash mark in \DeclareOption 573
1995-05-16 ltfloat.dtx v1.1i
\@makefnmark: Now use \textsuperscript. 929
\textsuperscript: Command added./pr1503 929
\thefootnote: Streamlined parts of code. 928
1995-05-17 ltboxes.dtx v1.0u
\@irsbox: Removed surplus braces . 835
1995-05-17 ltdefns.dtx v1.0o
\g@addto@macro: Make long for latex/1522 112
1995-05-17 ltlists.dtx v1.0g
\@item: Removed surplus braces . . . 815
\@nbitem: Removed surplus braces . 815
enumerate: Use \thr@@ and remove surplus braces 817
itemize: Use \thr@@ 817

1995-05-18 ltfloat.dtx v1.1j
\\@makefnmark: Added \\normalfont. 929
\\thempfootnote: Added \\itshape. . 928
1995-05-19 ltpictur.dtx v1.1a
General: Support autoloading feature 862
1995-05-20 ltcounts.dtx v1.1b
\\@definecounter: Streamlined code 514
\\@fnsymbol: Allowing both text and math 519
\\fnsymbol: Streamlined code 518
1995-05-20 ltcounts.dtx v1.1c
\\@definecounter: And do it right . 514
1995-05-20 ltfloat.dtx v1.1k
\\@makefnmark: Moved \\normalfont back and use \\@textsuperscript 929
Moved \\normalfont to \\textsuperscript 929
\\textsuperscript: Use \\normalfont. 929
1995-05-21 ltfssdcl.dtx v2.1t
\\DeclareMathRadical: Allow for undefined cs names 629
1995-05-21 ltlists.dtx v1.0f
General: Moved to doc.sty standard 801
1995-05-21 ltmath.dtx v1.0r
\\@sqrt: Use \\sqrtsign 795
General: Remove \\mathhexbox from this file 790
Update some plain macros 785
\\lefteqn: Use \\rlap 797
\\r@@t: Use \\sqrtsign instead of \\sqrt 787
1995-05-21 ltoutenc.dtx v1.7h
\\@inmathwarn: Added several \\@onlypreamble 470
1995-05-21 ltoutenc.dtx v1.7j
General: Updated some plain macros 482
1995-05-21 ltplain.dtx v1.1j
General: Moved some code to other files 14
1995-05-22 ltplain.dtx v1.1k
General: Definitions of \\footins and \\footnoterule moved to ltfloat. . 33
1995-05-22 lttab.dtx v1.1a
General: Support autoloading feature 837
1995-05-23 ltfssini.dtx v2.2e
\\newfont: Font assignment made local again. 656
1995-05-24 ltdefns.dtx v1.1l
\\newif: (DPC) New implementation . 87
1995-05-24 ltdefns.dtx v1.2m
\\typein: (DPC) New implementation 82
1995-05-24 ltfloat.dtx v1.1l
\\@textsuperscript: Command added. 929
General: Moved definition of \\footins and \\footnoterule from ltplain. 928
\\textsuperscript: Use \\@textsuperscript 929
1995-05-24 ltfssbas.dtx v3.0a
General: (DPC) Make file from previous file, fam.dtx 1995/05/20 v2.2d 523
\\mathgroup: (DPC) No need to redefine \\newfam as not outer . . 523
1995-05-24 ltfsscmp.dtx v3.0a
General: (DPC) Make file from previous file, fam.dtx 1995/05/20 v2.2d 597
1995-05-24 ltfssdcl.dtx v3.0a
General: (DPC) Make file from previous file, latint.dtx 1995/05/21 v2.1t 602
1995-05-24 ltfssini.dtx v3.0a
General: (DPC) Make file from previous file, lfonts.dtx 1995/05/23 v2.2e 634
\\cal: (DPC) Remove definition . . . 660
\\mit: (DPC) Remove definition . . . 660
1995-05-24 ltfsstrc.dtx v3.0a
General: (DPC) Make file from previous file, tracefnt 1995/05/16 v2.3o 572
1995-05-24 ltfsstrc.dtx v3.0b
General: (DPC) Fix \\ProvidesFile usage 572
1995-05-25 ltclass.dtx v1.0p
\\endfilecontents: Delete \\filec@ntents after preamble . 1010
1995-05-25 ltfilehook.dtx v1.0t
\\unqu@tefilef@und: (CAR) added \\long 1046
1995-05-25 ltfiles.dtx v1.0s
\\document: Added check for \\topskip zero 446
1995-05-25 ltfiles.dtx v1.0t
\\@iffileonpath: (CAR) added \\long 459
\\document: Corrected typo 446
\\IfFileExists@: (CAR) added \\long 457
\\nofiles: (CAR) added \\long 449
\\protected@write: (CAR) added \\long 449
1995-05-25 ltfloat.dtx v1.1m
\\@savemarbox: (CAR) Resettings moved to hook 924
\\@xfloat: (CAR) Resettings moved to hook 916

1995-05-25 ltlists.dtx v1.0i
\endtrivlist: Macros moved from ltspace.dtx 810
1995-05-25 ltmath.dtx v1.3c classes
\@eqnnum: replace \reset@font\rmfamily with \normalfont (PR 1578) 798
1995-05-25 ltspace.dtx v1.2f
\@vbsphack: (CAR) not used so 'removed'. 431
\@vspacer: (CAR) \@restorepar added to avoid possible infinite tail recursion caused by a typo in the argument. 435
(CAR) macros modified to be more efficient 435
General: Macros moved to ltlists.dtx 421
1995-05-26 ltdefns.dtx v1.2n
\@gobblefour: (CAR) Added \longs 89
1995-05-26 ltmath.dtx v1.0s
\@eqnnum: Removed \rmfamily (PR 1578), replaced \reset@font with \normalfont 794
1995-05-26 ltpage.dtx v1.0g
\ps@plain: removed \rmfamily (PR 1578) 970
1995-05-27 ltfssbas.dtx v3.0b
\mathgroup: (FMi) But a need to define \new@mathgroup 523
1995-06-05 fontdef.dtx v2.2k
General: Moved math commands from ltoutenc.dtx. 680
1995-06-05 ltfinal.dtx v1.0r
General: Added \MakeUppercase and \MakeLowercase. 1174
1995-06-05 ltoutenc.dtx v1.7k
\@inmathwarn: Removed \protected@cmd and replaced with explicit \noexpand. 470
General: Allowed \ProvideTextCommandDefault after the preamble. 472
Commented out \textless and \textgreater. 480
Moved math commands to fontdef.dtx. 482
Save some tokens in \textvisiblespace and \textunderscore. 480
1995-06-06 ltfinal.dtx v1.0s
General: Made \MakeUppercase and \MakeLowercase brace their argument. 1174
1995-06-09 ltoutenc.dtx v1.7l
\DeclareTextComposite: Rewrote \DeclareTextComposite to define the composite as a no-argument command rather than a two-argument command. 473
1995-06-11 ltspace.dtx v1.2g
\restorecr: (CAR) \relax added to stop silent eating of *. 440
1995-06-13 ltfinal.dtx v1.0t
General: Add patch level string more carefully 1190
Call \errorstopmode 1192
1995-06-13 ltpictur.dtx v1.1b
General: Use \ProvidesFile in autoload 862
1995-06-14 lttab.dtx v1.1b
General: Use \ProvidesFile in autoload 837
1995-06-15 ltfssbas.dtx v3.0c
General: (DPC) minor documentation changes 523
1995-06-15 ltfsscmp.dtx v3.0b
General: (DPC) minor documentation edits 597
1995-06-15 ltfssdcl.dtx v3.0b
General: (DPC) minor documentation changes 602
1995-06-19 ltbibl.dtx v1.1h
\bibcite: Call \@newl@bel so repeated keys produce better warning. 938
1995-06-19 ltclass.dtx v1.0q
\documentclass: Don't redefine \usepackage in compat mode for /1634 995
1995-06-19 ltxref.dtx v1.1e
\newlabel: Use \@newl@bel to share code with \bibcite 741
1995-06-28 ltfssini.dtx v3.0b
General: (DPC) Fix documentation typos 634
1995-06-28 ltmath.dtx v1.0t
General: minor doc edits 785
1995-07-02 ltplain.dtx v1.1n
General: Removed surplus 'by' and '=' in various places 14
\offinterlineskip: Replaced 1000 by \@m 30
\showoutput: Use \showoverfull to save space 33
\tracingall: Use \showoutput to save space 33

1995-07-03 ltdefns.dtx v1.2o
\set@typeset@protect: Use \@typeset@protect for init 92
1995-07-03 ltfntcmd.dtx v3.3s
\t@st@ic: Use clean interface for jump 692
1995-07-05 ltfntcmd.dtx v3.3s
\t@st@ic: Renamed from \test@next 692
1995-07-05 ltspace.dtx v1.2h
\@gnewline: Use \break 427
\@no@pgbk: Macro replaces \@pgbk and \@nopgbk 425
\nopagebreak: Reimplemented both using \@no@pgbk 424
1995-07-09 ltcntrl.dtx v1.0f
\@iforloop: Reimplemented using Kabelschacht method 387
\@iwhiledim: Reimplemented using Kabelschacht method 385
\@iwhilenum: Reimplemented using Kabelschacht method 385
\@iwhilesw: Reimplemented using Kabelschacht method 385
\@tfor: Reimplemented using Kabelschacht method 387
1995-07-09 ltlists.dtx v1.0j
enumerate: Use \expandafter 817
itemize: Use \expandafter 817
1995-07-12 ltpictur.dtx v1.1d
General: allow 2e commands in 209 mode. latex/1737 862
1995-07-13 ltdefns.dtx v1.0p
General: Updates to documentation . 80
1995-07-13 ltfiles.dtx v1.0u
General: Updates to docu 442
1995-07-13 ltfssbas.dtx v3.0d
\@@defaultsubs: macro added 543
\@defaultsubs: macro added 543
General: minor documentation changes 523
\wrong@fontshape: Change a macro not a switch to flag default font substitutions 542
1995-07-13 ltmiscen.dtx v1.0z
\@centercr: Use \nobreak 774
\@enddocument@kernel@warnings: Use \@defaultsubs instead of switch 763
\@writefile: Added missing percent and use \relax in the THEN case 767
\@xobeysp: Use \nobreak 777
General: Improve Documentation . . 760
\enddocument: Set \@setckpt to \@gobbletwo instead of defining it by hand 761
Shorten redefinition of \bibcite and \newlabel 762
1995-07-14 ltbibl.dtx v1.1i
\bibcite: Remove \@onlypreamble so still defined in new \enddocument 938
1995-07-14 ltxref.dtx v1.1g
\newlabel: Remove \@onlypreamble so still defined in new \enddocument 741
1995-07-19 ltfssini.dtx v3.0d
General: (DPC) TeX2 support 660
1995-07-20 ltboxes.dtx v1.0v
\@isavebox: Use \sbox 824
\@isavepicbox: Use \sbox 824
1995-07-21 ltoutput.dtx v1.1o
\@writesetup: Command added . . 1114
New, experimental, versions: need in-lining 1114
1995-08-09 ltmath.dtx v1.0u
General: Added code for class options leqno and fleqn 798
1995-08-11 ltlength.dtx v1.1b
General: Doc typos fixed for latex/753 521
1995-08-16 ltcntrl.dtx v1.0g
\@break@tfor: Made long 387
\@forloop: Made defs long 387
\@fornoop: Made defs long 387
\@iforloop: Made defs long 387
\@iwhiledim: Made defs long 385
Removed \@whilenoop 385
\@iwhilenum: Made defs long 385
Removed \@whilenoop 385
\@iwhilesw: Removed \@whileswnoop 385
\@tfor: Made defs long 387
1995-08-16 ltfiles.dtx v1.0v
\document: set \@maxdepth 446
set \do globally 446
set \topskip globally 446
1995-08-24 ltfssbas.dtx v3.0f
General: Added autoload code 523
1995-08-24 ltfsstrc.dtx v3.0c
General: Macro \gobble@font@spec removed 586
\tryis@simple: 593
1995-08-25 ltoutput.dtx v1.1p
General: Support autoloading feature (FMi). 1087
1995-09-01 lterror.dtx v1.2i
General: Add autoload support . . . 388
1995-09-01 ltplain.dtx v1.1m
\empty: Use \let to save space 27
\I: Use \let to save space 27

1995-09-14 ltplain.dtx v1.1o
General: Moved \multispan to lttab.dtx 14
1995-09-14 lttab.dtx v1.1c
\cline: (DPC) New implementation 860
1995-09-15 ltfssini.dtx v3.0e
General: (DPC) Modify TeX2 message 660
1995-09-19 ltmiscen.dtx v1.1a
\verb: Put \@noligs after \verbatim@font where it belongs. 783
1995-10-01 ltfiles.dtx LaTeX2e
\@addtofilelist: Macro added . . . 463
1995-10-02 ltdefns.dtx v1.2q
\@ifnch: Use \@let@token for internal/924, save \reserved@e . 108
\@ifnextchar: Use \@let@token . . 107
\@protected@testopt: Macro added . 85
\@testopt: Macro added 84
\@xargdef: New implementation, using \@test@opt 84
1995-10-03 fontdef.dtx v2.2l
General: \@@sqrt from patch file for /1701 662
1995-10-03 ltdefns.dtx v1.2r
\typein: Add missing \@typein for /1710 (from patch file) 82
1995-10-03 ltpictur.dtx v1.1e
General: New autoload code 862
1995-10-04 ltfssbas.dtx v3.0g
General: Modify autoload code 523
1995-10-04 ltfsstrc.dtx v3.0d
General: (DPC) Modify autoload code 572
1995-10-04 lttab.dtx v1.1d
General: Modify autoload support . 837
1995-10-06 ltfiles.dtx v1.0w
\@missingfileerror: Autoload error 461
1995-10-09 lterror.dtx v1.2j
General: Modify autoload support . 388
1995-10-09 ltoutenc.dtx v1.7m
\@inmathwarn: Autoload error 471
1995-10-10 ltfssbas.dtx v3.0h
\showhyphens: Use \normalfont and make colour safe, and autoloadable 547
1995-10-10 ltfssdcl.dtx v3.0c
\non@alpherr: (DPC) autoload error message 606
1995-10-10 ltplain.dtx v1.1r
General: Autoload tracing code 14
1995-10-10 ltthm.dtx v1.0f
General: Make \newtheorem 'only preamble' 894
1995-10-11 ltoutput.dtx v1.1r
\clearpage: Added a check so that it does not lose the argument of \twocolumn[...] 1100
1995-10-16 ltbibl.dtx v1.1j
\cite: (DPC) Make robust 938
1995-10-16 ltboxes.dtx v1.0w
General: Clarify makebox description 819
1995-10-16 ltdefns.dtx v1.2u
\@ifstar: (DPC) New implementation, for /1910 108
\new@command: (DPC) Use \@testopt /1911 84
\new@environment: (DPC) Use \@testopt /1911 86
\typein: (DPC) Use \@testopt /1911 82
1995-10-16 ltfssini.dtx v3.0f
\reset@font: Added \relax after \usefont, as the latter eats up spaces. 657
1995-10-16 ltmath.dtx v1.0y
\@yeqncr: (DPC) Use \@testopt /1911 796
\sqrt: (DPC) Make robust /1808 . . 795
1995-10-16 ltspace.dtx v1.2j
\nolinebreak: (DPC) Use \@testopt /1911 424
\nopagebreak: (DPC) Use \@testopt /1911 424
1995-10-16 ltthm.dtx v1.0g
General: Revert to previous \newtheorem behaviour 894
1995-10-17 ltclass.dtx v1.0r
\@providesfile: Delay definition of \ProvidesFile till ltfinal 989
\ProcessOptions*: Reset \CurrentOption for graphics/1873 994
1995-10-17 ltdirchk.dtx v1.0l
General: Modify initex version of \ProvidesFile 4
1995-10-17 ltfinal.dtx v1.0v
\@providesfile: reset macro 1191
\reserved@b: reset here after the \input above 1191
1995-10-17 ltplain.dtx v1.1s
\eject: Move \superejecct to compat file 30
1995-10-17 lttab.dtx v1.1e
\@cline: (DPC) Use \@multicnt . . 860
\@multispan: (DPC) Macro added. 860
1995-10-19 ltfinal.dtx v1.0w
\@filelist: Move after \reserved@a setting 1192

1995-10-20 ltbibl.dtx v1.1k
\\@citex: Removed refundefined flag 939
\\nocite: Removed refundefined flag 940
1995-10-20 ltclass.dtx v1.0s
\\@begindocumenthook: Make setting conditional, for autoload version 1007
1995-10-20 ltfssbas.dtx v3.0i
General: (DPC) Modify autoload code, change \\undefined 523
1995-10-20 ltfsstrc.dtx v3.0e
General: (DPC) Modify autoload code 572
1995-10-22 ltfssbas.dtx v3.0j
General: (RmS) New size function macro \\genb@sfcnt needs to be disabled at \\document. 523
1995-10-22 ltfsstrc.dtx v3.0f
General: Added 'genb' and 'sgenb' size functions to support new DC font naming scheme. 572
1995-10-23 lttab.dtx v1.1f
\\@settab: (CAR)Ensure that \\@hightab increases by at most one 844
\\@startline: (CAR)Ensure that \\@nxttabmar is never larger than \\@hightab 842
\\poptabs: (CAR)Ensure that \\@curtab is never larger than \\@hightab 845
\\tabbing: (CAR)Make \\@hightab consistently a local variable 844
1995-10-24 ltfiles.dtx v1.1a
\\document: Removed multiplelabels switch 445
Removed refundefined switch . . . 445
1995-10-24 ltfssbas.dtx v3.0k
\\@@defaultsubs: macro removed . . 543
\\wrong@fontshape: Make this code inline since it happens only here 542
1995-10-24 ltmiscen.dtx v1.1b
\\@enddocument@kernel@warnings: Changed logic for producing warning messages and removed switch 763
Use \\@refundefined instead of switch 763
1995-10-24 ltxref.dtx v1.1h
\\@multiplelabels: Switch for multiplelabels removed 742
\\@newl@bel: Switch for multiplelabels replaced by inline code 741
\\@refundefined: Switch for refundefined replaced 739
\\@setref: Switch for refundefined renamed 740
\\if@multiplelabels: Macro removed 742
1995-10-25 ltalloc.dtx v1.1b
General: General doc improvements 382
1995-10-25 ltfloat.dtx v1.1n
\\@endfloatbox: (CAR) macro added: to unify code for double and single versions 920
\\end@dblfloat: (CAR) unify code for double and single versions 919
\\end@float: (CAR) unify code for double and single versions 918
1995-10-25 ltidxglo.dtx v1.1d
General: Doc cleanup 934
1995-10-25 ltsect.dtx v1.0q
\\subparagraphmark: Use \\let not \\def to save space. 906
1995-10-27 ltpictur.dtx v1.1f
General: Move initialization to kernel from autoload file 889
1995-10-31 ltboxes.dtx v1.0x
\\@finalstrut: Add \\nobreak in horiz mode to allow hyphenation. internal/1931 835
1995-11-01 fontdef.dtx v2.2m
General: add \\nfss@catcodes for internal/1932 665
1995-11-01 ltdirchk.dtx v1.0n
General: Initialise \\@addtofilelist to \\@gobble 4
1995-11-01 ltfinal.dtx v1.0x
General: (DPC) Switch meaning of \\@addtofilelist for cfg files . . 1180
1995-11-01 ltfssbas.dtx v3.0m
\\DeclareFontShape@: (DPC) Test for \\relax not \\undefined, internal/1933 524
1995-11-01 ltfssini.dtx v3.0g
General: (DPC) Switch meaning of \\@addtofilelist for cfg files . . . 660
1995-11-02 ltfssbas.dtx v3.0n
\\wrong@fontshape: (DPC) Remove extra space with \\string for latex/1676 542
1995-11-02 ltoutenc.dtx v1.7n
General: Changed internal name \\a to \\@tabacckludge to protect against redefinition by malicious users. . 478
1995-11-07 ltlists.dtx v1.0k
\\@doendpe: Enclosed \\setbox0 assignment by a group so that it leaves the contents of box 0 intact. 812

1995-11-07 ltoutenc.dtx v1.7o
General: Added \leavevmode at start of \c, otherwise the output routine might be invoked within the macro. 482
Changed \char32 to \@xxxii (two tokens less). 483
Replaced octal number 27 by decimal number 23 to protect against the quote character being active. 483
Replaced some 0's by \z@ (faster). 483
1995-11-10 ltoutput.dtx v1.1s
\@shipoutsetup: Command removed 1114
\@writesetup: Command removed 1114
In-lined 1114
1995-11-14 ltclass.dtx v1.0t
\@@unprocessedoptions: Allow empty option 1009
\@loadwithoptions: macro added . 996
\LoadClassWithOptions: macro added 996
\RequirePackageWithOptions: macro added 997
1995-11-17 ltfssbas.dtx v3.0m
\@wrong@font@char: (DPC) Macro added. latex/1676 543
\define@newfont: Redefine \typeout latex/1676 537
\wrong@fontshape: Support \@wrong@font@char latex/1676 . 542
1995-11-17 ltoutenc.dtx v1.7p
\UseTextSymbol: Support \@wrong@font@char latex/1676 . 475
1995-11-18 ltoutenc.dtx v1.7q
\UseTextSymbol: Modify message slightly 475
1995-11-21 fontdef.dtx v2.2n
General: Incorporate changed figures, as in plain.tex 678
1995-11-27 ltfssbas.dtx v3.0n
\nfss@catcodes: Reset hash, for definitions in fd files 538
1995-11-28 ltfloat.dtx v1.1n
General: documentation fixes 911
1995-11-28 ltfsstrc.dtx v3.0g
General: documentation fixes 572
1995-11-28 ltoutenc.dtx v1.7r
General: Added math mode checks to text commands. 470
doc fixes 466
Renamed \@changed@x@err to \TextSymbolUnavailable. 470
1995-11-29 ltoutenc.dtx v1.7t
General: Added \textasciicircum, \textasciitilde, \textbackslash, \textbar, \textgreater and \textless. . . 486
Added \textasciicircum, \textasciitilde, \textregistered and \texttrademark. 480
Added \textbackslash and \textbar. 479, 490
Added \textless and \textgreater. 480, 491
1995-12-01 ltoutenc.dtx v1.7u
General: Made \SS a Default, rather than having the default point to the OT1 definition. 480
1995-12-04 ltspace.dtx v1.2k
\nobreakspace: Macro added 437
1995-12-04 ltspace.dtx v1.2l
\@xobeysp: braces added to definition of tilde 438
1995-12-04 preload.dtx v2.4e
General: Ulrik Vieth. added 12pt OMS and OML preloads /1989 . 685
1995-12-05 ltdefns.dtx 1.2w
\@unexpandable@protect: Removed \unexpandable@noexpand as never used. internal/1733 90
1995-12-05 ltfiles.dtx v1.1c
\document: \ignorespaces added for latex/1933 446
1995-12-05 ltfloat.dtx v1.1n
\@textsuperscript: Use \ensuremath for latex/1984. 929
1995-12-05 ltoutenc.dtx v1.7v
\@inmathwarn: Changed \TextSymbolUnavailable text . . 471
1995-12-06 ltfssbas.dtx v3.00
\nfss@catcodes: Reset hat, for typeouts etc in fd files 538
1995-12-07 ltbibl.dtx v1.1l
\@citex: Restored name of \G@refundefinedtrue 939
1995-12-07 ltfloat.dtx v1.1m
\@textsuperscript: Move \m@th out of the \ensuremath for latex/1984. 929
1995-12-07 ltxref.dtx v1.1i
\@setref: Switch for refundefined restored 740
\G@refundefinedtrue: Renamed (back) from \G@refundefined . . 739

1995-12-11 ltoutenc.dtx v1.7w
General: Modified \copyright 480
1995-12-13 ltdefns.dtx 1.2x
\-: Documentation changed. 110
1996-01-10 ltfiles.dtx v1.1d
\@iffileonpath: Change argument handling to not require doubled hash. latex/2024 459
1996-01-20 ltidxglo.dtx v1.1e
\makeglossary: Make no-op after use pr/2048 935
\makeindex: Make no-op after use pr/2048 935
1996-01-20 ltspace.dtx v1.2m
\vspace: Made robust 435
1996-03-25 ltmath.dtx v1.1a
\@ensuredmath: Macro added for amslatex/2104 797
\ensuremath: Reimplement for amslatex/2104 797
1996-04-18 ltpage.dtx v1.0i
General: Improve documentation . . 969
1996-04-22 ltmiscen.dtx v1.1c
General: Improve Documentation . . 760
1996-04-22 ltspace.dtx v1.2n
General: Documentation Improvements 421
1996-04-22 lttab.dtx v1.1g
\@tabclassz: (DPC) Extra \hskip keeps tabcolsep in empty columns internal/2122 857
1996-04-23 ltcounts.dtx v1.1d
General: Documentation improvements 512
1996-04-24 ltfiles.dtx v1.1e
\document: (DPC) Reset \AtBeginDocument eg for latex/1297 445
1996-05-08 ltfsstrc.dtx v3.0h
\math@egroup: Use \bgroup instead of \begingroup to match a kernel change made in 1994!! 584
1996-05-09 ltfntcmd.dtx v3.3t
\check@icr: Default definitions added 690
1996-05-17 fontdef.dtx v2.2o
General: \@@sqrt removed, at last 662, 678
1996-05-17 ltfiles.dtx v1.1f
\nofiles: added \write to \protected@write for latex/2146 449
1996-05-18 ltoutenc.dtx v1.7x
General: Produce error if encoding not found. pr/2054 509
1996-05-21 ltoutenc.dtx v1.7y
General: Corrected error message (CAR) 509
1996-05-21 ltsect.dtx v1.0s
\@sect: (DPC) Added extra braces for internal/2148 902
(DPC) Moved brace to allow commands like \MakeUppercase in 6th argument. Changed \par to \endgraf to allow non-long commands. internal/2148 902
\@ssect: (DPC) Added extra braces for internal/2148 905
(DPC) Moved brace to allow commands like \MakeUppercase in 4th argument. Changed \par to \endgraf to allow non-long commands. internal/2148 905
1996-05-23 ltoutenc.dtx v1.7z
\@strip@args: \expandafter added to match other changes for latex/2133 475
\add@accent: macro added. latex/2133 472
\DeclareTextAccent: Reimplemented using \add@accent to save space latex/2133 472
\DeclareTextCompositeCommand: Modified to cope with new \add@accent command: required removal of check for one argument-command 473
1996-05-24 ltoutput.dtx v1.1t
\@specialoutput: Check that \@colroom is less than \vsize, indicating that a float has been added 1105
Cut-off point changed to 1.5\baselineskip 1105
\@topnewpage: Cut-off point changed to 2.5\baselineskip 1104
1996-05-25 ltoutput.dtx v1.1u
\@specialoutput: Correct the above check 1105
1996-06-03 ltmiscen.dtx v1.1d
\@verbatim: Exchanged the following two code lines so that \dospecials cannot reset the category code of characters handled by \@noligs. 778
General: Move setting of verbatim font and \@noligs. 760
\verb: Put setting of verbatim font after \dospecials so that \dospecials cannot reset the

category code of characters handled by \@noligs. 783
1996-06-10 ltboxes.dtx v1.0y
\@parboxto: (DPC) Changed \endgraf to \@@par 828
1996-06-10 ltsect.dtx v1.0t
\@sect: (DPC) Changed \endgraf to \@@par 902
\@ssect: (DPC) Changed \endgraf to \@@par 905
1996-06-13 ltdirchk.dtx v1.0r
General: documentation improvements mainly from internal/2174 1
1996-06-14 lttab.dtx v1.1h
\@tabclassz: (DPC) Change both\z@skip to 1sp for latex/2160 857
1996-06-22 ltspace.dtx v1.2o
General: Documentation of problems added 421
1996-07-10 ltfinal.dtx v1.0y
\toks: Free up memory from scratch registers /2213 1191
1996-07-19 ltoutenc.dtx v1.8a
\@strip@args: Use char 0 not @ as carrier for \lowercase /2197 . . . 475
1996-07-26 ltboxes.dtx v1.0z
\if@minipage: put \global into definition 830
1996-07-26 ltclass.dtx v1.0u
\@classoptionslist: made only preamble 979
\@unusedoptionlist: made only preamble 979
1996-07-26 ltdefns.dtx v1.2y
\@reargdef: third arg picked up by \@yargdef 85
\renew@command: use \noexpand instead of \string 86
use \relax in place of empty arg . 86
\renew@environment: use \relax in place of empty arg 87
1996-07-26 ltfloat.dtx v1.1n
\@endfloatbox: remove unnecessary \global before \@minipage... . 920
\@savemarbox: remove unnecessary \global before \@minipage... . 924
\@setminipage: remove unnecessary \global before \@minipage... . 918
\@setnobreak: remove unnecessary \global before \@nobreak... . . 918
1996-07-26 ltfssbas.dtx v3.0p
\@DeclareMathSizes: use faster \if test 530
\nfss@catcodes: omit \relax as not needed 538
1996-07-26 ltfssdcl.dtx v3.0e
\init@restore@version: Removed \ifrestore@version switch and replaced by \init@restore@version 606
1996-07-26 ltfsstrc.dtx v3.0i
\init@restore@glb@settings: macro added replacing \if@inmath switch 583
1996-07-26 ltlists.dtx v1.0l
\@item: Remove unnecessary \global before \@minipage... 814
Remove unnecessary \global before \@nobreak... 815
1996-07-26 ltmath.dtx v1.1b
General: Removed \global before \@ignoretrue in various places. . 785
1996-07-26 ltmiscen.dtx v1.1e
\@ignorefalse: put \global into definition 761
\begin: remove \global before \@ignore... 769
\end: remove \global before \@ignore... 771
\ignorespacesafterend: user level macro added 761
1996-07-26 ltoutput.dtx v1.1v
\@testfp: remove \global before \@test... 1149
\@xtryfc: remove \global before \@test... 1123
\@ztryfc: remove \global before \@test... 1124
General: put \global into definition 1097
remove \global before \@test... 1097
\clearpage: add number of missing percents 1100
1996-07-26 ltplain.dtx v1.1t
\sh@ft: replace \dimen\z@ by \dimen@ 32
1996-07-26 ltsect.dtx v1.0u
\@starttoc: removed \global before \@nobreak... 907
\@xsect: Removed \global before \@nobreak... 903
1996-07-26 ltspace.dtx v1.2p
\if@nobreak: put \global inside definition 427
1996-07-27 ltfssbas.dtx v3.0q
General: \if@inmath switch removed 536
1996-07-27 ltspace.dtx v1.2q
General: Further documentation of problems 421

1996-07-27 ltspace.dtx v1.2r
General: Correct documentation of problems 421
1996-08-02 ltfloat.dtx v1.1o
\@xympar: Remove \global before \@ignore... 925
1996-08-02 ltsect.dtx v1.0v
\@afterheading: Removed \global before \@nobreak... 905
1996-08-02 ltspace.dtx v1.2s
\@Esphack: Remove \global before \@ignore... 430
1996-08-25 ltfssbas.dtx v3.0r
\nfss@catcodes: Reset the acute, grave and double quote chars as well 538
1996-09-21 ltoutput.dtx v1.1w
\@writesetup: Added \@parboxrestore and made consequent deletions: wait for the howls of protest 1114
1996-09-25 ltdirchk.dtx v1.0t
General: Move ltxcheck to separate file 13
1996-09-28 ltmiscen.dtx v1.1f
\@xobeysp: Moved to ltspace.dtx . . 777
1996-09-28 ltspace.dtx v1.2t
\@xobeysp: Moved from ltmiscen.dtx and redefined to use \nobreakspace 437
1996-09-29 ltfiles.dtx v1.1g
\document: Added disabling of \@nodocument 446
1996-09-29 ltoutput.dtx v1.1x
\newpage: Checks for noskipsec and inlabel added 1101
1996-09-29 ltsect.dtx 1.0w
\@noskipsectrue: Added documentation 899
1996-09-30 ltoutput.dtx v1.1y
\newpage: Checks for noskipsec and inlabel removed pending further tests 1101
1996-10-04 ltclass.dtx v1.0v
\RequirePackageWithOptions: Reset \@unprocessedoptions for /2269 997
1996-10-05 ltfiles.dtx v1.1h
\@clubpenalty: Added setting its value 444
1996-10-08 ltfntcmd.dtx v3.3u
\DeclareTextFontCommand: Removed \check@icr when in vmode since it causes various errors (see pr/2157) 688
1996-10-21 lttab.dtx v1.1i
\@array: Use \set@typeset@protect 850
General: Moved the code associated with \@mkpream into the group provided by the box, for robustness (latex/2183) 849
\multicolumn: Make \multicolumn long (latex/2180) 852
\tabbing: Moved the \indent so that the \everypar can remove it when necessary; this is needed because the code for items in lists has changed (see pr/22111) 844
1996-10-23 ltlists.dtx v1.0m
\@item: \@nobreak... moved into the \everypar and not executed unconditionally, see above 815
\kern... changed to \setbox... 814
Added setting of \clubpenalty and set \@nobreakfalse only when necessary 815
1996-10-23 ltsect.dtx v1.0x
\@xsect: Replaced \hskip... with \setbox... as used in \@afterheading 903
1996-10-24 ltboxes.dtx v1.1a
\@arrayparboxrestore: Added local settings of flags: dangerous! 829
\@iiiminipage: Use it or lose it (@setminpage): Frank will want to lose it 831
1996-10-24 ltfloat.dtx v1.1p
\@floatboxreset: Added local settings of flags: dangerous! 918
\@marginparreset: Added local settings of flags: dangerous! 924
\@xfloat: Added \@nodocument to trap floats in the preamble 915
1996-10-24 ltoutput.dtx v1.1z
\@addtocurcol: Added \nobreak, etc as appropriate 1129, 1132
\@specialoutput: Added \nobreak as appropriate 1107
\@topnewpage: Added \@nodocument to trap \twocolumn in the preamble 1103
\newpage: Better checks for noskipsec and inlabel added, plus nobreak 1101
1996-10-25 ltlists.dtx v1.0n
\endtrivlist: Change \indent to \leavevmode 810
Reset flags explicitly 810
1996-10-25 ltoutput.dtx v1.2a
\newpage: Reset all flags explicitly 1101

1996-10-26 ltlists.dtx v1.0o
\endtrivlist: Correct typo 810
1996-10-27 ltoutenc.dtx v1.8c
\@strip@args: Removed macro . . . 473
General: Added \r A 483
Added \textasteriskcentered . . 479, 490
Corrected syntax descriptions . . . 467
Removed \aa and \AA . 479, 483, 486
1996-10-28 ltplain.dtx v1.1u
General: (CAR) More doc changes . . 14
\dotfill: Removed math mode 32
1996-10-29 ltplain.dtx v1.1v
\dotfill: Got arithmetic correct (CAR) 32
1996-10-29 ltspace.dtx v1.2u
\@gnewline: Added macro 427
\@no@lnbk: Macro replaces \@lnbk and \@nolnbk 425
\\: Corrected and rationalised code 425
\nolinebreak: Reimplemented both using \@no@lnbk 424
1996-10-31 ltfinal.dtx v1.0z
General: Added extra \lcode, hoping it does no harm in T1 (pr/1969) 1180, 1187
1996-10-31 ltlists.dtx v1.0p
\@trivlist: Added check for missing item in outer list 809
1996-10-31 ltsect.dtx v1.0y
General: Corrected and tidied documentation; removed long lines 898
1996-11-03 ltplain.dtx v1.1w
\dotfill: Saved tokens by using \hb@xt@ 32
1996-11-04 lterror.dtx v1.2m
\@nodocument: Always define \@nodocument in kernel, so that it can be cleared by \document. . . . 395
1996-11-04 ltlists.dtx v1.0q
\@trivlist: Moved check for missing item: only checked when not inlabel flag is false 809
1996-11-05 ltfiles.dtx v1.1i
\nofiles: Standard \if@nobreak test added 449
1996-11-09 ltmath.dtx v1.1c
\@ensuredmath: Made long, as it was before. /2104 797
1996-11-18 ltfssbas.dtx v3.0s
\define@newfont: (DPC) lowercase fd file names. internal/1044 538
1996-11-18 ltoutenc.dtx v1.8d
General: (DPC) lowercase external file names. internal/1044 509
1996-11-20 fontdef.dtx v2.2p
General: lowercase fd and enc.def file names /1044 662
1996-11-20 ltvers.dtx v1.0f
General: Check for old format modified /2319 37
1996-11-23 ltoutenc.dtx v1.8e
General: Corrected description 467
Extended description 468
1996-11-28 ltvers.dtx v1.0g
General: Check for old format modified /2319 37
1996-12-06 ltdirchk.dtx v1.0u
\IfFileExists: *** removed from various messages for GNU Make. internal/2338 10
1996-12-06 ltfloat.dtx v1.1r
\@caption: Call \@setminpage if needed. latex/2318 914
1996-12-06 ltfssini.dtx v3.0h
General: (DPC) Remove *** from messages internal/2338 660
1996-12-17 ltdefns.dtx v1.0w
\g@addto@macro: Use \begingroup to save making a mathord 112
1996-12-20 ltsect.dtx v1.0z
\@dottedtocline: Added \nobreak for latex/2343 909
1997-01-08 fontdef.dtx v2.2q
General: Use \DeclareMathDelimiter to set delimiter codes 672
\mathparagraph: Define using \DeclareMathSymbol 680
1997-01-08 ltfiles.dtx v1.1j
\@include: reset \deadcycles latex/2365 453
1997-01-08 ltmath.dtx v1.1d
\root: (DPC) Remove spurious space tokens from plain TEX definition /2359 787
1997-02-05 ltdefns.dtx v1.0x
\g@addto@macro: missing percent /2402 112
1997-02-21 ltlists.dtx v1.0r
\@item: \ifvoid check added for \noindent. latex/2414 814
1997-03-21 ltcounts.dtx v1.1e
\fnsymbol: Use \mathsection and \mathparagraph. latex/2445 . . . 518

1997-04-14 ltfiles.dtx v1.1k
\document: Set the document space factor defaults. latex/2404 445
\normalsfcodes: Macro added (from patch file) latex/2404 449
1997-04-14 ltoutput.dtx v1.2b
\@writesetup: Call \normalsfcodes (from patch file) latex/2404 . . . 1116
Move \label and \index (from patch file) 1116
1997-04-24 ltbibl.dtx v1.1m
\@citex: \@empty to avoid primitive error on empty cite keys. latex/2432 939
1997-04-30 ltoutenc.dtx v1.9a
General: Changed \textsc to \scshape 480
Introduced \textcopyright and modified \copyright 480
Introduced \textcopyright and modify \copyright 481
Modified \textunderscore, removing \mathunderscore 480
Modified \underscore, removing \mathunderscore 481
1997-04-30 ltoutenc.dtx v1.9b
General: Added \leavevmode to \textunderscore 480
1997-05-04 ltoutenc.dtx v1.9c
General: Added 'hex index tabs' . . . 487
Added TS1 encoding v2.2.beta . . 493
1997-05-07 ltoutenc.dtx v1.9d
General: Added \leavevmode to \textcompwordmark 480
1997-05-07 ltspace.dtx v1.2v
\newline: Made completely robust. 426
1997-05-29 ltfsstrc.dtx v3.0j
General: Replaced \\ by \MessageBreak, as suggested by Donald Arseneau. 574
1997-05-29 ltlogos.dtx v1.1f
\LaTeXe: Added \m@th so that the LaTeX 2ε logo works with non-zero values of \mathsurround. 441
1997-06-16 ltdirchk.dtx v1.0v
General: documentation improvements mainly from internal/2520 1
1997-06-16 ltfloat.dtx v1.1s
General: documentation fixes 911
1997-06-16 ltfntcmd.dtx v3.3v
General: Fix typo in documentation. 686
1997-08-05 ltoutenc.dtx v1.9e
General: Corrected order of arguments in \UseTextSymbol example. . . . 467
1997-08-29 ltoutenc.dtx v1.9f
General: Added OT4 encoding, provided by Marcin Woliński. . . 466
1997-09-09 ltdefns.dtx v1.2z
\provide@command: Use \begingroup to avoid generating math ords if used in math mode. pr/2573 88
1997-09-15 ltpictur.dtx v1.1g
\@getcirc: Warn if lines become invisible pr/2524 883
\@picture@warn: Macro added pr/2524 883
\@sline: Warn if lines become invisible pr/2524 872
1997-10-06 ltcounts.dtx v1.1f
\@Roman: Change \@Roman to be fully expandable, so that the result is written properly to files. 518
\@slowromancap: Macro added. . . . 518
1997-10-08 ltlogos.dtx v1.1h
\LaTeX: Simplify macro (force loading of suitable math fonts once). . . . 441
1997-10-10 ltclass.dtx v1.0y
\endfilecontents: \@currenvir in banner 1012
\reserved@c not \verbatim@out to save a csname 1011
Check for text before or after \end environment. latex/2636 1012
Use \@gobbletwo 1011
1997-10-17 ltfntcmd.dtx v3.3w
\check@nocorr@: Check for vertical mode moved here, from \DeclareTextFontCommand (see PR/2646). 690
\DeclareTextFontCommand: Reinstalled \check@icr as check is now done in \check@nocorr@ (see PR/2646). 688
1997-10-20 ltfinal.dtx v1.1a
\@uclclist: Removed \aa and \AA from \@uclclist as these are macros. 1188
1997-10-21 ltdefns.dtx v1.2z1
\renew@command: Use \begingroup/\endgroup rather than braces for grouping, to avoid generating empty math atom. . . . 86
1997-10-21 ltfssbas.dtx v3.0t
\define@newfont: Move \makeatletter to \nfss@catcodes. 537

\nfss@catcodes: Moved \makeatletter from \try@load@font@shape. 538
1997-11-09 ltoutput.dtx v1.2c
\@specialoutput: Remove incorrect code: only one \@emptycol is needed here 1105
\@topnewpage: Documentation of vsize check enhanced 1102
1997-11-13 ltfssdcl.dtx v3.0f
\DeclareSymbolFont: (DPC) Really update \group@list don't leave new version in \toks@. latex/2661 615
\stepcounter: (DPC) Remove as never used. (Re)defined in ltcounts 604
1997-11-19 ltfloat.dtx v1.1t
\@footnotetext: Missing percent, again 930
1997-11-19 ltoutput.dtx v1.2d
\@vtryfc: Reindent code, to be understandable(DPC). 1122
1997-11-20 ltfssdcl.dtx v3.0g
\document@select@group: (DPC) inline use of \stepcounter (faster, and saves a csname per math version as no reset list) 607
\select@group: (DPC) inline use of \stepcounter (faster, and saves a csname per math version as no reset list) 605
1997-11-23 ltoutenc.dtx v1.9g
General: Use \textperthousand, \textpertenthousand and \textfractionsolidus not \textpermill, \textpertenmill and \textfraction. /2673 493
1997-12-17 ltoutenc.dtx v1.9h
General: Added \textperthousand and \textpertenthousand . 484, 485
Added code for textcomp.sty. . . . 509
Added section. 509
Added textcomp.sty. 466
As in OT1, Added \leavevmode at start of \c, otherwise the output routine might be invoked within the macro. 485
Changed to decimal codes in \ooalign. 495
Changed to decimal codes. 491
Documentation changes and additions. 466
Example corrected, braces removed. 466
Removed default settings, see next section. 493
1997-12-19 ltoutenc.dtx v1.9i
General: Documentation corrections. 466
1997-12-20 fontdef.dtx v2.2s
General: Added documentation . . . 664
1997-12-31 ltoutenc.dtx v1.9k
General: Further correction 467
1998-01-12 ltoutenc.dtx v1.9k
General: Added \ProvidesPackage for textcomp.sty 466
Adding missing braces and \ushape. 495
1998-01-16 ltoutenc.dtx v1.9m
General: fixed decimal codes. latex/2734 491
1998-03-04 ltdefns.dtx v1.2z2
\@xargdef: Unnecessary \expandafter removed: pr/2758 . 84
1998-03-05 ltoutenc.dtx v1.9n
General: Added masc/fem ords as in pr/2579 480
1998-03-20 ltdefns.dtx v1.2z3
\@thirdofthree: Macro added 89
1998-03-20 ltoutenc.dtx v1.9o
General: Documentation added about order of decls 469
Documentation added for pr/2783 468
\UndeclareTextCommand: Macro added for pr/2783 477
1998-03-20 lttextcomp.dtx v1.9o
General: Added various \UndeclareTextCommand declarations for pr/2783 725
Load decls after defaults for speed. 725
1998-03-21 ltclass.dtx v1.0z
General: Added to documentation of filecontents 974
1998-03-21 ltclass.dtx v1.1a
\@providesfile: Allow &. Internal/2702 989
General: Correct to new onlypreamble command list 1026
1998-03-25 ltfssbas.dtx v3.0u
\showhyphens: Suppress unnecessary error when used in preamble . . . 547
1998-04-11 fontdef.dtx v2.2t
General: Added \mathring accent (pr2785) 678
1998-04-15 fontdef.dtx v2.2u
General: Use new syntax for \DeclareMathDelimiter 672

1998-04-15 ltfssdcl.dtx v3.0h
\\@xxDeclareMathDelimiter: Macro added (pr/2662) 626
1998-04-17 fontdef.dtx v2.2v
General: Reinsert symbol defs for < and > chars. 672
1998-04-18 fontdef.dtx v2.2w
General: Reinsert symbol def for / char. 672
1998-05-07 ltclass.dtx v1.1b
\\@onefilewithoptions@clashchk: Modify help message for latex/2805 1003
1998-05-18 lttab.dtx v1.1j
\\@endpbox: Use \\setlength to set \\hsize, so that the changes in the calc package apply here. 860
\\tabular*: Use \\setlength, so that calc extensions apply. 849
1998-05-20 ltfinal.dtx v1.1b
General: Set up lccodes before loading hyphenation files: pr/2639 1179
Set up uc/lccodes after loading hyphenation files: pr/2639 1187
1998-05-28 lterror.dtx v1.2n
\\@notdefinable: Added message re 'end...' pr/1555 395
1998-06-04 ltboxes.dtx v1.1c
\\@rule: Support calc-expressions . . 834
1998-06-12 ltoutenc.dtx v1.9p
General: Corrected 130 and 131, see pr/2834 496
Renamed \\textmacron pr/2840 . 497
1998-06-12 ltoutenc.dtx v1.9q
\\add@accent: Explicitly set \\spacefactor after \\accent (pr/2877) 473
1998-06-12 lttextcomp.dtx v1.9p
General: Renamed \\textmacron pr/2840 721
1998-06-18 lttab.dtx v1.1k
General: Small addition to documentation 837
1998-07-06 lttab.dtx v1.1l
General: Small correction to documentation 837
1998-08-17 ltboxes.dtx v1.1e
General: (RmS) Minor Documentation fixes. 818
1998-08-17 ltclass.dtx v1.1c
General: (RmS) Minor documentation fixes. 974
1998-08-17 ltdirchk.dtx v1.0w
General: (RmS) Documentation improvements. 1
1998-08-17 ltfntcmd.dtx v3.3x
General: (RmS) Minor documentation fixes. 686
1998-08-17 ltfssbas.dtx v3.0v
General: (RmS) Documentation fixes. 523
1998-08-17 ltfssdcl.dtx v3.0i
General: (RmS) Corrected minor glitches in changes entries. 602
1998-08-17 ltfssini.dtx v3.0i
General: (RmS) Minor documentation fixes. 634
1998-08-17 ltlogos.dtx v1.1i
General: (RmS) Minor documentation fixes. 441
1998-08-17 ltmath.dtx v1.1c
General: (RmS) Minor documentation fixes. 785
1998-08-17 ltmiscen.dtx v1.1g
General: (RmS) Minor documentation fixes. 760
1998-08-17 ltspace.dtx v1.2w
General: Documentation fixes. 421
1998-08-17 preload.dtx v2.1g
General: (RmS) Minor documentation fixes. 683
1998-09-19 ltoutenc.dtx v1.9r
\\a: Added \\string (pr/2878) 478
1998-11-13 lttab.dtx v1.1m
\\@array: Check for hmode to see if something went wrong during parsing (pr/2884) 850
1999-01-05 fontdef.dtx v2.2x
General: Need special protection for character > in \\changes entry. . . 662
1999-01-06 ltfssbas.dtx v3.0w
\\DeclareFontEncoding: Added \\LastDeclaredEncoding to support cyrillic integration (pr/2988) 527
\\LastDeclaredEncoding: Added \\LastDeclaredEncoding to support cyrillic integration (pr/2988) 527
1999-01-06 ltoutenc.dtx v1.9r
\\@strip@args: New impl for latex/2930 475
General: Minor documentation fix. . 495
1999-01-06 ltoutput.dtx v1.2e
\\@makecol: Added negative vskip, as when processing outputbox below:

suggested by Fred Bartlett pr/2892 1111
1999-01-07 ltdefns.dtx v1.3a
\@ifnextchar: made long 107
\@newenvb: made long and brace optional arg. latex/2896 87
\@testopt: made long and brace optional arg. latex/2896 84
1999-01-07 ltdefns.dtx v1.3b
\@ifnextchar: extra \long. latex/2902 107
1999-01-07 ltoutenc.dtx v1.9r
General: Hackery to allow using fontenc several times 511
Hackery to temp support cyrillic uc/lc 509
1999-01-13 ltoutenc.dtx v1.9s
\@strip@args: Simplified solution for latex/2930 475
1999-01-18 ltdefns.dtx v1.3c
\@yargd@f: New implementation DPC /2942 85
1999-02-09 ltdefns.dtx v1.3d
\@yargd@f: catch bad argument forms by re-inserting #3 85
1999-02-12 lttextcomp.dtx v3.0j
\legacyoldstylenums: Use \rmdefault instead of cmm (pr/2954) 695
1999-02-24 ltoutenc.dtx v1.9t
General: Corrected hackery cyrillic uc/lc list 509
1999-03-01 ltdefns.dtx v1.3e
\@ifnextchar: remove extra \long. internal/2967 107
1999-04-15 ltpictur.dtx v1.1h
\@getlarrow: Replaced octal number, CAR 874
\@upvector: Replaced octal number, CAR 875
General: Replaced octal number, CAR 874, 875
Replaced octal numbers, CAR . . 862
1999-04-19 ltfloat.dtx v1.1u
\caption: Made caption an error outside a float: latex/2815 914
1999-04-27 ltboxes.dtx v1.1f
\@parboxto: (CAR) Changed \@empty to \relax as flag for natural width: pr/2975 828
1999-04-29 ltdefns.dtx v1.3f
\@yargd@f: Full expansion and conversion needed for digit in new version, see pr/3013 85
New macro added 85
1999-06-10 ltoutenc.dtx v1.9u
General: Ensure that we also forget old options (pr/2888) 511
1999-06-12 ltoutenc.dtx v1.9v
General: Extend \@uclclist only once 510
1999-10-09 ltmath.dtx v1.1e
\active@math@prime: Macro added, see PR 3104. 792
\prime@s: Introduce \active@math@prime. 792
1999-10-09 ltoutput.dtx 1.2f
\@activechar@info: Reset definition of active prime character (used in math mode) 1114
1999-10-28 ltoutenc.dtx v1.9w
\add@accent: Give \accent@spacefactor a default definition (pr/3084) 473
1999-12-08 ltoutenc.dtx v1.9x
General: Changed \CYRRHOOK and \cyrrhook to\CYRRHK and \cyrrhk as name changed in the cyrillic bundle for naming consistency with other “hook” glyphs. 509
2000-01-07 ltmiscen.dtx v1.1h
\@verbatim: Disable hyphenation even if the font allows it. 778
2000-01-15 ltpictur.dtx v1.1i
\@upvector: Removed space at end-of-line, CAR 875
2000-01-30 ltfntcmd.dtx v3.3y
\DeclareTextFontCommand: Use \hmode@bgroup now (pr/3160) . . 688
2000-01-30 ltoutenc.dtx v1.9y
General: Use \hmode@bgroup where applicable (pr/3160) . . . 482–485, 490–493, 495
\add@accent: Use \hmode@bgroup where applicable (pr/3160) 472
\hmode@bgroup: Macro added 473
2000-01-30 ltoutenc.dtx v1.9z
\@use@text@encoding: Macro reimplemented (pr/3160) . . 475, 476
\add@accent: Macro reimplemented (pr/3160) 472
\hmode@start@before@group: Macro added (pr/3160) 476
2000-05-19 ltmiscen.dtx v1.1i
\enddocument: Reset \AtEndDocument for latex/3060 761

2000-05-26 ltpage.dtx v1.0j
\\@markright: Reimplementation to fix expansion error (pr/3203). 972
\\leftmark: Use \\@empty instead of brace group (pr/3203). 972
\\markright: Reimplementation to fix expansion error (pr/3203). 970
\\rightmark: Use \\@empty instead of brace group (pr/3203). 972
2000-06-02 ltpage.dtx v1.0k
\\@markright: Small adjustment to give slightly less expansion, CAR 972
\\markright: Small adjustment to give slightly less expansion, CAR . . . 970
Tidied 1.0j reimplementation, CAR 970
2000-07-11 ltmiscen.dtx v1.1j
\\@enddocument@kernel@warnings: Fix typo in warning 763
2000-07-12 ltoutput.dtx 1.2g
General: Ensure that rule is in \\normalcolor 1155
2000-07-12 ltoutput.dtx 1.2i
\\@makecol: Removed negative vskip, as it gives unacceptable results when the depth is large: pr/3189 1111
2000-07-19 ltoutput.dtx v1.2h
\\@writesetup: Reset and restore \\@if@newlist for internal/3231 1115
2000-08-23 ltfinal.dtx v1.1c
General: Fix typo in warning 1181
2000-08-30 ltoutenc.dtx v1.91
\\@use@text@encoding: Rearranged but no change to final code, CAR (pr/3160) 475
\\add@accent: Rearranged but no change to final code, CAR (pr/3160) 472
2000-09-01 ltfinal.dtx v1.1d
\\errhelp: Set error help empty at very end (pr/449 done correctly). . . 1191
2000-09-24 ltfloat.dtx v1.2b
\\end@dblfloat: FMi: use output routine to defer float 919
2000-09-24 ltoutput.dtx v1.2b
\\@doclearpage: FMi: ensure \\doclearpage is called again until all floats are output. 1109
2000-09-24 ltoutput.dtx v1.2n
\\@addtocurcol: FMi: test for wide float was in wrong place 1128
2001-01-07 ltoutput.dtx v1.2j
\\@writesetup: And do it in the right macro (pr/3286) 1115
2001-02-16 ltxref.dtx v1.1k
\\@newl@bel: Added an extra grouplevel (PR3250), jlb 741
2001-05-25 ltclass.dtx v1.1d
\\@providesfile: Explicitly set catcode of \\endlinechar to 10 (pr/3334) 989
2001-05-25 ltdirchk.dtx v1.0x
General: Explicitly set catcode of \\endlinechar to 10 (pr/3334) . . . 4
2001-05-28 ltoutenc.dtx v1.93
General: Added composites for compatibility with T1, pr/3295 . 484
Changed the effect of \\.\\i, pr/3295 487
2001-06-02 fontdef.dtx v2.2y
General: Provide default cfg files (pr/3264) 681
2001-06-04 fontdef.dtx v2.2z
General: Guard against math active equal and pipe sign in \\models (pr/3333) 677
Guard against math active equal sign in \\Relbar (pr/3333) 677
2001-06-04 ltclass.dtx v1.1e
\\@providesfile: But only if it is a char (pr/3334) 989
2001-06-04 ltdirchk.dtx v1.0y
General: But only if it is a char (pr/3334) 4
2001-06-04 ltpictur.dtx v1.1j
\\@sline: Don't warn for exactly zero pr/3318 872
2001-06-04 ltvers.dtx v1.0i
General: Check for old format disabled 37
2001-06-05 ltoutenc.dtx v1.94
General: Text composite Commands need kludges for ',' – see tlb1903.lvt 484
2001-08-26 ltclass.dtx v1.1f
\\@providesfile: Readded setting of space char (pr/3353) 989
2002-02-24 ltplain.dtx v1.1x
\\loggingall: Macro added 33
\\loggingoutput: Macro added 33
\\showoutput: Use newly added \\loggingoutput 33
\\tracingall: Use newly added \\loggingoutput 33
2002-06-16 ltoutenc.dtx v1.95
General: Added \\textbardbl (pr/3400) 490
Added default for \\textbardbl (pr/3400) 479

2002-06-17 ltoutenc.dtx v1.95
General: Corrected \c for T1 (pr/3442) 485
Definition of \textexclamdown changed (pr/3368) 483
Definition of \textquestiondown changed (pr/3368) 483
2002-06-18 ltoutenc.dtx v1.95
General: Changed def for \textregistered to avoid small caps (pr/3420) 480
2002-10-01 ltfloat.dtx v1.1v
\thempfootnote: Use braces around \itshape to keep font change local (pr/3460). 928
2002-10-02 ltfssbas.dtx v3.0x
\DeclareFontSubstitution: Adding \LastDeclaredEncoding introduced a bug as on some occasions that macro name was stored in the internal lists instead of the actual encoding. (pr/3459) 527
2002-10-28 ltlists.dtx v1.0s
\endtrivlist: Check for math mode (pr/3437) 810
2002-10-28 ltoutenc.dtx v1.96
General: coding change, to follow bug fix by DEK in plain.tex (pr/3469) 483, 492
2002-12-13 ltbibl.dtx v1.1n
\@citex: Added \leavevmode in case citation is at start of paragraph (pr/3486) 939
2003-01-01 ltfntcmd.dtx v3.3z
General: Code checked and documentation extended by Chris 688
2003-05-18 ltbibl.dtx v1.1o
\nocite: Check if we are after \document 940
2003-08-27 ltpictur.dtx v1.1k
\@bezier: added missing displacement pr/3566 891
\@sline: check for \@linechar being empty pr/3570 872
2003-10-13 ltfinal.dtx v1.1e
General: Added extra \lccode for \- and \textcompwordmark 1180
2003-12-16 ltoutput.dtx v1.2k
\@makecol: Ensure that \@elt has a defined state (pr/3586) 1111
2003-12-30 ltpictur.dtx v1.1j
\@getcirc: issue warning if circle size can't be met pr/3473 883
2004-01-03 ltoutenc.dtx v1.99b
General: Added \textogonekcentered (pr/3532) 485
Added composites for \k (pr/3532) 490
Use \ooalign for \k (pr/3532) . . 485
2004-01-04 ltbibl.dtx v1.1p
\nocite: Changed error message . . 940
2004-01-04 ltoutenc.dtx v1.99c
General: More adjustments for ogonek (pr/3532) 485
2004-01-23 ltdefns.dtx v1.1g
\@newenva: Use kernel version of \@ifnextchar (pr/3501) 87
\@testopt: Use kernel version of \@ifnextchar (pr/3501) 84
\@xargdef: Use kernel version of \@ifnextchar (pr/3501) 84
\@xdblarg: Use kernel version of \@ifnextchar (pr/3501) 108
2004-01-23 ltdefns.dtx v1.3g
\kernel@ifnextchar: Added macro (pr/3501) 107
2004-01-28 ltclass.dtx v1.1g
\@providesfile: Use kernel version of \@ifnextchar (pr/3501) 989
2004-01-28 ltvers.dtx v1.0k
General: Check for old format made 5 years (pr/3601) 37
2004-02-02 fontdef.dtx v2.3
General: Many things from here on made robust 676
2004-02-02 ltoutenc.dtx v1.99
General: Added \textbigcircle . . 490
2004-02-04 fontdef.dtx v2.3a
General: Added bigtriangle synonyms for stmaryrd 674
2004-02-04 ltspace.dtx v1.3
\nobreakdashes: Macro added 437
2004-02-06 ltoutenc.dtx v1.99d
\@inmathwarn: New command added to fix severe bug: pr/3563 470
2004-02-07 ltoutput.dtx v1.2l
\@doclearpage: Empty kludgeins box if necessary, pr/3528 1109
2004-02-13 ltoutenc.dtx v1.99e
General: Documentation fixes: typos 466
2004-02-15 ltbibl.dtx v1.1q
\@cite@ofmt: Added hook with default value \hbox 941
\@citex: Changed to use a hook with default value \hbox 939
2004-02-15 ltspace.dtx v1.3a
\nobreakdashes: Added spacefactor setting 437

2004-10-20 ltoutput.dtx v1.2m
\\@makecol: Removed dead code . . 1111
2005-07-27 ltfssdcl.dtx v3.0j
\\DeclareMathAlphabet: (MH) Make document commands robust . . . 618
\\DeclareSymbolFontAlphabet: (MH) Make document commands robust 631
\\new@mathalphabet: (MH) Make document commands robust . . . 619
\\non@alpherr: (MH) Change because command is now properly robust 606
\\SetMathAlphabet: (MH) Make document commands robust . . . 620
2005-09-27 ltoutenc.dtx v1.99g
General: Replace \\sh@ft by \\ltx@sh@ft 482, 485, 492
2005-09-27 ltplain.dtx v1.1y
\\ltx@sh@ft: New macro 32
\\sh@ft: Macro no longer used but left for compatibility 32
2005-11-08 ltoutenc.dtx v1.99h
General: Added \\ij and \\IJ from babel. (pr/3771) 479, 484, 486
2005-11-10 ltmath.dtx v1.1g
\\[: (MH) Fixed potential problem in \\[(pr/3399). 793
General: (MH) Minor documentation fixes. 785
2006-05-18 ltboxes.dtx v1.1g
\\@parboxto: Ensure \\@parboxto holds the value of \\@tempdimb not the register itself (pr/3867) 828
2006-09-13 ltoutput.dtx v1.1m
General: Ensure that rule is in \\normalcolor 1156
2007-08-05 ltclass.dtx v1.1h
\\@fileswithoptions: Prevent loss of brackets PR/3965 999
2007-08-06 ltcntrl.dtx v1.0h
\\@fornoop: Really make defs long . . 387
2007-08-31 ltfssdcl.dtx v3.0l
\\SetSymbolFont@: Font warning changed to info for encoding change (pr/3975) 617
2009-09-24 ltvers.dtx v1.0l
General: Stop checking for old format 37
2009-10-20 ltfssdcl.dtx v3.0m
\\in@: More robust thanks to Heiko. 602
2009-10-28 lttextcomp.dtx v1.99k
General: Added Latin Modern and TeX Gyre subsets 726
2009-11-04 lttextcomp.dtx v1.99l
General: Added more Latin Modern and TeX Gyre subsets 726
2009-12-14 ltfntcmd.dtx v3.4a
\\ifmaybe@ic: Macro added 691
\\maybe@ic@: Use switch \\ifmaybe@ic instead of \\if@tempswa 691
\\t@st@ic: Use switch \\ifmaybe@ic instead of \\if@tempswa 692
2010-08-17 ltmiscen.dtx v1.1k
\\enddocument: Use braces around \\input arg (pr/4124) 762
2010-08-17 ltmiscen.dtx v1.1l
\\enddocument: Change of plan: use \\@@input instead (pr/4124) 762
2011-05-08 ltfssdcl.dtx v3.0n
\\in@: Simplified thanks to Bruno. . . 602
2011-08-19 ltclass.dtx v1.1i
\\@ifclasswith: Re-jig definition after more stringent \\in@ test. 985
2011-09-03 ltfssdcl.dtx v3.0o
\\new@mathversion: (Will) Remove \\global before \\newcount (unnecessary and caused etex bug). 614
2012-01-20 ltplain.dtx v2.0b
\\loggingall: etex tracing if available 33
2013-07-07 ltclass.dtx v1.1i
General: Correctly describe how the date in \\@ifpackagelater is used 977
2014-04-18 ltoutput.dtx v1.1o
General: Handle infinite glue from \\enlargethispage (pr/4023) . . 1156
2014-04-24 ltoutput.dtx v1.2n
\\fl@tracemessage: Renamed internal trace commands; provide as package 1146
2014-04-27 ltfloat.dtx v1.2b
\\end@dblfloat: Inline the code to allow some coexistence with packages that hook into \\end@float and do not know about the algorithm change 919
2014-06-10 ltfloat.dtx v1.2b
\\end@dblfloat: missing \\fi added . 919
2014-12-30 ltfinal.dtx v2.0a
\\newmarks: macro added 1174
\\newXeTeXintercharclass: macro added 1175
2014-12-30 ltfloat.dtx v1.2a
\\@textsubscript: Command added (latexrelease) 930
\\textsubscript: Command added (latexrelease) 929
2014-12-30 ltfssbas.dtx v3.0y
\\mathgroup: move allocation to ltplain. 523

2014-12-30 ltoutput.dtx v1.2m
General: Command updated (latexrelease) 1155
2014-12-30 ltplain.dtx v2.0a
\e@alloc: macro added 19
\e@alloc@chardef: macro added . . . 18
\e@alloc@top: macro added 18
\e@ch@ck: macro added 19
\extrafloats: macro added 20
\newlanguage: New engine-specific allocation scheme (latexrelease) . . 17
2014-12-30 ltspace.dtx v1.3b
\@: \@ discards spaces when moving (pr3039)(latexrelease) 438
2015-01-03 ltdefns.dtx v1.4a
\typein: use modified definition in luatex 82
2015-01-03 ltdirchk.dtx v1.1
General: Enable extra primitives when LuaTEX is used 3
2015-01-03 ltfinal.dtx v2.0a
General: Skip resetting codes with Unicode engines 1187
Unicode data loading added . . . 1178
2015-01-07 ltvers.dtx v1.0n
\@check@IncludeInRelease: macro added 39
2015-01-08 ltboxes.dtx v1.1h
\framebox: Make Robust (latexrelease) 826
\makebox: Make Robust (latexrelease) 819
\parbox: Make Robust (latexrelease) 828
\raisebox: Make Robust (latexrelease) 834
\rule: Make Robust (latexrelease) . 833
\savebox: Make Robust (latexrelease) 823
2015-01-08 ltdefns.dtx v1.4a
\MakeRobust: Added macro 92
2015-01-08 ltlength.dtx v1.1c
\setlength: to ensure first length argument is terminated. (latexrelease) 521
2015-01-08 ltmath.dtx v1.1h
\): Make Robust (latexrelease) 793
\]: Make Robust (latexrelease) 793
2015-01-09 ltfssini.dtx v3.1a
\em: Allow \emph to produce small caps (latexrelease) 654
\eminnershape: macro added (latexrelease) 654
2015-01-09 ltspace.dtx v1.1h
\addpenalty: Donald Arseneau's fix from PR/377703 (latexrelease) . . 433
2015-01-10 ltcounts.dtx v1.1h
\@fnsymbol: Unse \TextOrMath (latexrelease) 519
\@stpelt: Reset all within counters in one go (latexrelease) 514
2015-01-11 ltcounts.dtx v1.1h
\TextOrMath: Add command to solve robustness issues (pr/3752) (latexrelease) 520
2015-01-11 ltfloat.dtx v1.2b
\@dblfloatplacement: float order in 2-column (latexrelease) 921
\@xfloat: Check for valid option (latexrelease) 915
\end@dblfloat: float order in 2-column (latexrelease) 919
2015-01-11 ltfssbas.dtx v3.0y
\@DeclareMathSizes: Allow arbitrary units (latexrelease) 530
2015-01-11 ltspace.dtx v1.3d
\@Esphack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease) 430
\@esphack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease) 429
2015-01-14 ltoutput.dtx v1.2n
\@addtocurcol: float order in 2-column (latexrelease) 1127
\@addtodblcol: float order in 2-column (latexrelease) 1137
\@addtonextcol: float order in 2-column (latexrelease) 1134
\@doclearpage: Empty kludgeins box if necessary, pr/3528 1108
float order in 2-column (latexrelease) 1108
\@startdblcolumn: float order in 2-column (latexrelease) 1121
\@xtryfc: float order in 2-column (latexrelease) 1123
\@ztryfc: float order in 2-column (latexrelease) 1124
2015-01-14 ltspace.dtx v1.3e
\addpenalty: Avoid adding redundant skips (DPC) 433
2015-01-17 ltvers.dtx v1.0m
\@check@IncludeInRelease: modified with \@currname 39
2015-01-19 ltvers.dtx v1.0o
\@check@IncludeInRelease: Optional argument 39

2015-01-20 ltoutput.dtx v1.2m
\fl@tracemessage: Reset \IncludeInRelease flags 1147
2015-01-22 ltvers.dtx v1.0p
General: Preserve any \everyjob material inserted by a loader (.ini file) 38
2015-01-23 ltfinal.dtx v2.0b
\newmarks: use reserved count 256 1174
\newXeTeXintercharclass: use reserved count 257 1175
2015-01-23 ltplain.dtx v2.0c
\extrafloats: reserve counts 256–265 20
2015-01-24 ltfinal.dtx v2.0c
General: Skip T1-code entirely with Unicode engines 1178
2015-02-03 ltfinal.dtx v2.0d
General: Set \lccode for - with Unicode engines 1179
2015-02-16 ltoutenc.dtx v1.99m
General: Added \textcommabelow latex/4414 481
2015-02-16 ltoutenc.dtx v1.99n
General: Added \textcommaabove . . 482
Added composites for ç 490
Added composites for \c 484
2015-02-16 lttextcomp.dtx v1.99m
General: Added lmtt (Heiko Oberdiek) latex/4415 726
2015-02-19 ltvers.dtx v1.0q
\@check@IncludeInRelease: Swap argument order 39
2015-02-20 ltplain.dtx v2.0d
\loggingall: Spell commands correctly :-) 33
2015-02-21 ltdefns.dtx v1.4b
General: Removed autoload support . 80
2015-02-21 lterror.dtx v1.2o
General: Removed autoload support 388
2015-02-21 ltfiles.dtx v1.1m
General: Removed autoload support 442
2015-02-21 ltfssbas.dtx v3.0z
General: Removed autoload code . . 523
2015-02-21 ltfsscmp.dtx v3.0d
General: Removed autoload code . . 597
2015-02-21 ltfssdcl.dtx v3.0p
General: Removed autoload code . . 602
2015-02-21 ltfsstrc.dtx v3.0k
General: Removed autoload code . . 572
2015-02-21 ltoutenc.dtx v1.99m
General: Removed autoload code . . 466
2015-02-21 ltoutput.dtx v1.2n
General: Removed autoload code . 1087
\f@depth: macro added(latexrelease) 1107
2015-02-21 ltpictur.dtx v1.1k
General: Removed autoload code . . 862
2015-02-21 ltplain.dtx v2.0e
General: Removed autoload code . . . 14
2015-02-21 lttab.dtx v1.1n
General: Removed autoload code . . 837
2015-02-21 ltvers.dtx v1.0r
General: Removed autoload code . . . 37
2015-02-21 ltvers.dtx v1.0w
\@check@IncludeInRelease: set \@currname empty here (in case \IncludeInRelease input early) . 39
2015-02-22 ltfsscmp.dtx v3.0e
General: Moved all code into latexrelease - obsolete commands are no longer automatically part of the kernel 597
2015-03-02 ltplain.dtx v2.0f
\e@mathgroup@top: macro added . . . 19
\newlanguage: allow 255 math groups in Unicode engines 17
2015-03-10 ltplain.dtx v2.0g
\hideoutput: macro added 36
\loggingall: Reorganize to be less noisy 33
\tracingnone: macro added 34
2015-03-12 ltoutput.dtx v1.2m
General: initialise \@dbldeferlist again 1098
2015-03-18 ltfssdcl.dtx v3.0q
\DeclareSymbolFont: Restrict Symbol fonts to 0-15 615
\document@select@group: Introduce \e@mathgroup@top 607
\select@group: Introduce \e@mathgroup@top 605
2015-03-26 ltfinal.dtx v2.0d
General: Use renamed unicode-letters.def 1178
2015-04-07 ltfssbas.dtx v3.1a
\wrong@fontshape: Try loading fd file if family has changed 542
2015-04-28 ltfinal.dtx v2.0f
\newXeTeXintercharclass: define \xe@alloc@intercharclass for compatibility with older xelatex initialization 1175
2015-05-10 ltlists.dtx v1.0t
\@doendpe: Explicitly reset \clubpenalty before clearing \everypar; see also pr/0462 and pr/4065 812

2015-06-19 ltfinal.dtx v2.0g
- \e@alloc@intercharclass@top: Use −1 for first range to get contiguous allocation . . . 1175
- \newmarks: Use −1 for first range to get contiguous allocation . . . 1174

2015-06-19 ltplain.dtx v2.0h
- General: delete spurious old definition of \newtoks . . . 23
- \e@alloc: extra braces in case arguments not single token . . . 19
- \newlanguage: Use −1 for first range to get contiguous allocation . . . 17

2015-06-23 ltfinal.dtx v2.0h
- General: set \patch@level in ltvers rather than in ltfinal/ltpatch . . . 1190

2015-06-23 ltvers.dtx v1.0t
- General: set \patch@level in ltvers rather than in ltfinal/ltpatch . . . 37

2015-08-06 ltplain.dtx v2.0i
- \extrafloats: Add \string in case argument is not an unexpandable primitive . . . 20

2015-08-23 ltdirchk.dtx v1.2
- General: Do not use luatex prefix . . . 3

2015-08-23 ltvers.dtx v1.0v
- General: Allow negative patchlevel for pre-release . . . 38

2015-08-30 ltplain.dtx v2.1a
- \newinsert: new \newinsert implementation . . . 22

2015-09-205 ltoutput.dtx v1.3a
- General: extended \@freelist . . . 1097

2015-09-24 ltluatex.dtx v1.0a
- call_callback: Function added . . . 67
- callback.register: Function modified . . . 64
- callback_descriptions: Function added . . . 71
- \catcodetable@atletter: Macro added . . . 49
- \catcodetable@initex: Macro added . . . 49
- \catcodetable@latex: Macro added . . . 49
- \catcodetable@string: Macro added . . . 49
- add_to_callback: Function added . . . 68
- remove_from_callback: Function added . . . 70
- new_attribute: Function added . . . 57
- disable_callback: Function added . . . 71
- in_callback: Function added . . . 71
- \newattribute: Macro added . . . 49
- \newcatcodetable: Macro added . . . 49
- \newluabytecode: Macro added . . . 52
- \newluachunkname: Macro added . . . 52
- \newluafunction: Macro added . . . 51
- \newwhatsit: Macro added . . . 51
- module_error: Function added . . . 56
- module_info: Function added . . . 56
- module_warning: Function added . . . 56
- modules: Function modified . . . 54
- create_callback: Function added . . . 67
- provides_module: Function added . . . 54
- luatexbase: Table added . . . 54

2015-10-02 ltdirchk.dtx v1.2a
- General: Allow backing out of unprefixed names . . . 3

2015-10-02 ltluatex.dtx v1.0b
- General: Fix backing out of TEX code . . . 53

2015-10-02 ltluatex.dtx v1.0c
- General: Allow backing out of Lua code . . . 53

2015-10-02 ltluatex.dtx v1.0e
- uninstall: Function added . . . 72

2015-10-03 ltluatex.dtx v1.0f
- provides_module: use luatexbase_ log . . . 54

2015-10-27 ltplain.dtx v2.1b
- \extrafloats: Use global assignment when switching to extended range . . . 20

2015-11-07 ltspace.dtx v1.3f
- \@esphack: Only space if there is no space at the end of the hlist latex/4443 . . . 429

2015-11-14 ltluatex.dtx v1.0g
- General: Track LuaTEX changes for (new)token.create . . . 56

2015-11-18 ltplain.dtx v2.2a
- \newlanguage: Extended stream allocation in luatex (0.85) . . . 17

2015-11-19 ltplain.dtx v2.2b
- \newlanguage: Only extend allocation of write streams (see luatex list) . . . 17

2015-11-27 ltluatex.dtx v1.0h
- callback_descriptions: Match test in in-callback latex/4445 . . . 71
- in_callback: Guard against undefined list latex/4445 . . . 71

2015-11-29 ltluatex.dtx v1.0i
- General: Declare this as local before used in the module error definitions (PHG) . . . 54
- call_callback: Check name is not nil in error message (PHG) . . . 67
- create_callback: Check name is not nil in error message (PHG) . . . 67

2015-12-02 ltluatex.dtx v1.0j
- General: Adjust hashtokens to store the result of tex.hashtokens()), not the function (PHG) . . . 56

Assorted typos fixed (PHG) 46
Declaration/use of first_ head fixed (PHG) 55
Remove nonlocal iteration variables (PHG) 46
Remove unreachable code after calls to error() (PHG) 46
2015-12-02 ltluatex.dtx v1.0k
General: resolve name and i.description (PHG) 65
`call_callback`: Give more specific error messages (PHG) 67
`add_to_callback`: Give more specific error messages (PHG) 68
`remove_from_callback`: adjust initialization of cb local (PHG) . . 70
Give more specific error messages (PHG) 70
`create_callback`: Give more specific error messages (PHG) 67
2015-12-10 ltfinal.dtx v2.0i
General: Use new common Unicode data loaders 1178
2015-12-18 ltluatex.dtx v1.0l
General: Load Unicode data from source 49
2016-01-04 ltfinal.dtx v2.0j
General: Do not set up inter character classes for XeTeX 1178
`\e@alloc@intercharclass@top`: Start allocation at one not three 1175
2016-01-05 ltfinal.dtx v2.0k
`\e@alloc@intercharclass@top`: Remove duplicated code 1175
2016-01-05 ltfinal.dtx v2.0l
General: Correct latexrelease guards 1178
Ensure old definitions for inter-character class toks are available using latexrelease 1178
Missing brace 1178
2016-01-05 ltfinal.dtx v2.0m
General: Undefine XeTeX classes when using patching an older kernel . 1178
2016-01-05 ltfinal.dtx v2.0p
General: Only apply XeTeX change if XeTeX is in use 1178
2016-02-11 ltluatex.dtx v1.0m
General: pdf_stream_filter_callback removed 64
process_ rule, [hv]pack_ quality append_ to_ vlist_ filter added . 63
read_ cidmap_ file added 62
show_ warning_ message added . . 63
token_ filter removed 63
2016-02-18 ltfssdcl.dtx v3.0r
`\@DeclareMathDelimiter`: Check for delimiter not `\delimiter` 626
`\DeclareMathAccent`: Check for mathaccent not `\mathaccemt` . . . 622
`\DeclareMathRadical`: Check for radical not `\radical` 629
`\DeclareMathSymbol`: Check for mathchar not `\mathchar` 624
2016-03-13 ltluatex.dtx v1.0n
General: contribute_ filter added . . . 63
insert_ local_ par added 63
2016-03-29 ltpictur.dtx v1.1l
`\@oval`: add setting of line tests 884, 885
initialise tests 884
`\@ovhorz`: use glue not leaders if horizontal line not required 887
`\@ovvert`: use glue not leaders if vertical line not required 886
`\if@ovhline`: macro added (latex/4452) 884
`\if@ovvline`: macro added (latex/4452) 884
2016-04-22 ltfinal.dtx v2.0q
`\e@alloc@intercharclass@top`: XeTeX 0.99996 has 4096 char classes not 256 1175
2016-06-19 ltoutenc.dtx v1.99m
General: OT1 definition (was duplicate T1 definition) 484
2016-06-20 ltclass.dtx v1.1j
General: don't declare as `\@onlypreamble` 984
2016-07-29 ltplain.dtx v2.2c
`\extrafloats`: use `\global \chardef` 20
`\newinsert`: fix for tlb-newinsert-001 22
2016-10-02 ltclass.dtx v1.2a
`\@ifclasswith`: Ignore spaces while checking for option clash 985
`\ExecuteOptions`: Ignore spaces in argument 995
2016-10-15 ltdirchk.dtx v1.2b
General: Require eTEX 3
2016-10-15 lterror.dtx v1.2p
General: Require eTEX 388
2016-10-15 ltfinal.dtx v2.0r
General: Require eTEX 1174
2016-10-15 ltfinal.dtx v2.0s
General: Tidy up status of char 127 1174
2016-10-15 ltfssini.dtx v3.1b
General: Require eTEX 634
2016-10-15 ltplain.dtx v2.2d
General: Require eTEX 14

2016-10-16 ltplain.dtx v2.3a
\newlanguage: Allow languages up to 16383 in luatex 17
2016-10-19 ltcounts.dtx v1.1j
\TextOrMath: Test directly for \protected 520
2016-11-06 ltplain.dtx v2.3b
General: Drop \outer entirely 14
2016-11-09 ltclass.dtx v2.1b
\@fileswithoptions: Improve \ifx tests PR/4497 999
2016-11-17 ltluatex.dtx v1.0p
General: call_ edit added 63
2016-12-03 fontdef.dtx v3.0a
General: (DPC) Default to TU encoding for Unicode TeX engines 664
\shapedefault: (DPC) Default to TU encoding for Unicode TeX engines 668
2016-12-04 ltoutenc.dtx v2.0a
General: Added TU encoding 498
2017-01-01 ltoutput.dtx v1.3b
General: make fpmin negative so ignored even if float height is negative 1155
2017-01-10 ltfssbas.dtx v3.2a
\showhyphens: Add version of \showhyphens that works with XeTEX. 547
2017-01-23 ltoutenc.dtx v2.0b
General: Added TU specific commands in ASCII range pr/4500 498
2017-01-24 ltoutenc.dtx v2.0c
General: Declare TU composites for i and j 498
Make \textasteriskcentered U+2217 not U+204E 498
TeX ligature syntax for xetex and luatex reversed 498
2017-01-24 ltoutenc.dtx v2.0d
General: Declare macron composites for YyGg 498
2017-02-12 ltoutenc.dtx v2.0e
General: Declare fallback code for \textasteriskcentered 498
2017-02-18 ltluatex.dtx v1.1c
new_attribute: Parameterize count used in tracking 57
new_bytecode: Parameterize count used in tracking 58
new_chunkname: Parameterize count used in tracking 58
new_whatsit: Parameterize count used in tracking 58
2017-02-19 ltoutenc.dtx v2.0f
General: add \@empty to guard against 3rd argument being empty 483
declare composites with empty base for hat and tilde, use same slots for \textasciicircum ans \textasciitilde 498
declare straight quotes using new \remove@tlig command 498
2017-02-22 ltoutenc.dtx v2.0g
General: Fix typo introduced at 2.0f 498
2017-02-24 ltoutenc.dtx v2.0h
General: introduce \DeclareUnicodeAccent 498
\DeclareTextCompositeCommand: add check whether the accent command is defined for this encoding 473
2017-03-08 ltclass.dtx v1.2c
General: add \@parse@version@dash to support yyyy-mm-dd as well as yyyy/mm/dd 984
2017-03-09 ltfinal.dtx v2.0t
\l@nohyphenation: ensure \l@nohyphenation is defined. . 1181
2017-03-09 ltmiscen.dtx v1.1m
\@verbatim: Use \language not \hyphenchar 778
\verb: Use \language to stop hyphenation 783
2017-03-10 ltfiles.dtx v1.1n
\document: Save language default . . 445
2017-03-10 ltoutput.dtx v1.3c
\@writesetup: Reset \language . . 1115
2017-03-13 ltdefns.dtx v1.5a
\-: Define \- in terms of \hyphenchar 110
2017-03-27 ltdefns.dtx v1.5b
\@dischyph: Define \@dischyph after \- 110
2017-03-28 ltluatex.dtx v1.1e
General: glyph_ stream_ provider added 63
2017-03-29 ltboxes.dtx v1.3a
\@arrayparboxrestore: Reset \lineskiplimit 830
2017-04-05 ltoutenc.dtx v2.0i
\DeclareTextCompositeCommand: Declare accent command if not already declared when declaring a composite. 473
2017-04-10 ltplain.dtx v2.3c
\newlanguage: Correction to code to skip write18 in luatex 17

2017-04-11 ltoutput.dtx v2.4a
\newpage: account for the depth of the last row of the page 1101
2017-12-17 ltoutput.dtx v1.4b
\@addtonextcol: fix doc guards . . 1134
2018-01-06 ltdefns.dtx 1.5c
\@ifundefined: Avoid defining undefined commands to \relax . 106
2018-02-18 ltclass.dtx v1.2d
\@ifl@ter: Added 0 up front to make bad data come out as 0. 984
General: Introduce rollback concept 1018
2018-03-08 ltcounts.dtx v1.1k
\@ifbothcounters: Interface added 516
\@removefromreset: Interface added 515
\counterwithin: Interface added . . 517
2018-03-24 ltclass.dtx v1.2e
\pkgcls@use@this@release: Use full file name for old release 1023
2018-03-25 ltfinal.dtx v2.1a
General: default to UTF-8 1182
\UseRawInputEncoding: Macro added 1183
2018-03-27 ltclass.dtx v1.2f
\endfilecontents: Use full file name for old release 1012
2018-04-06 ltfinal.dtx v2.1b
\UseRawInputEncoding: Undo changes to \DeclareFontEncoding@ and definition of \DeclareUnicodeCharacter . . . 1183
2018-04-07 ltfinal.dtx v2.1c
\UseRawInputEncoding: Undefine \inputencodingname 1183
2018-04-08 ltclass.dtx v1.2g
\@ifl@ter: Strip leading spaces from dates. 984
2018-04-08 ltclass.dtx v1.2h
\@onefilewithoptions: Pass expanded date 1019
2018-04-08 ltfinal.dtx v2.1d
General: Delay full UTF-8 handling to \everyjob 1184
2018-04-11 ltcounts.dtx v1.1l
\counterwithin: Correct default (issue/38) 517
2018-05-02 ltluatex.dtx v1.1g
General: find_ sfd_ file removed . . . 62
finish_ synctex_ callback added . . 63
glyph_ not_ found added 63
read_ sfd_ file removed 62
2018-05-08 ltclass.dtx v1.2i
\pkgcls@parse@date@arg: Make suspicious rollback a warning not error: github issue 43 1021
2018-05-11 ltfinal.dtx v2.18
General: Make invalid UTF-8 also safe, for legacy filesystem encodings . 1184
2018-05-29 ltclass.dtx v1.2j
\endfilecontents: use \csname not \@undefined 1013
2018-08-11 ltoutenc.dtx v2.0j
General: Provide \guillemetleft and \guillemetright 486, 492, 501
2018-08-18 ltluatex.dtx v1.1h
General: append_ to_ vlist_ filter is exclusive 63
2018-08-24 ltfinal.dtx v2.1f
\document@default@language: Add to latexrelease (github/68) 1181
2018-09-02 ltsect.dtx v1.1b
\@dottedtocline: Prevent protrusion (https://tex.stackexchange.com/q/172785/10109) 909
2018-09-24 fontdef.dtx v3.0b
General: Start LR-mode if necessary (git/49) 680
2018-09-24 ltmath.dtx v1.2b
\smash: Start LR-mode if necessary (git/49) 789
\vphantom: Start LR-mode if necessary (git/49) 788
2018-09-24 ltspace.dtx v1.3h
\enspace: Start LR-mode if necessary (git/49) 440
\leavevmode@ifvmode: Macro added (git/49) 440
2018-09-26 ltdefns.dtx v1.5e
\renew@command: Always explicitly generate a space after the csname and not rely on \noexpand to save tokens (git/41) 86
2018-09-26 ltmiscen.dtx v1.1n
\@writefile: Sometimes mask the endline char when writing to files (github/73) 767
\add@percent@to@temptokena: Sometimes mask the endline char when writing to files (github/73) 767
\protected@file@percent: Sometimes mask the endline char when writing to files (github/73) 766
2018-09-26 ltsect.dtx v1.1c
\addcontentsline: Sometimes mask the endline char when writing to

files (github/73) 907
2018-10-10 ltspace.dtx v1.3i
\@esphack: Don't introduce breakpoints if @nobreak is true and after sections 429
2018-10-11 ltmiscen.dtx v1.1o
\@@sverb: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) 781
\@setupverbvisiblespace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) 780
\@verbvisiblespacebox: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) 779
\asciispace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) 779
verbatim*: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) 779
\verbvisiblespace: Provide \verbvisiblespace such that it is usable in normal text (github/70) 779
Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) 779
2018-10-21 ltluatex.dtx v1.1i
new_luafunction: Function added . . 59
2018-11-09 ltbibl.dtx LaTeX2e
\bibliography: Zap spaces in the argument as BibTeX doesn't support them (github/88) 939
2018-11-18 ltoutenc.dtx v2.0k
General: Provide \Hwithstroke and \hwithstroke 503
2018-11-19 ltoutenc.dtx v2.0k
General: Added \Hwithstroke and \hwithstroke 485
2018-11-28 ltoutput.dtx v1.4d
\@combinedblfloats: Unbox \@outputbox to preserve boxing level (github/94) 1120
2018-12-30 lttab.dtx v1.1p
\@tabclassz: Add extra \hskip to guard against an \unskip at the start of a c-column cell (gh/102) 857
2019-02-07 ltfilehook.dtx v1.1o
\unqu@tefilef@und: Expand \@filef@und before executing second argument (github/109) . 1046
2019-02-07 ltfiles.dtx v1.1o
\@swaptwoargs: Helper macro added 460
2019-02-07 ltfssbas.dtx v3.2b
\define@newfont: Changed wording of warning (github/107) 537
2019-06-18 ltluatex.dtx v1.1j
General: finish_ synctex_ callback renamed finish_ synctex 63
font_ descriptor_ objnum_-provider added 63
make_ extensible added 63
new_ graf added 63
page_ objnum_ provider added . . 63
process_ pdf_ image_ content added 63
wrapup_ run added 63
2019-07-01 ltclass.dtx v1.3a
\endfilecontents: Support UTF8 and spaces in filecontents environment file name 1010
2019-07-01 ltfiles.dtx v1.2a
\IfFileExists: Support UTF-8 . . . 456
\includeonly: Support UTF-8 450
\set@curr@file: Support UTF-8 . . 455
2019-07-09 ltfssbas.dtx v3.2c
\DeclareErrorFont: Don't set any \f@... macros 541
2019-07-09 ltfssini.dtx v3.1c
General: Explicitly set some defaults 659
2019-08-22 ltxref.dtx v1.1l
\labelformat: Commanded moved from varioref.sty 744
\Ref: Commanded moved from varioref.sty 744
\refstepcounter: Allow \p@... to have an argument 743
2019-08-27 fontdef.dtx v3.0c
General: Various commands made robust throughout the file 673
2019-08-27 ltboxes.dtx v1.3b
General: Various commands made robust 818
2019-08-27 ltclass.dtx v1.3b
\endfilecontents: Make various commands robust 1010
2019-08-27 ltdefns.dtx v1.5f
General: Make various commands robust 111
\MakeRobust: Make the assignments global as we may need to apply them inside a group 92
2019-08-27 ltfilehook.dtx v1.2b
\unqu@tefilef@und: Make command robust 1046

2019-08-27 ltfiles.dtx v1.2b
\IfFileExists: Make command robust 456
2019-08-27 ltfssbas.dtx v3.2d
General: Make various commands robust 523
2019-08-27 ltfssdcl.dtx v3.0s
\DeclareMathAccent: Make math accents robust 622
\set@mathdelimiter: Make math delimiters robust 628
2019-08-27 ltfssini.dtx v3.1d
General: Make various commands robust 634
2019-08-27 ltidxglo.dtx v1.1f
General: Make \index and \gloassary robust 934
2019-08-27 ltlength.dtx v1.1d
General: Make various command robust 521
2019-08-27 ltlogos.dtx v1.1j
\TeX: Make \TeX command robust . 441
2019-08-27 ltmath.dtx v1.2c
General: Make various commands robust 785
2019-08-27 ltmiscen.dtx v1.1p
General: Make various commands robust 760
\begin: Make command robust . . . 769
\end: Make command robust 771
2019-08-27 ltoutput.dtx v1.4e
\@begindvibox: Make \AtBeginDvi robust 1099
2019-08-27 ltpage.dtx v1.0l
General: Make various commands robust 969
2019-08-27 ltpictur.dtx v1.1m
General: Make various commands robust 862
Remove several unnecessary \gdef definitions 862
2019-08-27 ltsect.dtx v1.1d
General: Make various commands robust 898
2019-08-27 ltspace.dtx v1.3j
General: Make various commands robust 421
2019-08-27 lttab.dtx v1.1q
General: Make various commands robust 837
Remove several unnecessary \gdef definitions 837
2019-08-30 lterror.dtx v1.2q
\conditionally@traceoff: Macro added 398
\conditionally@traceon: Macro added 398
2019-09-09 ltfssdcl.dtx v3.0s
\DeclareMathSymbol: Allow definition if the math symbol was a command already robust 624
2019-09-11 ltclass.dtx v1.3c
\endfilecontents: Support optional argument for filecontents 1010
2019-09-14 ltfinal.dtx v2.1h
\@uclclist: Expand UTF8 chars when case changing (github/177) 1188
2019-09-16 ltxref.dtx v1.1m
General: Correctly revert the \p@... change 744
2019-09-21 fontdef.dtx v3.0d
General: Distangle alias (gh/184) 675, 679
2019-10-02 ltexpl.dtx v0.0
General: Initial version 73
2019-10-02 ltfinal.dtx v2.2
General: Load ltexpl 1191
2019-10-02 ltluatex.dtx v1.1k
General: linebreak_ filter is exclusive 63
mlist_ to_ hlist is exclusive 63
process_ rule is exclusive 63
2019-10-07 lttab.dtx v1.1q
\extracolsep: This needs to expand 849
2019-10-11 ltfiles.dtx v1.2c
\set@curr@file: Remove one brace group 455
2019-10-11 ltfsstrc.dtx v3.0l
\@font@aliasinfo: Added 'alias' size function 595
2019-10-18 ltclass.dtx v1.3d
\load@onefilewithoptions: Initialize \...-h@@k only when loading the package or class (gh/198) 1001
2019-10-22 ltluatex.dtx v1.1j
General: page_ objnum_ provider and process_ pdf_ image_ content classified data 63
2019-10-25 ltmiscen.dtx v1.1q
\add@percent@to@temptokena: Allow unbalanced conditionals in #1 (gh/202) 767
2019-10-26 ltfiles.dtx v1.2d
\@iffileonpath: quote on openin . . 459
\IfFileExists: don't quote name . 456
\IfFileExists@: quote on openin . . 457
\set@curr@file: remove quotes . . . 455

2019-11-01 ltdirchk.dtx v1.3a
\filename@parse: take last . not first 12
2019-11-02 ltmiscen.dtx v1.1s
\@centercr: Make \@centercr robust (gh/203) 774
2019-11-02 ltspace.dtx v1.3k
\@normalcr: Make also \@normalcr robust 425
2019-11-09 ltfiles.dtx v1.2e
\set@curr@file: expand and \string before removing quotes 455
2019-11-10 ltmiscen.dtx v1.1r
\add@percent@to@temptokena: fix to special comment catcodes (gh/202) 767
2019-11-11 ltfiles.dtx v1.2f
\@iffileonpath: make \@filef@und match quoting used on \openin . 459
2019-11-22 ltoutenc.dtx v2.0l
General: Avoid spurious if fontenc selects LY1 as default encoding (gh/199) 511
2019-11-29 ltclass.dtx v1.3e
\@pr@videpackage: Protect package info text (gh/52) 988
2019-12-17 fontdef.dtx v3.0e
\mddefault: Set \bfdefault to “b” 667
\shapedefault: Set \shapedefault explicitly to “n” 668
\updefault: Set \updefault to “up” 667
2019-12-17 ltfntcmd.dtx v3.4c
\textssc: Macro added 689
2019-12-17 ltfssbas.dtx v3.2e
\usefont: Don’t call \fontseries or \fontshape 532
2019-12-17 ltfssini.dtx v3.1e
General: Provide custom series settings a la mweights 635
\DeclareEmphSequence: Provide \emph sequences 653
2019-12-18 ltoutenc.dtx v2.0m
General: Don’t fake \textcompwordmark; take default from T1 instead 480
\add@accent: Avoid code that breaks \accent 473
2019-12-21 fontdef.dtx v3.0e
General: Distangle alias (gh/184) 673–676
2020-01-05 ltclass.dtx v1.3f
\endfilecontents: Support more write streams in LuaTeX gh/238 1010
2020-01-11 ltfssini.dtx v3.1f
\rmfamily: Streamlined implementation with hook 648
\ttfamily: Streamlined implementation with hook 648
2020-01-20 ltfssdcl.dtx v3.0t
\set@mathdelimiter: fix for gh/251 628
2020-01-20 ltoutenc.dtx v2.0n
General: fix for gh/251 481
2020-01-22 lttextcomp.dtx v1.0b
\tc@subst: The overall default is \textcompsubstdefault not \substdefault 698
2020-01-25 fontdef.dtx v3.0f
General: Load t1enc.def last (gh/255) 664
2020-01-25 ltoutenc.dtx v2.0m
General: Load each encoding file only once (gh/255) 510
2020-01-27 ltclass.dtx v1.3g
\endfilecontents: Fix typo in error message 1012
2020-01-28 ltclass.dtx v1.3h
\endfilecontents: Allow spaces in option string and display only unknown options not the whole option list (gh/256) 1010
2020-01-31 ltvers.dtx v1.1e
General: Allow for upcoming format as pre-release 0 38
2020-02-02 ltluatex.dtx v1.1l
General: Add reverselist callback type 66
glyph_ info added 63
page_ order_ index added 63
post_ linebreak_ filter is reverselist 63
create_callback: Provide proper fallbacks for user-defined callbacks without user-provided default handler 67
2020-02-05 ltfssini.dtx v3.1g
\DeclareFontSeriesDefault: Clarified error text 636
Corrected misspelled csname (gh/264) 636
2020-02-05 lttextcomp.dtx v2.0n
General: Changed the package default to info (gh/262) 716
Ensure we are on a new format (gh/260) 716
2020-02-07 ltfssini.dtx v3.1h
\symbol: XeTeX-specific version to avoid bug in maths mode. 656
2020-02-10 ltfssaxes.dtx v1.0c
\fontseries: Switch \if@forced@series added 560
\fontseriesforce: Switch \if@forced@series added 560

\if@forced@series: Switch \if@forced@series added 560
2020-02-10 ltfssini.dtx v3.1h
\@defaultfamilyhook: Add \@defaultfamilyhook to \normalfont (gh/269) 649
\reset@font: Add \@defaultfamilyhook to \normalfont (gh/269) 657
2020-02-10 lttextcomp.dtx v1.0c
General: Use \@tabacckludge for tabbing where necessary (gh/271) 701
2020-02-11 fontdef.dtx v3.0g
General: Provide value for \@fontenc@load@list (gh/273) . 664
2020-02-11 ltfssini.dtx v3.1h
General: Provide default value for \@fontenc@load@list (gh/273) . 659
2020-02-11 ltoutenc.dtx v2.0o
General: Update \@fontenc@load@list with option list (gh/273) 511
2020-02-14 ltpictur.dtx v1.1n
\linethickness: Suppress spaces following the declaration (gh/274) 867
2020-02-18 ltfssini.dtx v3.1i
\bfseries: Make the \ifx selection outside of \fontseries argument so that it is not done several times 643
\mdseries: Make the \ifx selection outside of \fontseries argument so that it is not done several times 644
\prepare@family@series@update: No series auto-update when forced (gh/277) 639
Recognize current family if it is not a "meta" family and auto-update series using \bfdefault (gh/277) 639
2020-02-18 ltmath.dtx v1.2d
\mathindent: Make \mathindent a skip register to match amsmath (gh/252) 798
equation: Separate formula and eqn number by at least a space in fleqn option 799
2020-02-20 ltclass.dtx v1.3j
\endfilecontents: Fix missing quotes around file name (gh/284) 1012
2020-02-24 ltfssbas.dtx v3.2f
\DeclareFontShape@: Drop surplus "m" in series when defining fontshape (gh/289) 524
2020-02-25 ltfssini.dtx v3.1j
\bf@def@ult: Drop surplus "m" from \bfdef@ult and \mddef@ult (gh/291) 646
\prepare@family@series@update: Drop surplus "m" from \target@series@value (gh/291) 640
\update@series@target@value: Drop surplus "m" from \reserved@d (gh/291) 640, 641
2020-02-27 ltdefns.dtx v1.5g
\@gobblethree: Macro added 89
2020-02-27 ltfssaxes.dtx v1.0d
\series@maybe@drop@one@m: Drop "m" in certain values from a fixed list (gh/293) 564
\set@target@series: Drop "m" only in a specific set of values (gh/293) 563
2020-02-27 ltfssbas.dtx v3.2g
\DeclareFontShape@: Only "m" if the series value is a member of a fixed list and issue warning if doing it (gh/293) 524
2020-03-02 ltexpl.dtx v1.0a
General: Don't load expl3 if already in the format (gh/295) 73
2020-03-05 ltexpl.dtx v1.1
General: Load xparse.ltx if \NewDocumentCommand is not defined by expl3.ltx 73
2020-03-06 ltboxes.dtx v1.3c
\clap: Macro \clap added 836
2020-03-07 ltluatex.dtx v1.1m
remove_from_callback: Do not call callback.register for user-defined callbacks 70
2020-03-07 ltmath.dtx v1.2e
\negthickspace: Add amsmath math/text spacing commands to the kernel (gh/303) 791
2020-03-07 ltspace.dtx v1.3l
General: Moved \thinspace, \negthinspace and \, to ltmath.dtx (gh/303) 421
2020-03-19 fontdef.dtx v3.0h
General: Support legacy use of \bfdefault and \mddefault (gh/306) 667
2020-03-19 ltfssdcl.dtx v3.0u
\document@select@group: fix for (gnats/3357) 608
2020-03-19 ltfssini.dtx v3.1k
\DeclareFontSeriesDefault: Support legacy use of \bfdefault and

\mddefault (gh/306) 636
\maybe@update@bfseries@defaults: Support legacy use of \bfdefault and \mddefault (gh/306) 643
\mdseries: Support legacy use of \bfdefault and \mddefault (gh/306) 644
2020-04-06 ltfssini.dtx v3.1m
\bf@def@ult: Hook added (gh/306) 646
\maybe@update@bfseries@defaults: Hook added (gh/306) 643
\maybe@update@mdseries@defaults: Hook added (gh/306) 644
2020-04-07 ltclass.dtx v1.3k
\IfFormatAtLeastTF: Macro added; also in rollback (gh/168) 983
\load@onefile@withoptions: Use different method to ignore unprocessed options (gh/22) . . 1005
\ProcessOptions*: Use different method to ignore unprocessed options (gh/22) 994
\RequirePackageWithOptions: Use different method to ignore unprocessed options (gh/22) . . . 997
2020-04-09 ltfloat.dtx v1.2d
\@textsubscript: Set non-zero baseline (gh/249) 930
\textsubscript: Set non-zero baseline (gh/249) 929
2020-04-13 ltfssdcl.dtx v3.0v
\process@table: Small update for speed. 613
2020-04-13 ltfssini.dtx v3.1n
\init@series@setup: Handling \seriesdefault changes (gh/315) 642
\seriesdefault@kernel: Handling \seriesdefault changes (gh/315) 660
2020-04-21 ltmath.dtx v1.2f
\@yeqncr: Support calc syntax (gh/152) 796
2020-04-21 ltmiscen.dtx v1.1t
\@icentercr: Support calc syntax (gh/152) 775
2020-04-21 ltpictur.dtx v1.1o
\@istackcr: Support calc syntax (gh/152) 867
2020-04-21 ltspace.dtx v1.3m
\@hspace: Support calc syntax (gh/152) 439
\@newline: Support calc syntax (gh/152) 426
\@vspace@calcify: Support calc syntax (gh/152) 426
\@vspacer: Support calc syntax (gh/152) 435
\addvspace: Support calc syntax (gh/152) 432
2020-04-21 lttab.dtx v1.1r
\@itabcr: Support calc syntax (gh/152) 843
\@yargarraycr: Support calc syntax (gh/152) 851
2020-04-22 ltmiscen.dtx v1.1u
\@@sverb: Drop spaces before \verb delimiter (gh/327) 781
2020-04-22 ltoutenc.dtx v2.0p
General: y unicode value in tuenc.def 466
2020-04-29 lttextcomp.dtx v1.0d
General: Make all capital accents text commands for hyperref (gh/332) 701
2020-05-02 ltfiles.dtx v1.2g
\@include: Support spaces in filenames by enclosing the names of .aux-files in quotes (gh/217) . 452
\includeonly: Get rid of leading and trailing spaces from the filename (gh/217) 450
Improved support for spaces in filenames (gh/217) 450
Pass the filename to \@include by value instead of by reference (gh/217) 450
2020-05-05 ltxref.dtx v1.1n
\refstepcounter: record the counter name in \@currentcounter 742
2020-05-06 ltspace.dtx v1.3n
General: Made softhyphen active in TU engines 440
2020-05-09 ltdefns.dtx v1.5j
\@if@DeclareRobustCommand: Added \DeclareCommandCopy (gh/239) . 101
\DeclareCommandCopy: Added \DeclareCommandCopy (gh/239) . . 98
2020-05-11 ltdefns.dtx v1.5j
\@dischyph: Do not overwrite \- under LuaTEX 110
2020-05-15 ltdefns.dtx v1.5g
\typeout: Allow \par in the argument (gh/335) 80
2020-05-19 ltfssaxes.dtx v1.0e
\series@maybe@drop@one@m: Need to use \edef (gh/336) 564
2020-05-19 ltfssini.dtx v3.2a
\IfFontSeriesContextTF: Macros added (gh/335) 651

2020-05-31 ltmiscen.dtx v1.1u
\centering: Added \finalhyphendemerits setting (gh/247) 775
\raggedleft: Added \finalhyphendemerits setting (gh/247) 776
\raggedright: Added \finalhyphendemerits setting (gh/247) 776
2020-06-04 ltexpl.dtx v1.2c
General: Define a local version of some LaTeX 2ε basic macros to support package loading 73
2020-06-04 ltfinal.dtx v2.2a
General: Load ltexpl in ltdefns . . 1191
2020-06-05 ltclass.dtx v1.3l
\@currnamestack: Added \@expl@pop@filename@@ 981
Added \@expl@push@filename@@ and \@expl@push@filename@aux@@ . . 980
2020-06-05 ltfiles.dtx v1.2h
\document: Added hook to load l3backend code 444
2020-06-10 ltluatex.dtx v1.1n
General: Define \@gobble/\@firstofone even for LaTeX to allow early loading. 47
2020-07-04 ltoutenc.dtx v2.0q
General: Implement \remove@tlig in LuaTeXwithout font reloading . . 498
2020-07-08 ltexpl.dtx v1.2d
General: Add a last-minute hook for expl3 74
2020-07-08 ltfinal.dtx v2.2b
General: Add a last-minute hook for expl3 1180
2020-07-27 ltmath.dtx v1.2g
\cases: Don't make the command \long (gh/354) 789
\matrix: Don't make the command \long (gh/354) 789
\pmatrix: Don't make the command \long (gh/354) 789
2020-07-27 ltoutenc.dtx v2.0r
\@use@text@encoding: Don't make the command \long (gh/354) . 475, 476
2020-07-27 ltpage.dtx v1.0m
\markright: Don't make the command \long (gh/354) 970, 971
2020-07-27 ltpictur.dtx v1.1p
\linethickness: Don't make the command \long (gh/354) 867
2020-07-27 ltsect.dtx v1.1e
\author: Don't make the command \long (gh/354) 898
\date: Don't make the command \long (gh/354) 898
2020-08-01 ltluatex.dtx v1.1p
General: new_ graf is exclusive . . . 63
2020-08-02 ltluatex.dtx v1.1q
\newattribute: Move reset to 0 inside conditional 49
\newluabytecode: Move reset to 0 inside conditional 52
\newluachunkname: Move reset to 0 inside conditional 52
\newluafunction: Move reset to 0 inside conditional 51
\newwhatsit: Move reset to 0 inside conditional 51
2020-08-08 ltclass.dtx v1.3m
\endfilecontents: define \q@curr@file directly as the quotes have already been removed (gh/220) 1011
2020-08-10 ltluatex.dtx v1.1r
General: Load ltluatex Lua module during format building 52
2020-08-15 ltpictur.dtx v1.2a
\@defaultunitsset: Macro added . 864
2020-08-19 ltdefns.dtx v1.5k
\@carcube: Made \long for \NewCommandCopy 83
\robust@command@act: Made \robust@command@act (was \declare@command@copy) more generic 96
\ShowCommand: Added \ShowCommand (gh/373) 100
2020-08-19 ltexpl.dtx v1.2e
General: Add \@expl@cs@⟨*thing*⟩@spec@@N for \ShowCommand (gh/373) 77
Add \@expl@cs@to@str@@N and \@expl@str@if@eq@@nnTF for \NewCommandCopy (gh/239) 76
2020-08-20 ltplain.dtx v2.3d
\alloc@: Define \alloc@ in terms of \e@alloc 21
2020-08-21 ltclass.dtx v1.3o
General: Integration of new hook management interface 974
2020-08-21 ltdefns.dtx v1.5l
General: Integration of new hook management interface 80

2020-08-21 ltdefns.dtx v1.5m
\MakeRobust: Make \MakeRobust produce the same command structure as \DeclareRobustCommand 92
2020-08-21 ltexpl.dtx v1.2d
General: Dropped unused command . 73
2020-08-21 ltfiles.dtx v1.2i
General: Integration of new hook management interface 442
2020-08-21 ltfinal.dtx v2.2i
General: Integration of new hook management interface 1174
2020-08-21 ltfssaxes.dtx v1.0g
General: Integration of new hook management interface 571
2020-08-21 ltfssini.dtx v3.2b
\bf@def@ult: Integration of new hook management interface 646
mdseries/defaults: Integration of new hook management interface 649
\maybe@update@bfseries@defaults: Integration of new hook management interface 643
\maybe@update@mdseries@defaults: Integration of new hook management interface 644
\reset@font: Integration of new hook management interface 657
\rmfamily: Integration of new hook management interface 648
\ttfamily: Integration of new hook management interface 648
2020-08-21 ltmiscen.dtx v1.1v
General: Integration of new hook management interface 760
2020-08-21 ltoutput.dtx v1.4f
\@begindvibox: Integration of new hook management interface . . . 1099
2020-08-23 ltxref.dtx v1.1o
\refstepcounter: add default definition of \@currentcounter . 742
2020-09-06 ltclass.dtx v1.3q
\load@onefilewithoptions: Save \@currpkg@reqd so that we don't lose track of package substitutions. 1002
2020-09-06 ltdefns.dtx v1.5n
\char@if@alph: Macro added 109
2020-09-06 ltexpl.dtx v1.2f
General: Add \@expl@str@map@function@@NN and \@expl@str@map@function@@nnN for \string@makeletter (gh/386) 77
2020-09-09 ltshipout.dtx v1.0b
__shipout_picture_overlay:n: Prevent overfull box warnings (gh/387) 1079
2020-09-26 ltfinal.dtx v2.2j
General: Load first aid file if existing 1192
2020-09-30 ltfssini.dtx v3.2d
\maybe@update@bfseries@defaults: \bfdefault@previous not \bfseries@previous (gh/395) . . 643
\mdseries: \mddefault@previous not \mdseries@previous (gh/395) . . 644
2020-10-01 ltclass.dtx v1.3r
\@pr@videpackage: Allow for package substitution 988
2020-10-01 ltsect.dtx v1.1e
\addcontentsline: add a fourth argument for better hyperref compatibility 907
2020-10-04 ltfiles.dtx v1.2j
\@include: Quotes around the aux file name removed, they are not needed and upset BibTeX (gh/400) 452
2020-10-04 lthooks.dtx v1.0d
General: Definition \AddToHookNext was supposed to be for \AddToHook vice versa (gh/401) 301
2020-10-08 ltclass.dtx v1.3s
\@currnamestack: Added missing 2020/02/02 \IncludeInRelease . 980
2020-10-11 ltclass.dtx v1.3t
\load@onefilewithoptions: Restore \@currpkg@reqd after finished loading a package file (gh/408). 1003
2020-10-18 ltclass.dtx v1.3t
\PassOptionsToClass: Drop path from \input@path (gh/414). . . . 990
2020-10-23 ltmiscen.dtx v1.1w
\enddocument: Make enddocument/afteraux one-time 762
2020-10-26 ltmiscen.dtx v1.1x
\@kernel@before@enddocument: \enddocument should always start out in vmode (gh/385) 765
2020-11-09 ltclass.dtx v1.3u
\pkgcls@rollbackdate@error: Change help text because the package may have existed then — there is just no rollback data (gh/423). 1024
2020-11-09 ltmath.dtx v1.2h
\negmedspace and \negthickspace have been only in

amsmath, so we need to undefine for rollback (gh/423) 792
2020-11-20 ltclass.dtx v1.3u
\\@currpath: Macro added 979
\\@kernel@currpathstack: Macro added 982
\\load@onefile@withoptions: Copy option list to the requested package. 1004
\\PassOptionsToClass: Copy option list to the requested package. . . . 990
\\ProvidesPackage: Use string comparison instead of \\ifx 987
2020-11-20 ltcmd.dtx v1.0a
General: Initial version derived from xparse.dtx 113
2020-11-20 ltfilehook.dtx v1.0d
\\unqu@tefilef@und: Move loading to \\@input@file@exists@with@hooks and expand \\@filef@und to avoid getting the wrong file name in the case of a substitution. 1045
2020-11-23 ltshipout.dtx v1.0d
__shipout_execute_cont:: Check for both kernel and user hook (gh/431) 1069
__shipout_execute_main_cont:Nnnn: Check for both kernel and user hook (gh/431) 1071
2020-11-24 ltexpl.dtx v1.2g
General: Support for roll forward (gh/434) 75–77
2020-11-24 ltfilehook.dtx v1.0d
General: Support for roll forward (gh/434) 1043
2020-11-24 lthooks.dtx v1.0f
__hook_end_document_label_check:: Support for roll forward (gh/434) 237
2020-11-24 ltshipout.dtx v1.0d
General: Support for roll forward (gh/434) 1083
\\AtBeginDvi: Support for roll forward (gh/434) 1082
2020-11-25 ltdefns.dtx v1.5o
\\@carcube: Added missing latexrelease entry 83
2020-12-02 ltluatex.dtx v1.1s
General: Fix return value of list callbacks 65
2020-12-03 ltfsstrc.dtx v3.0m
\\selectfont: Install a hook in \\selectfont (gh/444) 577
2020-12-04 ltfilehook.dtx v1.0d
\\undeclare@file@substitution: Don't drop file substitution commands on rollback 1048
2020-12-04 ltfssaxes.dtx v1.0h
General: Reorganized the rollback data 551
\\fontseries: Distangle series and shape update (gh/444) 560
\\fontshape: Distangle series and shape update (gh/444) 568
\\fontshapeforce: Distangle series and shape update (gh/444) 568
2020-12-04 ltfssini.dtx v3.2f
General: Adjust start values for series and shape (gh/444) 659
2020-12-10 ltbibl.dtx v1.1s
\\nocite: Delay any \\nocite in the preamble instead of raising an error 940
2020-12-10 ltfssbas.dtx v3.2h
\\usefont: Drop "m" if the series value is a member of a fixed list and issue warning if doing it (gh/453) 532
2020-12-14 ltclass.dtx v1.3v
\\@currnamestack: Removed \\@expl@@@hook@curr@name@push@@n 980
2020-12-18 ltexpl.dtx v1.2h
\\@kernel@after@enddocument@afterlastpage: Define kernel \\enddocument hooks early 73
2020-12-22 ltfssaxes.dtx v1.0h
\\delayed@merge@font@series: Distangle series and shape update (gh/444) 562, 563
\\delayed@merge@font@shape: Distangle series and shape update (gh/444) 569
2020-12-22 ltfsstrc.dtx v3.0n
\\selectfont: Execute delayed series and shape updates (gh/444) . . . 576
2021-01-07 ltfilehook.dtx v1.0e
General: Added rollback for this case to avoid spurious errors (part of gh/463) 1057
\\unqu@tefilef@und: Restore \\CurrentFile(Path)(Used) after the input (gh/464) 1046
2021-01-07 lthooks.dtx v1.0h
__hook_strip_double_slash:w: Assume hook name has at least three nonempty parts (gh/464) . 249

__hook_tl_set:cn: Manually define some l3tl commands to work around expl3 changes 224
2021-01-08 ltshipout.dtx v1.0f
__shipout_execute_cont:: Added another kernel hook for more flexibility (cf. https://github.com/pgf-tikz/pgf/issues/1900 1069
2021-01-10 ltshipout.dtx v1.0g
\@kernel@after@shipout@background: Internal hook \@kernel@after@shipout@background added 1073
\RawShipout: Macro added 1072
2021-01-19 ltshipout.dtx v1.0h
__shipout_run_firstpage_hook:: Handling of firstpage hook altered 1073
2021-01-21 ltclass.dtx v1.3w
\@kernel@currpathstack: Add empty entry for latexrelease 982
2021-01-21 ltexpl.dtx v1.3a
General: Move xparse rollback code to ltcmd.dtx 76
2021-01-21 ltfinal.dtx v2.2l
General: Load glyphtounicode.tex for pdfTEX 1182
2021-01-22 ltshipout.dtx v1.0i
__shipout_finalize_box:: Add pre_ shipout_ filter Lua callback 1068
2021-01-24 ltexpl.dtx v1.3a
General: Define expl3 hooks conditionally 73
2021-01-31 ltfilehook.dtx v1.0f
\@curr@file@reqd: set \protect to \string gh/481 1049
2021-02-03 ltfloat.dtx v1.2e
\@savemarbox: Explicitly end with \par (gh/489) 924
2021-02-04 ltboxes.dtx v1.4b
\color@endbox: Always add the color groups (gh/488) 822
2021-02-08 ltfilehook.dtx v1.0g
\unqu@tefilef@und: Undo the internal for robust \InputIfFileExists in rollback (gh/494) 1047
2021-02-08 ltmiscen.dtx v1.1y
\begin: Undo the internals for robust \begin and \end in rollback (gh/494) 770
\end\verbvisiblespace: Undo the internals for robust \begin and \end in rollback (gh/494) 773
2021-02-10 ltboxes.dtx v1.4b
\@mpfootnotetext: Explicitly run \par in support for paragraph tagging 832
2021-02-10 ltfloat.dtx v1.2e
\@footnotetext: Explicitly run \par at the end of footnote text in preparation for paragraph hooks 930
2021-02-15 ltfssdcl.dtx v3.0w
\document@select@group: fix for (gh/501) 608
2021-02-16 ltfloat.dtx v1.2f
\footref: \footref added 933
2021-02-17 ltoutenc.dtx v2.0t
General: Adjust values for \textasteriskcentered To match TS1 definition (gh/502) 503
Special definition for \textasteriskcentered when missing in TS1 (gh/502) 494
2021-02-18 ltclass.dtx v1.3x
\@fileswithoptions: save raw class option list (gh/85) 999
\@remove@eq@value: macro added (gh/85) 991
\@use@ption: value from unused option list (gh/85) 994
\OptionNotUsed: value from unused option list (gh/85) 991
\PassOptionsToClass: save raw option lists (gh/85) 990
2021-02-19 ltoutenc.dtx v2.0u
General: Add \textnonbreakinghyphen, \textfiguredash and \texthorizontalbar (gh/404) 483, 487, 502
2021-02-25 ltfinal.dtx v2.2m
General: Improve speed of ToUnicode everyjob loading code 1182
2021-03-03 ltclass.dtx v1.3y
\endfilecontents: Fix overwrite check for files with UTF-8 (gh/415) 1011
2021-03-05 ltclass.dtx v1.3z
\ProcessOptions*: modify so braces to not give errors (gh/513) 992
2021-03-12 ltfiles.dtx v1.2k
\IfFileExists@: Allow unbalanced conditionals (gh/530) 457

2021-03-18 ltcmd.dtx v1.0b
General: Use \NewModuleRelease. . 113
2021-03-18 ltfilehook.dtx v1.0h
__filehook_file_pop_assign:nnnn: Define \g_@@_input_file_seq to avoid losing data when rolling back. 1043
2021-03-18 ltfssaxes.dtx v1.0i
General: Fix rollforward definition. . 561
2021-03-18 ltfssini.dtx v3.2g
General: Add legacy hook definitions for rollback. 649
2021-03-18 lthooks.dtx v1.0i
__hook_end_document_label_check:: Only add top-level if not already there. 237
Remove the (empty) "top-level" from \@currnamestack. 237
General: Use \NewModuleRelease. . 222
2021-03-18 ltvers.dtx v1.1f
\@check@IncludeInRelease: Add support for usage in \NewModuleRelease 39
\new@moduledate: Added \NewModuleRelease. 40
2021-03-19 lttextcomp.dtx v1.0e
General: Use \NewModuleRelease . . 695
2021-03-26 ltplain.dtx v2.3e
\@unused: Allocate \@inputcheck and \@unused early so that they are before expl3 allocates more streams (gh/538) 23
2021-03-27 ltclass.dtx v1.4a
\@currnamestack: Do not completely roll back if expl3 is loaded. . . . 981
2021-04-16 ltvers.dtx v1.1g
\new@moduledate: \NewModuleRelease with the same arguments as \IncludeInRelease. 40
2021-04-17 ltfiles.dtx v1.2m
\@kernel@after@begindocument: Move \@kernel@before@begindocument and \@kernel@after@begindocument init earlier so that other modules can write to the hooks 447
2021-04-18 ltluatex.dtx v1.1t
General: input_ level_ string added . 63
2021-04-18 ltplain.dtx v2.3f
\loggingall: 3 33
Drop pre-ε-TEX support 33
\tracingnone: 3 34
Drop pre-ε-TEX support 34
2021-04-19 ltcmd.dtx v1.0d
__cmd_cmd_type_cases:NnnnnnF: Renamed __cmd_cmd_if_xparse:NTF to __kernel_cmd_if_xparse:NTF for cross-module usage 181
2021-04-20 ltexpl.dtx v1.3c
\@kernel@after@enddocument@afterlastpage: Don't empty kernel hooks on rollback 73
2021-04-20 ltfilehook.dtx v1.0i
\@curr@file@reqd: Make expand to a string (tracks change in l3kernel) 1049
2021-04-26 ltfssbas.dtx v3.2i
\usefont: Unconditionally switch to the requested font face (gh/444) 532
2021-04-26 ltfssini.dtx v3.2h
\reset@font: Unconditionally switch to the requested font face (gh/444) 657
2021-04-26 ltfsstrc.dtx v3.0o
\selectfont: Unset the forced series boolean when reaching \selectfont (gh/444) 577
2021-04-29 lthooks.dtx v1.0m
\ActivateGenericHook: Add \ProvideHook etc. 296
2021-04-29 ltoutenc.dtx v2.0v
General: Add composites for \ae/\AE/æ/Æ (gh/552) 507
2021-05-18 ltclass.dtx v1.4b
\@raw@classoptionslist: Initialise to \relax to match \@classoptionslist 979
2021-05-24 ltcmd.dtx v1.0e
General: Use \msg_... instead of __kernel_msg... 113
2021-05-24 ltcmdhooks.dtx v1.0b
General: Use \msg_... instead of __kernel_msg... 308
2021-05-24 ltfilehook.dtx v1.0k
General: Use \msg_... instead of __kernel_msg... 1041
2021-05-24 lthooks.dtx v1.0n
General: Use \msg_... instead of __kernel_msg... 222
2021-05-24 ltpara.dtx v1.0g
General: Use \msg_... instead of __kernel_msg... 409
2021-05-26 ltdefns.dtx v1.5p
\MakeRobust: Normalize error message in \MakeRobust 92

2021-06-03 ltclass.dtx v1.4c
\@kernel@currpathstack: Take care of \@kernel@currpathstack when rolling back/forward. 982
2021-06-04 ltcmd.dtx v1.0f
General: Normalize various error messages 183
2021-06-05 ltmiscen.dtx v1.1z
\@@sverb: Normalize error message . 781
2021-06-06 ltclass.dtx v1.4c
\@loadwithoptions: handle raw options for gh/580 996
\load@onefile@withoptions: Copy raw options for gh/580 1004
\PassOptionsToClass: apply \expandafter to raw options for gh/580 990
2021-06-09 ltclass.dtx v1.4b
\endfilecontents: Use \@latex@note@no@line to display the information 1011
2021-06-09 lterror.dtx v1.2r
\@latex@note@no@line: Macros added 393
2021-06-09 ltfssbas.dtx v3.2j
\DeclareFontShape@: Improve information message 524
2021-07-08 ltcounts.dtx v1.1m
\counterwithin: New implementation for \counterwithout and \counterwithin 516
2021-07-11 lterror.dtx v1.2s
\PackageNoteNoLine: Provide \ClassNote and \PackageNote . 392
2021-07-12 ltclass.dtx v1.4d
\@fileswithoptions: add \unexpanded 999
2021-07-16 ltplain.dtx v2.3g
General: Use 2 as default value for \tracinglostchars 24
2021-07-19 ltclass.dtx v1.4e
\@classoptionslist: Drop \@onlypreamble 979
\@ifclasslater: Drop \@onlypreamble 983
\@ifclassloaded: Drop \@onlypreamble 983
\@ifclasswith: Drop \@onlypreamble 985
\@ifl@ter: Drop \@onlypreamble . 984
\@pkgextension: Drop \@onlypreamble 979
\@ptionlist: Drop \@onlypreamble 983
\@unusedoptionlist: Drop \@onlypreamble 979
\IfFormatAtLeastTF: Drop \@onlypreamble 983
2021-07-20 ltcmdhooks.dtx v1.0c
__hook_patch_DeclareRobustCommand:Nnn: Use \robust@command@chk@safe before \@if@newcommand. 312
2021-07-22 lthooks.dtx v1.0o
__hook_gremove_code:nn: Do not queue removals (gh/625) 251
__hook_prop_gput_labeled_do:Nnnn: Do not queue removals (gh/625) 239
2021-07-23 ltclass.dtx v1.4e
\load@onefile@withoptions: Make class/name/after a one-time hook 1005
Make class/name/before a one-time hook 1004
Make package/name/after a one-time hook 1005
Make package/name/before a one-time hook 1004
2021-07-23 ltfiles.dtx v1.2n
\@include: Make include/name/after a one-time hook 452
Make include/name/before a one-time hook 452
Make include/name/end a one-time hook 452
2021-07-27 lthooks.dtx v1.0o
\ClearHookNext: Macro added 297
\hook_gclear_next_code:n: Macro made public 283
2021-07-28 ltsect.dtx v1.1f
\contentsline: Pick up four arguments (gh/633) 908
2021-07-30 ltcmd.dtx v1.0d
__cmd_cmd_type_cases:NnnnnnF: Added \@@_cmd_type_cases:NnnnnF for \NewCommandCopy and \ShowCommand support 181
2021-07-31 ltoutput.dtx v1.4e
\fl@tracemessage: Enable display when doing \tracefloatvals . 1146
2021-07-31 ltoutput.dtx v1.4g
\ShowFloat: Macro added 1144
2021-08-02 lthooks.dtx v1.0o
\ActivateGenericHook: Change name 296
\DisableGenericHook: Change name 296

2021-08-07 ltcmd.dtx v1.0g
__cmd_add_grabber:N: Replicate argument processors for all embellishments (gh/639) 140
__cmd_add_type_E:w: Replicate argument processors for all embellishments (gh/639) 138
2021-08-08 ltfinal.dtx v2.2p
\IfPDFManagementActiveTF: Default definition added (gh/640) 1192
2021-08-10 ltvers.dtx v1.1h
\@check@IncludeInRelease: Add error to aid debugging 39
2021-08-11 ltluatex.dtx v1.1u
General: Define missing local function 54
2021-08-20 lterror.dtx v1.2t
\@badend: Improve \@badend error message (gh/587) 396
2021-08-20 lthooks.dtx v1.0p
General: Added deprecation warnings for old generic hook commands (gh/638) 299
Documentation updates for generic hook commands (gh/638) 195
Renames of generic hook commands (gh/638) 233
Section on generic hooks added (gh/638) 212
2021-08-25 ltclass.dtx v1.4f
General: Standardise generic hook names (gh/648) 974
\load@onefile@withoptions: Declare non-generic package and class hooks 1005
2021-08-25 ltcmdhooks.dtx v1.0d
__hook_try_put_cmd_hook:w: Simplify generic hook detection . 309
2021-08-25 ltfilehook.dtx v1.0l
\unqu@tefilef@und: Declare non-generic file hooks 1046
2021-08-25 ltfiles.dtx v1.2o
\@include: Declare non-generic include hooks 453
Standardise generic hook names (gh/648) 452
2021-08-25 lthooks.dtx v1.0p
__hook_try_declaring_generic_next_hook:nn Standardise generic hook names (gh/648) 243
2021-08-27 ltcmd.dtx v1.0h
\NewDocumentEnvironment: Check for end-of-environment command . . 189
2021-08-27 ltfilehook.dtx v1.0l
__filehook_file_pop_assign:nnnn: Internal message name changes 1043
__filehook_file_subst_cycle_error:cN: Use \msg_... not __kernel_msg_... 1054
2021-08-27 lthooks.dtx v1.0q
General: Internal message name changes 292
2021-08-27 ltpara.dtx v1.0i
General: Internal message name changes 417
2021-08-30 ltcmd.dtx v1.0h
General: Added support for \NewCommandCopy 144
Added support for \ShowCommand 150
2021-09-03 ltoutput.dtx v1.4h
\ShowFloat: Renamed, original name never distributed 1144
2021-09-06 ltfinal.dtx v2.2q
\@uclclist: Correctly upper and lowercase \ij and \IJ (gh/658) 1190
2021-09-06 lthooks.dtx v1.0r
__hook_if_execute_immediately:n: Macro added (gh/606) 288
__hook_prop_gput_labeled_do:Nnnn: Use dedicated conditional (gh/606) 239
__hook_use_once_clear:n: Clean up after \UseOneTimeHook (gh/606) 288
\hook_use_once:nnw: Clean up after \UseOneTimeHook (gh/606) 286
2021-09-10 ltfssini.dtx v3.2i
\bfseries: Do delayed changes to \bfdefault in a separate macro for better reuse (gh/663) 643
\DeclareFontSeriesDefault: Do delayed changes to \bfdefault or \mddefault first (gh/663) 636
\maybe@update@bfseries@defaults: Do delayed changes to \bfdefault in a separate macro for better reuse (gh/663) 643
\maybe@update@mdseries@defaults: Do delayed changes to \mddefault in a separate macro for better reuse (gh/663) 644
\mdseries: Do delayed changes to \mddefault in a separate macro for better reuse (gh/663) 644
2021-09-12 ltfntcmd.dtx v3.5a
\check@nocorr@: use \unexpanded to make # safe 690

2021-09-12 ltoutenc.dtx v2.0w
General: Move zero skip between i and j for hyphenation (gh/658) 484
2021-09-18 ltpara.dtx v1.0i
`\para_end:`: Use skip rather than kern as guard. 415
2021-09-26 ltfssdcl.dtx v3.0x
`\c@localmathalphabets`: Counter added for (gh/676) 607
`\document@select@group`: Test if we should freeze the version (gh/676) 607
`\freeze@math@version`: Macro added for (gh/676) 610
2021-09-28 ltcmdhooks.dtx v1.0e
`\__hook_make_prefixes:w`: Make patching of commands a global operation (gh/674) 318
`\__hook_patch_retokenize:Nnnn`: Make patching of commands a global operation (gh/674) 325
2021-09-28 lthooks.dtx v1.0s
`\__hook_if_usable_use:n`: Correct usage of older `\@@_if_file_hook:wTF` (gh/675) 286
`\__hook_try_declaring_generic_hook_split:nNNnn`: Correct usage of older `\@@_if_file_hook:wTF` (gh/675) 244
2021-10-14 ltboxes.dtx v1.4c
`\@mpfootnotetext`: Explicitly set `\@currentcounter` (gh/687) . . . 832
2021-10-14 ltfiles.dtx v1.2p
`\@include`: Warn about use in preamble 452
2021-10-14 ltfloat.dtx v1.2g
`\@footnotetext`: Explicitly set `\@currentcounter` (gh/687) . . . 930
2021-10-14 ltmath.dtx v1.2j
`\@eqnsel`: Explicitly set `\@currentcounter` (gh/687) . . . 795
`eqnarray`: Explicitly set `\@currentcounter` (gh/687) . . . 799
2021-10-15 ltfssdcl.dtx v3.0y
`\DeclareMathVersion`: Initialize variable for freezing math version (gh/676) 613
2021-10-15 ltluatex.dtx v1.1v
General: provide_ charproc_ data added 63
2021-10-16 ltoutenc.dtx v2.0x
`\add@accent`: Dont set `\spacefactor in math mode gh/643` 473
2021-10-19 ltpara.dtx v1.0k
General: Remove content from `\tex_everypar:D` on rollback . . . 417
2021-10-20 ltcmdhooks.dtx v1.0f
`\__hook_make_prefixes:w`: Correct patching by expansion+redefinition when the macro contains a parameter token (gh/697). 313
2021-11-17 ltluatex.dtx v1.1w
General: hpack_ quality is `exclusive` 63
Never pass on `true` return values for list callbacks 65
vpack_ quality is `exclusive` 63
2021-11-30 ltclass.dtx v1.5a
`\@onefilewithoptions@clashchk`: Separated out from `\@onefilewithoptions` 1003
2021-11-30 ltexpl.dtx v1.3d
`\skipeval`: Moved over from `xfp` package (gh/711) 77
2021-12-02 ltcmd.dtx v1.0i
`\__cmd_run_code:`: Correct defaults for optional arguments in end-of-environment code (gh/712) 123
`\__cmd_start_aux:ccnnnn`: Correct defaults for optional arguments in end-of-environment code (gh/712) 123
2021-12-07 ltexpl.dtx v1.3d
`\skipeval`: Added `\dimeval` and `\skipeval` (gh/711) 78
2021-12-11 ltdirchk.dtx v1.3a
General: Add comment lines into latex.ltx to indicate temp definitions that are later overwritten (gh/725) 2
2021-12-12 ltoutenc.dtx v2.0y
General: `\DeclareUnicodeAccent` now makes the encoding argument implicit (gh/253) 500
Added `\DeclareUnicodeCommand` and `\DeclareUnicodeSymbol` (gh/253) 500
2021-12-27 ltluatex.dtx v1.1x
`\newprotectedluacmd`: Macros added 51
2021-12-28 ltexpl.dtx v1.3d
`\ExpandArgs`: Added document level names for `\exp_ args:Nc` and the like (gh/735) 78
2021-13-31 ltcmd.dtx v1.0j
`\__cmd_show:x`: Make `\ShowCommand` stop for interaction 152, 153

2022-01-06 ltexpl.dtx v1.3e
\ExpandArgs: Adjust document level names for \exp_ args:Nc and the like (gh/735) 78
2022-01-06 ltshipout.dtx v1.0l
\AtBeginShipoutNext: Correctly simulate \AtBeginShipout and \AtBeginShipoutNext without extending the syntax 1085
\AtEndDvi: Correctly simulate \AtEndDvi without extending the syntax 1084
2022-01-15 ltkeys.dtx v1.0b
__keys_options_aux:n: Clear option list in end-of-package hook . . . 1030
2022-01-25 ltplain.dtx v2.3h
\obeyspaces: Provide redirection to support special use cases (gh/367) 28
2022-02-05 ltkeys.dtx v1.0b
.usage: Create properties in ltlkeys 1028
2022-02-07 ltkeys.dtx v1.0c
.usage: Correct ..code property . . 1028
2022-02-15 ltkeys.dtx v1.0c
\DeclareKeys: Expand module argument 1033
\DeclareUnknownKeyHandler: Added \DeclareUnknownKeysHandler . 1033
\ProcessKeyOptions: Expand module argument 1033
\SetKeys: Expand module argument 1035
2022-02-16 ltkeys.dtx v1.0d
\DeclareKeys: Allow for active characters in module argument 1033
\DeclareUnknownKeyHandler: Allow for active characters in module argument 1033
Rename \DeclareUnknownKeysHandler to \DeclareUnknownKeyHandler . . 1033
\ProcessKeyOptions: Allow for active characters in module argument 1033
\SetKeys: Allow for active characters in module argument 1035
2022-02-19 ltcmd.dtx v1.0k
\IfBlankTF: Added \IfBlankTF and friends 192
2022-02-20 ltfinal.dtx v2.2r
\@uclclist: Use \@expl@text@uppercase@@n, removing local redefinition of \UTF@two@octets@noexpand . . . 1188
use \text_lowercase:n 1188
2022-02-21 ltkeys.dtx v1.0e
__keys_options_expand_module:Nn: Faster approach to module expansion 1033
2022-02-28 ltcmd.dtx v1.0k
General: Move latexrelease redefinitions from ltcmd.dtx . . . 193
2022-02-28 ltexpl.dtx v1.3f
General: Move latexrelease redefinitions from ltcmd.dtx 76
2022-02-28 ltvers.dtx v1.1i
\new@moduledate: Detect a missing \IncludeInRelease{0000/00/00}. 40
2022-03-10 ltbibl.dtx v1.1t
\cite: Ensure that an empty argument generates a warning (gh/790) 938
2022-03-10 ltfilehook.dtx v1.0m
\@curr@file@reqd: Add \set@curr@file@nosearch for graphicx 1049
2022-03-18 ltclass.dtx v1.5b
\load@onefilewithoptions: Switch to \ProcessKeyOptions 1002
2022-03-18 ltcmd.dtx v1.0l
__cmd_cmd_type_cases:NnnnnnF: Fix \@@_cmd_type_cases:NnnnnF prematurely expanding macros (gh/795) 181
2022-03-18 ltkeys.dtx v1.0f
__keys_options:n: Simplify to always cover global options . . . 1030
__keys_options_global:n: Simplify to always cover global options . 1030
\ProcessKeyOptions: Remove \ProcessKeyPackageOptions . . 1033
2022-04-01 ltfiles.dtx v1.2q
\@include: Process some hooks is an include file is bypassed 453
2022-04-01 lthooks.dtx v1.0t
\c__hook_generic_include/./end_tl: Support generic include/.../excluded hooks . . 250
2022-04-03 ltfinal.dtx v2.2s
General: Integration of new mark management interface 1174
2022-04-03 ltoutput.dtx v1.ih
\@opcol: Interface with new mark mechanism 1110
2022-04-04 ltpage.dtx v1.0n
\markright: Interface with new mark mechanism 970, 971
2022-04-08 ltmath.dtx v1.2k
\openup: Make \protected (gh/123) 790

2022-04-12 ltxref.dtx LaTeX2e
\pageref: Macro reimplemented with a starred version 740
\ref: Macro reimplemented with a starred version 740
2022-04-12 ltxref.dtx v1.1p
General: Add starred variants for the ref commands 738
\Ref: Macro reimplemented with a starred version 744
2022-04-21 ltfinal.dtx v2.2t
\@uclclist: Support \noexpand in argument of \@expl@text@uppercase@@n . . . 1188
2022-05-06 ltmarks.dtx v1.0c
__mark_new_class:nn: Wrong command made \@onlypreamble 950
2022-05-08 ltmath.dtx v1.2l
General: Use consistent math styles under LuaTEX 797
2022-05-08 ltshipout.dtx v1.0m
\@kernel@after@enddocument@afterlastpage: Handle case where shipout/lastpage is run too early (gh/813) 1081
__shipout_execute_main_cont:Nnnn: Handle case where shipout/lastpage is run too early (gh/813) 1071
2022-05-13 lthooks.dtx v1.0u
__hook_use_once_clear:n: Check if prop exists to avoid l3debug error 288
2022-05-17 lthooks.dtx v1.0u
__hook_initialize_hook_code:n: Refuse sorting one-time hooks (gh/818). 265
2022-05-17 ltmeta.dtx v1.0b
General: Default definition for targets added 419
2022-05-27 ltfiles.dtx v1.2r
\listfiles: Try saved version string, if ver@.. is \relax (gh/825) . . . 463
2022-05-27 ltoutenc.dtx v2.0z
General: Save the version string (gh/825) 511
2022-06-01 ltmarks.dtx v1.0d
__mark_prepare_and_extract:nn: Extend the logic for detecting the marks in the box (gh/836) 953
General: Marks are kernel errors . . . 962
2022-06-02 ltfinal.dtx v2.2u
\@uclclist: Add \NoCaseChange . 1189
2022-06-15 lthooks.dtx v1.0v
__hook_activate_generic:n: Ensure that a newly activated generic hook gets its execution code set . 234
2022-06-16 ltkeys.dtx v1.0g
__keys_options_class:n: Better handling of option removal . . . 1031
__keys_options_package:n: Better handling of option removal . . . 1031
2022-06-19 ltkeys.dtx v1.0h
__keys_options_class:n: Further work on handling of option removal 1031
__keys_options_package:n: Further work on handling of option removal 1031
__keys_options_package:nnn: New function 1031
2022-06-20 ltclass.dtx v1.5c
\load@onefilewithoptions: Pass raw options to \ProcessKeyOptions 1002
2022-06-20 ltkeys.dtx v1.0h
__keys_options_class:n: Use raw options data 1031
__keys_options_class:nnn: New function 1031
__keys_options_global:n: Use raw options data 1030
__keys_options_local:: Use raw options data 1032
__keys_options_package:n: Use raw options data 1031
2022-06-22 ltkeys.dtx v1.0i
.usage: Add ..notif property 1028
2022-06-30 ltfinal.dtx v2.2u
\@uclclist: Add \AddToNoCaseChangeList 1189
2022-06-30 ltfinal.dtx v2.2v
\@uclclist: Just use \text_lowercase:n without \protectd@edf gh/881x 1188
2022-07-04 ltfinal.dtx v2.2w
\@uclclist: Introduced \CaseSwitch, \DeclareCaseChangeEquivalent and \MakeTitlecase to support hooking into case changing gh/889 1188
2022-07-04 ltfssbas.dtx v3.2k
\frozen@everydisplay: Ignore spaces if necessary (gh/886) 535
\frozen@everymath: Ignore spaces if necessary (gh/886) 535

2022-07-04 ltfssdcl.dtx v3.0z
- \freeze@math@version: Ignore spaces if necessary (gh/886) 611

2022-07-05 ltkeys.dtx v1.0i
- __keys_options_aux:n: Support \CurrentOption 1030
- __keys_options_class:nnn: Correct naming of raw class options storage 1031

2022-07-23 ltkeys.dtx v1.0j
- __keys_options_end:: Output 'public' package name in messages 1030
- __keys_options_loaded:nn: Output 'public' package name in messages 1034

2022-08-07 lttextcomp.dtx v1.0g
- \DeclareEncodingSubset: Make global declaration (gh/905) 697

2022-08-10 ltcmd.dtx v1.1a
- __cmd_arg_to_keyvalue:nn: New internal arg-to-keyval processor . 177
- __cmd_grab_D_long_obey_spaces_no_strip:w: Add support for skipping brace stripping 157
- __cmd_grab_R_aux:NNnN: Track changes in D-type implementation 163
- __cmd_grab_b_end:Nw: Track changes in D-type implementation 156
- General: New switch 115

2022-08-13 ltluatex.dtx v1.1y
- General: shared_callbacks added . . . 64
- add_to_callback: Adapted code for shared_ callbacks 68
- remove_from_callback: Adapted code for shared_ callbacks 70
- mlist_to_hlist: Use shared_callback system for pre/post/mlist_to_hlist 72

2022-08-18 ltfilehook.dtx v1.0n
- \@disable@packageload@do: Inhibit checking the loaded version when package is load-disabled, and write to the .log (gh/888) 1055

2022-08-20 ltoutput.dtx v1.4j
- \stockheight: Register added . . . 1099
- \stockwidth: Register added 1099

2022-08-21 ltkeys.dtx v1.0k
- __keys_options_loaded:nn: Correct error message 1034

2022-09-03 ltmath.dtx v1.2m
- \smash: Guard against reboxing in fractions (gh/517) 789

2022-09-07 ltboxes.dtx v1.4d
- \@iiiminipage: Check for nested minipages and warn (gh/168) . . 831
- \if@in@minipage@env: Check for nested minipages and warn (gh/168) 831

2022-09-17 ltfssdcl.dtx v3.1a
- \DeclareMathVersion: New logic for freezing math versions (gh/921) . 613
- \freeze@math@version: New logic for freezing math versions (gh/921) . 610

2022-09-20 ltfssdcl.dtx v3.1b
- \DeclareSymbolFont: Drop any surplus 'm' in series argument (gh/918) 614
- \SetSymbolFont: Drop any surplus 'm' in series argument (gh/918) 616

2022-10-03 ltluatex.dtx v1.2a
- General: Add rules for callback ordering 60
- add_to_callback: Add rules for callback ordering 68
- declare_callback_rule: Add function 69
- remove_from_callback: Add rules for callback ordering 70

2022-10-10 ltclass.dtx v1.5d
- \@@unprocessedoptions: Use \protected@edef. 1009
- \load@onefilewithoptions: Use \protected@edef. 1002
- \PassOptionsToClass: Use \protected@xdef. 990
- \ProcessOptions*: Use \protected@edef. 992

2022-10-20 ltclass.dtx v1.5e
- \load@onefilewithoptions: Define key option handler in ltkeys . . . 1002

2022-10-20 ltkeys.dtx v1.0l
- __keys_options_aux:n: Define key option handler in ltkeys 1030
- __keys_options_loaded:nn: Correct an argument for a message . . . 1034

2022-10-22 ltclass.dtx v1.5e
- \@fileswithoptions: Use \protected@xdef. 999
- \@use@ption: Use \detokenize . . . 994
- \ProcessOptions*: Use \detokenize 992, 993

2022-10-22 ltdefns.dtx v1.5r
- \@expandtwoargs: Use \protected@edef. 89

2022-10-22 ltkeys.dtx v1.0l
__keys_options_class:nnn: Correct handling of unused option list . 1031
2022-10-26 ltfinal.dtx v2.2x
\@uclclist: Auto-detect babel locale 1188
Introduce optional argument for case-changing commands 1188
Make case changing commands language-aware 1188
2022-11-06 ltmiscen.dtx v1.2a
\enddocument: Repeat release info at the end (gh/944) 762
2022-11-06 ltvers.dtx v1.1j
General: Repeat release info at the end (gh/944) 38
2022-11-08 ltshipout.dtx v1.0n
__shipout_execute_main_cont:Nnnn: Add shipout hook (gh/920) . . . 1071
shipout/lastpage: Add shipout hook (gh/920) 1073
2022-11-16 ltclass.dtx v1.5f
\endfilecontents: Do not show "current dir" in message (gh/917) 1012
Introduce key 'nowarn' on filecontents (gh/958) 1010–1012
2022-11-24 ltdefns.dtx v1.5s
\DeclareEnvironmentCopy: Add \NewEnvironmentCopy, \RenewEnvironmentCopy, and \DeclareEnvironmentCopy (gh/963) 99
\ShowEnvironment: Added \ShowEnvironment 104
2022-11-28 ltspace.dtx v1.3o
\@hspace: Support calc syntax correctly (gh/967) 439
\@vspace@calcify: Support calc syntax without a group (gh/967) 426
2022-11-29 ltcmd.dtx v1.1b
__cmd_show:x: Add \@showenvironmentlisthook . . . 153
2022-11-30 ltcmd.dtx v1.1b
__cmd_show:x: Don't stop for the \begin part of an environment 152, 153
2022-11-30 ltfinal.dtx v2.2y
\@uclclist: Set \oe/\OE equal to act as a marker for babel 1189
2023-01-05 ltfiles.dtx v1.2s
\document: \do now with default definition in the kernel (gh/975) 446
2023-01-19 ltluatex.dtx v1.2b
General: Remove unused local variable tex_setattribute 54
2023-01-30 ltpara.dtx v1.0l
\g__para_standard_everypar_tl: Backout \parskip at top of minipage (gh/989) 410
2023-03-12 ltcmd.dtx v1.1c
__cmd_copy_expandable:NnNNNNnnn: Distinguish (non-expandable) document commands starting with \@@_start_expandable:nNNNNn . 146
__cmd_show:x: Distinguish (non-expandable) document commands starting with \@@_start_expandable:nNNNNn . 151
2023-03-22 ltspace.dtx v1.3p
\samepage: Add \predisplaypenalty setting (gh/1022) 424
2023-03-28 ltclass.dtx v1.5g
\IfFormatAtLeastTF: Added \IfFileAtLeastTF (gh/1015) . . . 983
2023-03-28 ltfinal.dtx v2.2z
\@uclclist: Use groups for gh/1021 1189
2023-04-01 ltfssbas.dtx v3.2l
\math@version: Reset frozen mathversion gh/1028 534
2023-04-02 ltfilehook.dtx v1.0o
\@set@curr@file@aux: Make \@set@curr@file@aux \long gh/942 1051
2023-04-06 ltcmdhooks.dtx v1.0g
__hook_double_hashes_space:w: Add case for \c_@@_hashes_tl (hook-args). 319
__hook_make_prefixes:w: Rename to \c_@@_hashes_tl (hook-args). 315
\c__hook_hashes_tl: Rename to \c_@@_hashes_tl and add \c_@@_hash_tl (hook-args). 309
2023-04-06 lthooks.dtx v1.1a
__hook_activate_generic:n: Changes to add hook arguments (hook-args). 234
__hook_braced_real_loop:w: Macro added (hook-args). 258
__hook_chk_args_allowed:nn: Macro added (hook-args). 242
__hook_clear_next:n: Changes to add hook arguments (hook-args). 283
__hook_code_gset_aux:nnn: Macro added (hook-args). 254
__hook_code_gset_auxi:eeen: Macro added (hook-args). 252

__hook_cs_end:w: Macro added (hook-args). 258
__hook_cs_if_empty:c: Macro added (hook-args). 257
__hook_gput_next_do:nn: Changes to add hook arguments (hook-args). 282
__hook_gremove_code:nn: Changes to add hook arguments (hook-args). 251
__hook_if_execute_immediately:n: Changes to add hook arguments (hook-args). 288
__hook_init_structure:n: Changes to add hook arguments (hook-args). 230
__hook_initialize_all:: Changes to add hook arguments (hook-args). 263
__hook_initialize_hook_code:n: Changes to add hook arguments (hook-args). 265
__hook_initialize_single:ccn: Changes to add hook arguments (hook-args). 268
__hook_log:nN: Changes to add hook arguments (hook-args). 275
__hook_log_next_code:n: Changes to add hook arguments (hook-args). 279
__hook_make_usable:nn: Changes to add hook arguments (hook-args). 228
__hook_new:nn: Add \hook_new_with_args:nn (hook-args). 227
Update hook code after declaring. 228
__hook_new_reversed:nn: Add \hook_new_reversed_with_-args:nn (hook-args). 231
__hook_normalise_cs_args:nn: Macro added (hook-args). 254
__hook_parameter:n: Macro added (hook-args). 259
__hook_post_initialization_defs:: Macro added (hook-args). 285
__hook_prop_gput_labeled_do:Nnnn: Add \hook_gput_code_with_args:nnn (hook-args). 239
Changes to add hook arguments (hook-args). 239, 240
Macro added (hook-args). 241
__hook_set_normalise_fn:nn: Macro added (hook-args). 255
__hook_tl_set:cn: Clean-up unused variants (hook-args). 224
__hook_try_declaring_generic_hook:wn: Changes to add hook arguments (hook-args). 245
__hook_use_i_delimit_by_s_mark:nw: Use standard naming scheme (hook-args). 224
__hook_use_initialized:nnw: Add \hook_use:nnw (hook-args). 285
__hook_use_once:nn: Changes to add hook arguments (hook-args). 287
__hook_use_once_clear:n: Changes to add hook arguments (hook-args). 288
General: Add dedicated rollback code to revert data structures (hook-args). 302
Messages 'too-many-args', 'without-args' and 'one-time-args' added (hook-args). 293
\AddToHookNextWithArguments: Add \AddToHookNextWithArguments (hook-args). 297
\AddToHookWithArguments: Add \AddToHookWithArguments (hook-args). 296
\c__hook_??_parameter_tl: Token list added (hook-args). 260
\c__hook_nine_parameters_tl: Add auxiliary token lists (hook-args). 224
\g__hook_replacing_stack_seq: Macro added (hook-args). 292
\hook_gput_next_code:nn: Add \hook_gput_next_code_with_-args:nn (hook-args). 281
\hook_if_empty:n: Changes to add hook arguments (hook-args). . . . 289
\hook_new_pair_with_args:nnn: Add \hook_new_pair_with_args:nnn (hook-args). 231
\hook_use_once:nnw: Add \hook_use_once:nnw (hook-args). 286
\NewMirroredHookPairWithArguments: Add \NewHookWithArguments (hook-args). 296
\UseOneTimeHookWithArguments: Add \UseHookWithArguments (hook-args). 298

2023-04-11 ltfinal.dtx v2.3a
\@uclclist: Use new generic mechanism to detect locale . . . 1188
2023-04-13 ltcmd.dtx v1.1d
__cmd_grab_v_group_end:: Set \tex_endlinechar:D earlier (gh/876). 164
2023-04-14 ltclass.dtx v1.5h
\load@onefilewithoptions: Define \load@onefilewithoptions when in latexrelease (gh/992) 1001
2023-04-15 ltplain.dtx v2.3i
\extrafloats: Protect box 255 in lualatex gh/1041 21
unwind numexpr and ifnum nesting 21
2023-04-16 ltfinal.dtx v2.3a
\BCPdata: Command added 1187
2023-04-16 lthooks.dtx v1.1b
__hook_include_legacy_code_chunk:n: \@@_replacing_args_false: in \@@_include_legacy_code_-chunk:n. 232
2023-04-19 ltfinal.dtx v2.3b
\@uclclist: Add \DeclareLowercaseMapping, \DeclareTitlecaseMapping and \DeclareUppercaseMapping . . . 1189
2023-04-19 lthooks.dtx v1.1c
__hook_gput_next_do:nn: Initialise hook structure when adding 'next' code (gh/1052). 282
\hook_if_empty:n: Simpler and faster version (gh/1052). 289
2023-05-12 ltxref.dtx v1.1q
\label: (UFi)added a hook with argument 742
(UFi)extended to store five arguments 742
2023-05-13 ltmath.dtx v1.2n
\leqno: add \ignorespaces gh/1059 797
2023-05-15 ltfiles.dtx v1.2t
\IfFileExists@: Use expl3 file existence test 457
\IfFileExists@@: Macro added . . . 457
2023-05-21 ltcmdhooks.dtx v1.0h
__hook_guess_arg_count:nw: Macro added (cmd-args). 323
__hook_make_prefixes:w: Changes to allow support arguments in cmd hooks (cmd-args). 316
__hook_patch_retokenize:Nnnn: Changes to allow support arguments in cmd hooks (cmd-args). 324
2023-05-21 lthooks.dtx v1.1d
__hook_if_cmd_hook:w: Changes to allow support arguments in cmd hooks (cmd-args). 291
__hook_make_usable:nn: Changes to allow support arguments in cmd hooks (cmd-args). 229
__hook_new:nn: Changes to allow support arguments in cmd hooks (cmd-args). 228
__hook_try_declaring_generic_hook:wn: Changes to allow support arguments in cmd hooks (cmd-args). 245
__hook_try_declaring_generic_next_hook:nn: Changes to allow support arguments in cmd hooks (cmd-args). 243
\c__hook_parameter_cmd/./after_tl: Token lists added (cmd-args). . . 250
2023-05-26 ltcmd.dtx v1.1e
__cmd_defaults_error:w: Use simpler variant \cs_generate_-from_arg_count:NNno 124
2023-05-30 ltfinal.dtx v2.3c
\@uclclist: Fix a typo in implementation of \DeclareLowercaseMapping, etc. 1189
2023-06-06 lthooks.dtx v1.1e
__hook_code_gset_auxi:eeen: Short-circuit when the hook is declared without args (gh1078). . 252
2023-06-14 ltmiscen.dtx v1.2b
\@vobeytabs: Macro added 777
verbatim*: Support visible tabs in \verb* 779
2023-06-14 ltspace.dtx v1.3q
\@xobeytab: Macro added 438
2023-06-15 ltmiscen.dtx v1.2b
\@@sverb: Support visible tabs 781
\@setupverbvisiblespace: Support visible tabs 780
\@vobeyspaces: Support tabs 777
2023-06-15 ltmiscen.dtx v1.2q
\@setupverbvisibletab: Provide visible tab in \verb* 781
2023-06-16 ltfiles.dtx v1.2u
\IfFileExists@@: Support piped input 457
2023-07-02 ltluatex.dtx v1.2c
new_luafunction: Ensure existing table entries not overwritten gh/1100 59

2023-08-03 ltluatex.dtx v1.2c
mlist_to_hlist: Fix callback type of post__mlist__to__hlist__callback . . . 72
2023-08-19 ltcmd.dtx v1.2a
General: Removed commands that should have remained only in xparse.dtx 113
2023-09-01 ltmiscen.dtx v1.2c
\@vobeytabs: Provide global definition for active tab 777
2023-09-06 ltmiscen.dtx v1.2b
\enddocument: Test changes of values in \new@label@record 762
2023-09-20 ltproperties.dtx v1.0c
\RecordProperties: use \protected@edef for safer handling of active chars. 753
\SetProperty: use \protected@edef 752
2023-10-13 ltexpl.dtx v1.3g
\IfExplAtLeastTF: Provide a test for expl3 date (gh/1004) 79
2023-10-26 ltboxes.dtx v1.4e
\@iframebox: Guard against unknown alignment gh/1072 826
\@imakebox: Guard against unknown alignment gh/1072 821
\@parboxto: Guard against unknown alignment gh/1072 828
2023-10-26 ltspace.dtx v1.3r
\nobreakspace: Protected definition for tilde 437
2023-11-07 ltcounts.dtx v1.1n
\@definecounter: Do not change \the... if already defined (gh/823) 514
2023-11-07 ltoutenc.dtx v2.1a
General: Add more explanation to error message (gh/1102) 510
2023-11-07 ltplain.dtx v2.3j
\tracingnone: Set \tracinglostchars to 2 in \tracingnone (gh/549) 35
2023-11-15 ltfiles.dtx v1.2v
\if@listfiles@hashes: Extend file list information 463
\if@listfiles@sizes: Extend file list information 463
\listfiles: Extend file list information 463
2023-11-16 ltpara.dtx v1.0m
\g__para_standard_everypar_tl: Correct error message: hook left horizontal not vertical mode (gh/1182) 410
2023-12-01 ltcmd.dtx v1.2b
__cmd_cmd_type_cases:NnnnnnF: Extend for optimized commands 181
__cmd_start_optimized:: Optimize cmd creation for all-m arguments 118
2023-12-22 ltcmd.dtx v1.2c
General: Generalize message invalid-bang (gh/1198) 183
2024-01-03 lthooks.dtx v1.1g
__hook_log:nN: Fix expansion of \@@_print_args:nn argument (gh/1221). 275
2024-01-13 ltkeys.dtx v1.0m
__keys_options_class:n: Trim spaces off key names 1031
__keys_options_package:n: Trim spaces off key names 1031
2024-01-17 ltproperties.dtx v1.0d
__property_record_value_aux:e: Use \protected@write 752
2024-01-24 lthooks.dtx v1.1h
__hook_if_usable_use:n: Correct usage of older \@@_if_file_hook:wTF (gh/1243) 286
__hook_try_declaring_generic_hook_split:nNNnn: Correct usage of older \@@_if_file_hook:wTF (gh/1243) 244
2024-01-27 lttextcomp.dtx v1.1a
General: Added check file for encoding subset 727
Adjusted/corrected TS1 sub-encoding declarations for various families (gh/1257) 707
2024-01-29 ltmarks.dtx v1.0e
__mark_extract_and_handle_marks:nn: Macro added 952
__mark_update_structure_from_material:nn: Macro renamed 955
\ShowMarksAt: Macro added 964
2024-01-30 ltclass.dtx v1.5i
\@fileswithoptions: Test group level 998
2024-03-18 ltthm.dtx v1.0g
\@endtheorem: Insert link target in the label (UFi) 897
\@thm: Use \@kernel@refstepcounter to avoid an unwanted target (UFi) 896
2024-03-21 ltcmd.dtx v1.2d
__cmd_grab_v_aux_put:N: Collect \endlinechar as \obeyedline . . 167
2024-03-22 ltclass.dtx v1.5j
\load@onefilewithoptions: Apply one-step expansion to raw option

list, to be consistent with change for gh/580 (gh/1298). 1002
2024-04-10 ltclass.dtx v1.5k
\IfClassLoadedWithOptionsTF: Provide T and F conditionals not just TF (gh/1262) 986
\IfFileLoadedF: Provide \IfFileLoadedTF and variants (gh/1222) 986
2024-04-17 ltcmd.dtx v1.2e
__cmd_cmd_type_cases:NnnnnnF: Use __kernel_cs_parameter_spec:N instead of \cs_argument_spec:N/\cs_parameter_spec:N . 181
2024-04-17 ltcmdhooks.dtx v1.0j
General: Use __kernel_cs_parameter_spec:N instead of \cs_argument_spec:N/\cs_parameter_spec:N . 308
2024-04-17 ltdefns.dtx v1.5t
General: Rename \@expl@cs@argument@spec@@N to \@expl@cs@parameter@spec@@N (gh/1014) 80
2024-04-17 ltexpl.dtx v1.3h
General: Add a kernel-level copy of \cs_parameter_spec:N 77
Rename \@expl@cs@argument@spec@@N to \@expl@cs@parameter@spec@@N (gh/1014) 77
Update name of expl3 function . . . 77
2024-04-17 ltproperties.dtx v1.0e
\IfLabelExistsF: Renamed \IfLabelExistTF to \IfLabelExistsTF (gh/1262) . . . 756
\IfPropertyExistsF: Renamed \IfPropertyExistTF to \IfPropertyExistsTF (gh/1262) 755
2024-04-18 ltboxes.dtx v1.4f
\@finalstrut: Use a \hrule strut not a \vrule if already in vertical mode (bug seen first with footmisc/14) 835
2024-04-22 lttextcomp.dtx v2.1a
General: Drop default option info (gh/1333) 716
2024-04-24 lttextcomp.dtx v2.1b
General: Load the 2018 version when rolling back prior to 2018-08-11 (gh/1333) 716
2024-05-30 ltmarks.dtx v1.0f
__mark_id:n: Use sequence marker to make all marks unique on nearby regions (gh/1359) 960
__mark_update_dblcol_structures:: Correct logic for first mark in page region if first column contains no marks (gh/1359) 967
\mark_insert:nn: Use sequence marker to make all marks unique on nearby regions (gh/1359) . . . 960
insertmark: Use sequence marker to make all marks unique on nearby regions (gh/1359) 961
2024-05-31 ltmarks.dtx v1.0g
__mark_new_class:nn: Initialize all marks with an id, use 0 when a new class is made (gh/1359) . . . 951
\mark_use_last:nn: Remove the id when returning the mark value (gh/1359) 961
2024-06-04 ltclass.dtx v1.5l
\@cls@pkg: Allow for extensions other than "cls" and "pkg" 1008
\@onefilewithoptions: New argument \pkgcls@ext (gh/870) 1019
\pkgcls@parse@date@arg: New argument \pkgcls@ext (gh/870) 1021
2024-06-04 ltluatex.dtx v1.2d
provides_module: Add v to version string if required (gh/1364) 54
2024-06-10 ltboxes.dtx v1.4g
\@finalstrut: Always use a \vrule strut after all, but back up by a baseline if already in vertical mode. Otherwise empty table p-cells will not get the correct width (bug seen first with colortbl) 835
2024-06-10 lttagging.dtx v1.0b
\tbl_crcr:n: Always issue a \crcr even if we are at the start of a row to avoid problems with tabulary and similar code 1171
2024-06-17 ltfssbas.dtx v3.2m
\showhyphens: set \tracinglostchars to 0 in \showhyphens 547
2024-06-19 ltkeys.dtx v1.0n
__keys_options_class:n: Refactor function 1031
__keys_options_class:nn: New function 1031
__keys_options_class:nnn: Refactor function 1031
Track options used by classes . . 1031
__keys_options_package:nn: New functions 1031

__keys_options_package:nnn: Skip options given to packages 1031
__keys_scope:N: New function . . 1028
__keys_scope:n: New function . . 1028
\l__keys_class_only_clist: New variable 1029
\l__keys_forced_global_clist: New variable 1029
.pass-to-packages: New key property 1028

2024-06-20 ltkeys.dtx v1.0o
__keys_scope:N: Ensure only key name is stored 1028

2024-06-23 ltboxes.dtx v1.4h
\color@endgroup: Adjust for new @endpe handling 823

2024-06-23 ltlists.dtx v1.0u
\@doendpe: Set \if@endpe to false before calling \par (needed for tagging) 812
\propagate@doendpe: Set \if@endpe globally and also set up migration to the outside 813

2024-06-23 ltmiscen.dtx v1.2d
\begin: Separate \begin and \end definitions for individual rollback 769
\end: Separate \begin and \end definitions for individual rollback 770
\end\verbvisiblespace: Adjust for new @endpe handling 772

2024-06-23 ltpara.dtx v1.0n
\g__para_standard_everypar_tl: Append \everypar toks to \g_@@_standard_everypar_tl, rollback 2023/06/01 (gh/1386) . . 411

2024-06-24 fontdef.dtx v3.0j
General: load ts1 cmr fd file in Unicode engines 666

2024-07-06 ltcmd.dtx v1.2f
__cmd_declare_env:ennn: Use space-trimmed envname directly (gh/1399) 121
\NewDocumentEnvironment: Trim spaces from envname first (gh/1399) 189

2024-07-10 ltmiscen.dtx v1.2e
General: Drop code chunks before adding them to avoid duplication in rollback (gh/1407) 764
\enddocument: Drop code chunks before adding them to avoid duplication in rollback (gh/1407) 762

2024-07-13 lttagging.dtx v1.0c
tagsupport/tbl/leaders/end: Sockets for \cline leaders added (tagging/134) 1165

2024-08-03 ltclass.dtx v1.5m
\pkgcls@use@this@release: Add selected release to the file list (gh/1413) 1023

2024-08-09 lthooks.dtx v1.1i
__hook_normalize_hook_args_aux:Nn: Distinguish between empty label and empty hook (gh/1423) 236
__hook_parse_dot_label:nN: Distinguish between empty label and empty hook (gh/1423) 235
__hook_parse_label_default:nN: Distinguish between empty label and empty hook (gh/1423) 235
General: New message “empty-hook” (gh/1423). 294

2024-08-16 ltluatex.dtx v1.2e
provides_module: Support missing version string (gh/1443) 54

2024-09-03 fontdef.dtx v3.0k
General: Load also t1cmss.fd and t1cmtt.fd 665

2024-09-04 ltlength.dtx v1.1e
\@settodim: (UF) Suspend tagging . 521

2024-09-05 ltproperties.dtx v1.0f
__property_record_value_aux:e: Remove \if@filesw test to be in line with \label, tagging-project issue 696 752

2024-09-10 ltspace.dtx v1.3s
\addpenalty: Drop unnecessary \@noitemerr and instead generate \@LRmoderr if we are in restricted hmode (gh/1460) 433
\addvspace: Drop unnecessary \@noitemerr and instead generate \@LRmoderr if we are in restricted hmode (gh/1460) 432

2024-09-11 lterror.dtx v1.2u
\@LRmoderr: Error message added . 397

2024-09-13 lttagging.dtx v1.0h
tagsupport/float/hmode/end: Sockets for floats added 1166

2024-09-20 ltcounts.dtx v1.1o
\@addtoreset: add the parent theHfoo if a counter is reset 515
\@definecounter: define theHfoo (used for internal links) 514

2024-09-20 lttagging.dtx v1.0h
General: moved
`\@kernel@refstepcounter` into ltxref 1171
2024-09-20 ltxref.dtx v1.1r
`\refstepcounter`: add sockets 742
provide a kernel copy kernel@refstepcounter 742
set also currentHref 742
2024-09-21 ltmeta.dtx v1.0c
General: Added tagging support . . . 419
2024-09-25 ltproperties.dtx v1.0g
`\IfLabelExistsF`: Fixed definitions of `\IfLabelExistsT` and `\IfLabelExistsF` 756
2024-10-10 lttagging.dtx v1.0i
`default`: Restore also paratagging (tagging/723) 1163
2024-10-12 ltmiscen.dtx v1.2f
`\begin`: Make `\begin` engine-protected 769
2024-10-21 lthooks.dtx v1.1j
`\IfHookEmptyF`: Define `\IfHookEmptyT`, `\IfHookEmptyF` . 299
2024-10-21 ltmarks.dtx v1.0h
`\IfMarksEqualF`: Define `\IfMarksEqualT`, `\IfMarksEqualF` 964
2024-10-21 ltproperties.dtx v1.0h
`\IfPropertyRecordedF`: Define `\IfPropertyRecordedT`, `\IfPropertyRecordedF` 756
2024-10-21 lttagging.dtx v1.0k
`\UseExpandableTaggingSocket`: Added expandable variants . . . 1162
Changed behavior of two argument tagging sockets when disabled. . 1162
2024-10-26 ltcounts.dtx v1.1p
`\@addtoreset`: Fully expand counter name in theHfoo commands (gh/1508) 515
2024-10-27 ltsockets.dtx v0.9b
`\socket_set_plug:nnn`: Make plug definition non-protected 338
`\socket_use_expandable:n`: Added `\socket_use_expandable:n` 340
2024-10-29 lthooks.dtx v1.1k
`\__hook_list_if_rule_exists:nnnF`: Skip mapping over undeclared `\g_@@_`⟨*hook*⟩`_code_prop` (gh/1513) 280
`\__hook_log:nN`: Skip mapping over undeclared `\g_@@_`⟨*hook*⟩`_code_prop` (gh/1513) 275

v1.0b lttemplates.dtx 2024-02-15
`\IfInstanceExistsTF`: New macros 380
v1.0c lttemplates.dtx 2024-04-17
`\IfInstanceExistsTF`: Use plural names 380
v1.0d lttemplates.dtx 2024-10-07
`\__template_assign_variable:`: Correct passing of `\KeyValue` contents 373

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols

\! 02:431, 02:433, 57:518
\! 38:214
\" 04:512, 04:513, 21:226, 21:378, 21:419, 21:457, 21:468, 21:579, 21:611, 21:638, 21:646, 21:652, 21:656, 21:662, 21:666, 21:672, 21:678, 21:685, 21:686, 21:692, 21:696, 21:746, 21:792, 21:1244, 21:1262, 21:1268, 21:1272, 21:1278, 21:1282, 21:1288, 21:1294, 21:1301, 21:1302, 21:1308, 21:1312, 21:1314, 21:1421, 24:441, 33:171, 33:198, 57:519
\# 01:46, 01:59, 02:6, 02:14, 02:531, 06:801, 09:446, 24:428, 57:502
\$ 01:58, 02:4, 02:13, 06:800, 21:307, 21:444, 21:451, 21:565, 21:804, 21:811, 33:631, 33:638, 57:503
\$ *1212*
\% 01:59, 01:89, 01:91, 01:111, 02:14, 02:529, 06:801, 21:491, 21:493, 24:430, 33:640, 33:760, 37:210, 50:1566, 50:1567, 57:504
\& 01:58, 02:5, 02:13, 02:530, 06:800, 50:439, 57:505
\' 02:551, 21:227, 21:379, 21:421, 21:455, 21:465, 21:581, 21:591, 21:597, 21:599, 21:602, 21:604, 21:612, 21:618, 21:624, 21:626, 21:629, 21:631, 21:639, 21:643, 21:650, 21:654, 21:659, 21:664, 21:667, 21:669, 21:676, 21:681, 21:682, 21:689, 21:694, 21:697, 21:747, 21:794, 21:813, 21:815, 21:816, 21:817, 21:820, 21:822, 21:823, 21:824, 21:826, 21:827, 21:1238, 21:1259, 21:1266, 21:1270, 21:1275, 21:1280, 21:1283, 21:1285, 21:1292, 21:1297, 21:1298, 21:1305, 21:1310, 21:1313, 21:1321, 21:1322, 21:1369, 21:1370, 21:1375, 21:1376, 21:1387, 21:1388, 21:1393, 21:1394, 21:1422, 21:1423, 21:1437, 21:1438, 21:1439, 21:1440, 21:1453, 21:1454, 24:440, 29:775, 30:260, 33:161, 37:738, 38:254, 40:387, 40:408, 41:72, 54:588, 57:520
\(. 07:2364, 38:345
\(. 38:271
\) 02:551, 07:2386, 38:346
\) 38:271
* 24:433, 50:1247, 50:1378, 50:1492, 50:1568
* 38:251
\+ 41:72
\+ *1210*
\, ... 02:432, 02:434, 30:513, 37:738, 38:7, 38:8, 38:40, 38:167, 38:169, 38:172, 38:197, 38:220, 38:221, 38:237
\, ... *1251*
\- 02:378, 06:24, 18:463, 18:554, 21:416, 21:417, 21:574, 21:788, 21:789, 24:435, 37:738, 40:386, 40:407, 41:72, 57:224, 57:264
\- 06:835, *110*
\. 02:431, 02:433, 20:45, 20:115, 20:173, 21:228, 21:380, 21:452, 21:453, 21:474, 21:587, 21:588, 21:614, 21:615, 21:641, 21:748, 21:818, 21:825, 21:1243, 21:1325, 21:1326, 21:1335, 21:1336, 21:1345, 21:1346, 21:1363, 21:1424, 21:1425, 21:1461, 21:1462, 24:434, 33:173, 33:199
\. *1239*
\..default *552*
.code 51:4
.if 51:4
.ifnot 51:4
.pass-to-packages 51:22
.store 51:4
.usage 51:4
\/ 01:81, 06:25, 09:202, 09:277, 24:382, 24:436, 50:438
\/ *1206*
\: 38:214, 38:224, 38:225, 38:226, 38:242, 38:252, 38:252, 06:795, 06:796, 02:432, 02:434
\; 02:432, 02:434, 30:507, 38:198
\; 38:214, *791*
\< 21:575, 21:739, 24:431, 37:738, 41:71, 41:109
\= 21:229, 21:381, 21:473, 21:749, 21:1241, 21:1315, 21:1316, 21:1331, 21:1332, 21:1354, 21:1355, 21:1356, 21:1381, 21:1382, 21:1407, 21:1408, 21:1441, 21:1442, 21:1443, 21:1444, 21:1459, 21:1460, 21:1467, 21:1468, 29:775, 33:179, 40:387, 40:408, 41:71

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\> 21:572, 21:740, 24:432, 37:738, 38:226, 38:241, 38:242, 38:252, 41:71
\? 02:431, 02:433, 57:520
@@ commands:
  \g_@@_⟨*hook*⟩_code_prop *1270*
  \@@_cmd_type_cases:NnnnnTF . . . *1261*
  \c_@@_hash_tl *1264*
  \c_@@_hashes_tl *1264*
  \@@_if_file_hook:wTF *1260*
  \@@_include_legacy_code_chunk:n *1266*
  \g_@@_input_file_seq *1257*
  \@@_print_args:nn *1267*
  \@@_replacing_args_false: *1266*
  \@@_set_curr_file:nNN 52:386
  \@@_set_curr_file_assign:nnnNN 52:386
  \g_@@_standard_everypar_tl . . . *1269*
  \@@_start_expandable:nNNNNn . . *1264*
\@@\textvisiblespace␣\meta␣{hook} . *225*
\@@_next\textvisiblespace␣\meta␣{hook} *225*
\@@_toplevel\textvisiblespace␣\meta␣{hook} *225*
\@@par *399*
\@TeXversion *6*, *1*
@botlist commands:
  \@botlist: . . 54:1033, 54:1127, 54:1286
@botnum commands:
  \@botnum: 54:1007
\@car *82*
\@cdr *82*
@citeb commands:
  \@citeb: 47:36, 47:65, 47:82
@colht commands:
  \@colht: 54:545, 54:559
@colnum commands:
  \@colnum: 54:1192, 54:1360
\@cons *82*
\@currdir *5*, *1*
@currname commands:
  \@currname: 20:761
\@dblarg *81*
@dbldeferlist commands:
  \@dbldeferlist: 54:1759, 54:1801
@dbltopnum commands:
  \@dbltopnum: 54:1658, 54:1785
@deferlist commands:
  \@deferlist: 54:1118, 54:1215, 54:1276, 54:1384, 54:1428, 54:1457, 54:1515, 54:1547, 54:1633, 54:1673
\@ifclasslater *977*
\@ifclassloaded *977*
\@ifclasswith *977*
\@ifdefinable *81*
\@ifnextchar *81*
\@ifpackagelater *977*
\@ifpackageloaded *977*
\@ifpackagewith *977*
\@ifstar *81*
\@ifundefined *81*
\@missingfileerror *978*
\@namedef *81*
\@nameuse *81*
\@restorepar *399*
\@setpar *399*
@topnum commands:
  \@topnum: 54:1053
\[. 24:437, 29:43, 29:54, 29:76, 29:87, 38:347, 57:521
\[. 38:289, 38:462, *1241*
\\ 01:58, 01:231, 01:232, 01:233, 01:234, 01:237, 01:244, 01:245, 01:246, 01:247, 01:250, 01:257, 01:258, 01:259, 01:260, 01:263, 01:270, 01:276, 01:277, 01:281, 01:283, 01:284, 01:288, 01:293, 01:294, 01:297, 01:303, 02:13, 02:377, 04:272, 04:424, 06:231, 06:288, 06:449, 06:572, 06:590, 06:800, 07:468, 07:532, 07:563, 07:704, 07:716, 07:747, 07:1118, 07:1124, 07:2547, 07:2575, 07:2589, 07:2596, 07:2625, 07:2722, 07:2750, 07:2761, 07:2787, 07:2793, 07:2800, 08:1777, 08:1788, 08:2580, 08:2594, 08:2595, 08:2596, 08:2597, 08:2598, 08:2603, 08:2606, 08:2607, 08:2608, 08:2618, 08:2619, 08:2633, 08:2654, 08:2658, 08:2659, 08:2661, 08:2667, 08:2678, 08:2679, 08:2684, 08:2689, 08:2694, 08:2710, 08:2875, 08:2876, 08:2877, 09:443, 09:581, 11:965, 11:973, 11:979, 11:980, 11:981, 11:982, 11:995, 11:1001, 11:1002, 11:1008, 11:1014, 11:1015, 11:1021, 11:1022, 11:1029, 11:1036, 11:1043, 11:1050, 11:1051, 11:1060, 11:1078, 11:1079, 11:1109, 11:1110, 11:1111, 11:1112, 11:1113, 11:1114, 11:1115, 11:1116, 11:1117, 11:1118, 11:1119, 14:279, 16:146, 16:153, 16:160, 16:161, 16:162, 16:163, 18:556, 20:754, 20:773, 21:560, 24:425, 30:258, 36:248, 37:451, 37:456, 37:461, 37:471, 37:475, 37:479, 37:520, 38:370, 38:523, 40:419, 40:578, 40:580, 41:73, 41:170, 41:180, 41:194, 42:139, 57:506, *421*
\\ 18:53, *1223*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\{ 01:6, 01:10, 01:58, 02:2, 02:13, 07:635, 07:1918, 09:444, 14:22, 21:308, 21:562, 24:426, 30:256, 37:519, 38:59, 38:167, 57:509
\} 01:11, 01:58, 02:3, 02:13, 09:445, 14:21, 21:309, 21:563, 24:427, 30:257, 37:519, 38:59, 57:510
\⟨*addto-cmd*⟩ *196*
\⟨*cmd*⟩␣ *146*
\⟨*cmd*⟩␣(arg␣⟨*num*⟩) *148*
\⟨*cmd*⟩␣␣ *146*
\⟨*cmd*⟩␣code *146*
\⟨*cmd*⟩␣defaults *146*
\⟨*filename*⟩ *1051*
\⟨*function*⟩ *168*
\␣ 01:58, 01:75, 02:13, 02:381, 02:437, 02:466, 02:467, 02:484, 06:800, 08:2659, 11:1002, 11:1015, 11:1022, 11:1030, 11:1051, 11:1079, 14:19, 14:20, 14:21, 14:22, 14:25, 18:468, 24:422, 24:672, 24:709, 24:734, 30:259, 37:503, 37:504, 37:513, 37:514, 37:659, 37:675, 37:689, 43:53, 43:55, 43:60, 43:62, 47:37, 50:432, 57:501
\] 02:551, 24:438, 38:348, 57:522
\] 38:289, 38:486
\^ . . 01:47, 01:56, 01:59, 01:103, 01:314, 02:7, 02:9, 02:11, 02:14, 02:380, 02:437, 02:438, 02:457, 02:458, 02:479, 02:480, 06:20, 06:801, 07:6, 07:7, 07:1857, 07:2232, 07:2237, 07:3015, 18:556, 18:558, 18:560, 21:231, 21:286, 21:382, 21:456, 21:466, 21:558, 21:644, 21:651, 21:655, 21:660, 21:665, 21:670, 21:677, 21:683, 21:684, 21:690, 21:695, 21:750, 21:1239, 21:1256, 21:1260, 21:1267, 21:1271, 21:1276, 21:1281, 21:1286, 21:1293, 21:1299, 21:1300, 21:1306, 21:1311, 21:1323, 21:1324, 21:1341, 21:1342, 21:1349, 21:1350, 21:1364, 21:1365, 21:1366, 21:1395, 21:1396, 21:1417, 21:1418, 21:1419, 21:1420, 24:423, 24:424, 24:429, 33:169, 33:197, 37:202, 37:211, 37:505, 37:506, 37:660, 50:1248, 50:1249, 50:1250, 50:1331, 50:1334, 50:1337, 50:1379, 50:1380, 50:1381, 50:1463, 50:1466, 50:1469, 50:1493, 50:1494, 50:1495, 50:1553, 50:1556, 50:1559, 57:255, 57:256, 57:257, 57:258, 57:259, 57:260, 57:261, 57:262, 57:263, 57:507, 57:513, 57:514, 57:515, 57:516, 57:550, 57:551, 57:552, 57:553, 57:554, 57:555, 57:556, 57:557, 57:558
_ 21:314, 30:261, 38:269, 38:270, 02:8, 02:14, 06:801, 57:508, 01:59, *1219*
\‘ 21:232, 21:383, 21:420, 21:454, 21:464, 21:580, 21:642, 21:649, 21:653, 21:658, 21:663, 21:668, 21:675, 21:679, 21:680, 21:688, 21:693, 21:751, 21:793, 21:1237, 21:1258, 21:1265, 21:1269, 21:1274, 21:1279, 21:1284, 21:1291, 21:1295, 21:1296, 21:1304, 21:1309, 24:439, 29:775, 33:175, 37:738, 40:387, 40:408, 41:72, 57:523
\| 21:561, 22:217, 22:228, 30:580, 30:581, 57:524
\~ 01:59, 02:10, 02:14, 06:801, 14:20, 18:469, 21:239, 21:287, 21:384, 21:467, 21:559, 21:645, 21:657, 21:661, 21:671, 21:687, 21:691, 21:752, 21:1240, 21:1257, 21:1261, 21:1273, 21:1277, 21:1287, 21:1303, 21:1307, 21:1351, 21:1352, 21:1353, 21:1405, 21:1406, 33:187, 33:207, 37:653, 37:670, 37:684, 37:699, 37:742, 57:511

A

\A 57:252, 57:526, 57:547
\a 21:223, 41:1, 57:243, 57:527, 57:538, *1222*
\AA . . . 21:240, 21:428, 21:526, 02:443, *1234*
\aa . . . 21:245, 21:422, 21:536, 02:443, *1234*
\abovedisplayshortskip . . 38:531, 02:413
\abovedisplayskip 38:524, 38:526, 38:528, 38:529, 38:530, 38:531, 02:412, *1220*
\abspage *750*
abspage 36:232, *750*
\accent 21:73, 21:393, 21:423, 21:479, 21:763, *1250*
\ActivateGenericHook 08:2734, 08:2736, 08:2741, 08:2743, *213*
\active 01:103, 37:503, 37:504, 37:505, 37:506, 37:513, 37:514, 37:652, 37:659, 37:660, 37:669, 37:675, 37:683, 37:689, 37:698, 37:740, 38:254, 38:269, 50:1248, 50:1249, 50:1250, 50:1331, 50:1334, 50:1337, 50:1379, 50:1380, 50:1381, 50:1463, 50:1466, 50:1469, 50:1493, 50:1494, 50:1495, 50:1553, 50:1556, 50:1559, 01:314, 02:10,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

02:11, 54:588, 01:48, 02:457, 02:458, 02:466, 02:467, 02:479, 02:480, 02:484
\acute 30:523
add commands:
add_to_callback 04:817
\add_to_callback *45*
\addcontentsline 44:70, 44:80, 44:159, 45:16, *1219*
\AddEverypageHook *1066*
\Adding *1195*
\addpenalty 18:285, 39:124, 39:194, 39:199, 44:50, 54:340, 54:1157, 54:1323, *433*
\AddThispageHook *1066*
\addtocontents 44:164, 44:171, 44:177, 44:180, *1219*
\addtocounter *512*
\addtocounter 22:6, 22:18, *1207*
\AddToHook 08:2746, 08:2896, 26:152, 37:42, 37:43, 37:44, 37:107, 37:108, 37:109, 37:396, 37:397, 37:398, 37:399, 47:72, 50:1128, 50:1129, 50:1130, 52:561, 52:563, 52:573, 52:574, 52:575, 52:576, 53:190, 53:473, 53:496, 55:7, *305*
\AddToHookNext 08:2757, 08:2894, 52:562, 53:497, *199*
\AddToHookNextWithArguments 08:2757, *199*
\AddToHookWithArguments . . . 08:2746, *197*
\addtolength *521*
\addtolength 23:16, 38:526, 38:528
\AddToNoCaseChangeList 57:571, *1262*
\addtoversion 27:20, 27:139
\addvspace 18:234, 37:429, 39:124, 39:195, 39:196, 39:200, 39:248, 44:50, *1206*
\adjdemerits 02:336
\AE 21:241, 21:398, 21:527, 21:768, 21:1127, 21:1437, 21:1441, 57:649
\ae 21:246, 21:401, 21:537, 21:772, 21:1133, 21:1439, 21:1443, 57:649
\afterassignment 21:212, 21:220, 24:328, 38:199, 06:268, 06:274, 06:317, 02:501, 02:504, *93*
\AfterEndEnvironment 37:392, *218*
\aftergroup 24:91, 24:347, 24:358, 24:362, 26:202, 26:268, 28:114, 28:121, 28:129, 28:404, 32:64, 37:656, 37:673, 37:687, 37:702, 38:445, 38:446, 39:149, 40:187, 54:608, 54:609, 54:666, 54:667, *811*
\AfterLastShipout 52:573
\afterpreamble *446*
\aleph 30:315
\allocationnumber 16:81, 04:52, 04:53, 04:54, 04:91, 04:213, 41:4, 41:9, 53:34, 02:37, 57:58, 57:59, 57:60, 02:57, 02:69, 02:71, 02:143, 02:144, 02:145, 02:195, 02:196, 02:237, 02:238, 02:239, 02:252, 02:253, 02:254, 02:271, 02:277, 02:283, 02:284, 02:297, 02:298, 02:299, *412*
\allowbreak 38:40, 06:877, 06:878, 06:897, 06:899, 02:508
\Alph *512*
\Alph 22:188, *1064*
\alph *512*
\alph 22:187
\alpha 30:275
\amalg 30:386
\AmSfont *561*
\and *898*
\and 44:14, 44:27
\angle 30:344
\approx 30:429
\arabic *512*
\arabic 22:127, 22:135, 22:184, 43:46, 53:350, *512*
\arccos 38:13
\arcsin 38:10
\arctan 38:16
\arg 38:26
array (env.) 41:168
\array 41:168
\arraycolsep 38:373, 38:374, 38:536, 38:537, 41:258, 41:338
\arrayrulewidth 41:324, 41:338, 41:346, 41:347, 41:359, 41:363, 41:366, 41:376, 41:378
\arraystretch 41:186, 41:187, 41:342
\Arrowvert 30:576
\arrowvert 30:574
\asciispace 37:576, 37:578, 37:581, 37:595, 37:606, 37:607, 37:620, 37:621
\AssignSocketPlug 10:170, 10:187, 35:121, 55:48, 55:61, *334*
\AssignTemplateKeys . 11:387, 11:1193, *374*
\ast 30:239, 30:402
\asymp 30:456
\AtBeginDocument 20:126, 20:181, 25:737, 47:54, 50:1119, *212*
\atbegindocumenthook *446*
\AtBeginDvi . . . 53:412, 53:448, 54:86, *1066*
\AtBeginEnvironment 37:392, *218*
\AtBeginShipout 53:453, 53:496, *1261*
\AtBeginShipoutAddToBox . . . 53:501, *1065*
\AtBeginShipoutAddToBoxForeground 53:501, *1065*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\AtBeginShipoutBox 53:494, *1065*
\AtBeginShipoutDiscard 53:500, *1066*
\AtBeginShipoutFirst 53:456, 53:498, *1066*
\AtBeginShipoutInit 53:495, *1066*
\AtBeginShipoutNext . 53:454, 53:496, *1066*
\AtBeginShipoutOriginalShipout 53:509, *1065*
\AtBeginShipoutUpperLeft . . . 53:501, *1065*
\AtBeginShipoutUpperLeftForeground 53:501, *1065*
\AtEndAfterFileList 52:575
\AtEndDocument 37:144, 50:1119, 52:579, *1238*
\AtEndDvi 53:463, 53:468, *1261*
\AtEndEnvironment 37:392, *218*
\AtEndOfClass 38:461, 50:1119, *1038*
\AtEndOfPackage 50:606, 50:625, 50:723, 50:734, 50:1119, 51:85, *1038*
\AtEndPreamble *219*
\AtNextShipout *1066*
\atopwithdelims 38:57, 38:58, 38:59
\attribute 04:79, *42*
\attributedef 04:79, 04:223
\attributezero 04:223
\AtVeryEndDocument 52:574
\AtVeryVeryEnd 52:576
\author *898*
\author 44:8, 44:24, 44:32

B

\b 21:233, 21:389, 21:475, 21:759, 21:1248, *1222*
\backslash 30:258, 30:597
\bar 30:527
\baselineskip 26:186, 26:187, 26:188, 26:190, 26:191, 30:517, 38:171, 38:172, 38:191, 38:197, 38:201, 40:396, 40:415, 40:601, 41:198, 42:136, 42:314, 42:373, 53:222, 53:273, 54:244, 54:275, 54:626, 54:641, 54:685, 54:700, 02:409, 02:436, 02:499, 02:535, *1209*
\baselinestretch 24:319, 26:120, 26:121, 26:166, 26:167, 26:184, 26:245, *1193*
\baselinstretch *1213*
\batchmode 20:691, 20:722, 20:723, 27:106, 05:89, 05:95, 05:105, 57:667, 57:688, *462*
\BCPdata 57:560, 57:583, 57:587
\BeforeBeginEnvironment 37:392, *218*
\BeforeClearDocument 52:577
\begin . . . 09:66, 14:246, 14:248, 20:345, 20:397, 26:7, 30:4, 31:4, 37:242, 37:243, 37:313, 37:335, 37:360, 37:376, 37:384, 38:466, 38:478, 44:14, 44:17, 50:744, 51:253, 53:394, 06:709, 56:3, 07:1426, 07:1536, *770*
\begingroup *444*
\belowdisplayshortskip . . 38:530, 02:415
\belowdisplayskip 38:529, 02:414
\beta 30:276
\bezier *862*
\bezier 42:681, 42:682, 42:804, 42:805, 42:820, 42:822
\bf *1201*
\bfdefault 29:15, 29:261, 29:267, 29:268, 29:269, 29:270, 29:310, 29:311, 29:312, 29:313, 29:322, 29:358, 29:382, 29:401, 29:442, 29:476, 30:100, 30:111, 30:113, 30:121, *643*
\bfseries 29:13, 29:14, 29:253, 29:254, 29:302, 29:307, 29:308, 29:349, 29:352, 29:353, 29:378, 29:380, 29:381, 29:474, 29:475, 32:19, 35:17, 35:34, 35:51, 43:53, 43:55, 43:60, 43:62, 47:40, 53:395, *635*
bfseries 29:420
bfseries/defaults 29:420
\bgroup 02:450, *1206*
\bibcite 47:7, 47:9, 47:10, *1226*
\bibdata 47:45, 47:49
\bibitem 47:3
\bibliography *937*
\bibliography 47:47
\bibliographystyle *937*
\bibliographystyle 47:52
\bibstyle 47:45, 47:57
\Big 30:630, 30:633, 30:642, 30:644, 38:44, 38:45, 38:46
\big 30:631, 30:643, 38:41
\bigbreak 06:879, 06:900, 02:515
\bigcap 30:352
\bigcirc 30:399
\bigcup 30:353
\Bigg . . . 30:637, 30:646, 38:50, 38:51, 38:52
\bigg . . . 30:635, 30:645, 38:47, 38:48, 38:49
\Biggl 38:50
\biggl 38:47
\Biggm 38:51
\biggm 38:48
\Biggr 38:52
\biggr 38:49
\Bigl 38:44
\bigl 38:41
\Bigm 38:45
\bigm 38:42
\bigodot 30:360
\bigoplus 30:359
\bigotimes 30:358

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\Bigr 38:46
\bigr 38:43
\bigskip 18:447, 02:520
\bigskipamount 18:449, 18:450, 45:391, 02:519
\bigsqcup 30:363
\bigtriangledown 30:368, 30:369
\bigtriangleup 30:367, 30:370
\biguplus 30:351
\bigvee 30:349
\bigwedge 30:350
\binoppenalty 02:323
\bmod 38:35
\boldmath 19:14, 29:613
bool commands:
\bool_gset_false:N 10:16, 36:31, 48:313, 53:15, 53:81, 53:90, 08:15
\bool_gset_true:N . . . 10:11, 36:33, 48:308, 53:10, 53:120, 53:344, 08:10
\bool_if:NTF 09:162, 09:166, 09:178, 09:237, 09:241, 09:253, 09:380, 10:22, 10:24, 11:222, 11:228, 11:658, 11:761, 11:810, 11:827, 11:849, 36:71, 48:319, 51:24, 51:216, 53:21, 53:88, 53:365, 07:121, 07:130, 07:142, 07:143, 07:157, 07:169, 07:173, 07:182, 07:184, 07:194, 07:203, 07:204, 07:207, 07:212, 07:214, 07:318, 07:463, 07:472, 07:474, 07:529, 07:560, 07:575, 07:624, 07:657, 07:701, 07:711, 07:744, 07:752, 07:757, 07:758, 07:766, 07:771, 07:794, 07:807, 07:885, 07:931, 07:932, 07:991, 07:1004, 07:1021, 07:1041, 07:1954, 07:2207, 07:2539, 07:2550, 08:21, 08:340, 08:1913, 08:1922, 08:2013, 08:2022, *191*
\bool_if_exist:NTF 36:27
\bool_lazy_and:nnTF 08:2449, 08:2917, 53:66, 53:112, 53:376, 07:740, 07:933, 08:275, 08:1915, 08:2015
\bool_lazy_and_p:nn . . 08:2452, 07:112
\bool_lazy_any:nTF . . . 07:108, 07:483
\bool_lazy_or:nnTF 07:73, 07:1385, 07:1394, 07:2362, 07:2384, 08:886, 08:1866
\bool_new:N 10:6, 11:21, 11:22, 36:28, 48:304, 51:57, 53:6, 53:188, 53:203, 07:16, 07:18, 07:20, 07:31, 07:32, 07:34, 07:35, 07:37, 07:40, 07:43, 07:44, 07:45, 08:6, 08:24
\bool_set_false:N 11:277, 11:656, 11:737, 51:84, 55:44, 07:57, 07:83, 07:232, 07:254, 07:299, 07:449, 07:450, 07:451, 07:452, 07:453, 07:454, 07:475, 07:537, 07:567, 07:592, 07:606, 07:628, 07:629, 07:630, 07:645, 07:707, 07:750, 07:761, 07:762, 07:783, 07:784, 07:785, 07:787, 07:791, 07:798, 07:940, 07:941, 07:942, 07:1844, 08:1905, 08:2005
\bool_set_true:N 11:248, 11:290, 11:615, 11:746, 51:81, 55:46, 07:62, 07:233, 07:255, 07:288, 07:448, 07:473, 07:544, 07:551, 07:552, 07:566, 07:753, 07:754, 07:814, 07:815, 07:821, 07:822, 07:828, 07:829, 07:836, 07:837, 07:838, 07:958, 07:976, 07:1849, 08:1902, 08:2002
\bool_while_do:nn . . . 08:1592, 08:1677
\c_false_bool 09:141, 09:151, 09:370, 07:2208, 07:2511, 07:2802, 07:2970, 08:397, 08:403, 08:411, *191*
\l_tmpa_bool *346*
\c_true_bool 09:146, 07:2209, 07:2513, 07:2803, 07:2968, 08:404, 08:412, 08:414, *313*
bool internal commands:
\g__mark_debug_bool 48:304, 48:308, 48:313, 48:319
\BooleanFalse . . 07:1837, 07:2201, 07:2802
\BooleanTrue . . . 07:1836, 07:2200, 07:2802
\bordermatrix 38:185
\bot 30:328
\botfigrule 54:746, 54:2363
\botmark 49:79, 54:651, 54:710, *942*
\bottomfraction 45:275, 54:2332
\bowtie 30:489
\Box 29:726
\box *404*
box commands:
\box_dp:N 53:194
\box_gclear:N 16:89
\box_gset_to_last:N 16:16, 16:49, 16:117
\box_ht:N 53:193, 53:302, *1070*
\box_if_empty:NTF . 48:86, 53:78, 53:97
\box_if_horizontal:NTF 53:234, 53:286
\box_if_vertical:NTF 48:81, 48:382, 48:413, 53:208, 53:259
\box_move_up:nn 53:248, 53:302
\box_new:N 16:84, 48:61, 48:62, 53:23, 53:25, 53:187, 53:204
\box_set_dp:Nn . . . 53:221, 53:230, 53:247, 53:272, 53:283, 53:301, 53:332
\box_set_eq:NN . 53:149, 53:179, 53:184

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\box_set_eq_drop:NN 53:93
\box_set_ht:Nn ... 53:220, 53:229, 53:246, 53:271, 53:282, 53:300, 53:331
\box_set_to_last:N 48:78
\box_set_wd:Nn 53:219, 53:245, 53:270, 53:299
\box_use:N 53:126, 53:225, 53:250, 53:278, 53:303, 53:333
\box_use_drop:N 16:86
\box_wd:N 53:195, 53:296, 53:304
box internal commands:
\l__mark_box .. 48:61, 48:74, 48:78, 48:81, 48:82, 48:85, 48:86, 48:96, *954*
\l__mark_ii_box .. 48:61, 48:85, 48:95
\boxmaxdepth ... 42:474, 42:502, 42:532, 42:610, 42:627, 54:490, 54:510, 54:550, 54:719, 54:728, 54:768, 02:395
\brace 38:59
\braceld 30:560, 30:564, 30:565, 30:567, 30:569
\bracelu 30:562, 30:566, 30:568
\bracerd 30:561, 30:566, 30:568
\braceru 30:563, 30:565, 30:569
\bracevert 30:615
\brack 38:58
\break 18:116, 06:880, 06:901, 02:508, 02:513, *1227*
\breve 30:528
\brokenpenalty 24:692, 02:328
\buildrel 30:476, 38:162
\bullet 30:388

C

\c 11:383, 21:234, 21:335, 21:337, 21:339, 21:341, 21:343, 21:345, 21:347, 21:349, 21:351, 21:372, 21:374, 21:392, 21:459, 21:478, 21:606, 21:608, 21:633, 21:635, 21:648, 21:674, 21:701, 21:704, 21:705, 21:706, 21:707, 21:708, 21:709, 21:710, 21:711, 21:712, 21:762, 21:1250, 21:1264, 21:1290, 21:1347, 21:1348, 21:1367, 21:1368, 21:1371, 21:1372, 21:1377, 21:1378, 21:1389, 21:1390, 21:1397, 21:1398, 21:1401, 21:1402, 33:167, 33:196, *1243*
\cal 29:776
call commands:
call_callback 04:798
\call_callback *46*
callback commands:
callback_descriptions 04:983
callback.register 04:686
\callback_descriptions *46*
\cap 30:379
\capitalacute 21:839, 33:160, 33:161, 33:193, 33:690, 33:940, 33:1306
\capitalbreve 21:846, 33:162, 33:163, 33:194, 33:691, 33:947, 33:1313
\capitalcaron 21:845, 33:164, 33:165, 33:195, 33:692, 33:946, 33:1312
\capitalcedilla 21:832, 33:166, 33:167, 33:196, 33:693, 33:937, 33:1316
\capitalcircumflex ... 21:840, 33:168, 33:169, 33:197, 33:694, 33:941, 33:1307
\capitaldieresis 21:842, 33:170, 33:171, 33:198, 33:695, 33:943, 33:1309
\capitaldotaccent ... 21:848, 33:172, 33:173, 33:199, 33:696, 33:949, 33:1315
\capitalgrave 21:838, 33:174, 33:175, 33:200, 33:697, 33:939, 33:1305
\capitalhungarumlaut . 21:843, 33:176, 33:177, 33:201, 33:698, 33:944, 33:1310
\capitalmacron 21:847, 33:178, 33:179, 33:202, 33:699, 33:948, 33:1314
\capitalnewtie 21:852, 33:190, 33:191, 33:203, 33:700, 33:1014, 33:1015, 33:1321, *701*
\capitalogonek 21:835, 33:180, 33:181, 33:204, 33:701, 33:938, 33:1317
\capitalring 21:844, 33:182, 33:183, 33:205, 33:702, 33:945, 33:1311
\capitaltie 21:850, 33:184, 33:185, 33:206, 33:703, 33:1010, 33:1011, 33:1319
\capitaltilde 21:841, 33:186, 33:187, 33:207, 33:704, 33:942, 33:1308
\caption 45:4, *304*
\cases 38:166, 38:167, 38:177, 38:179
\CaseSwitch 57:571, *1262*
\catcode *767*
\catcodetable 04:89, 04:109, *42*
\catcodetable@atletter *43*
\catcodetable@initex *43*
\catcodetable@latex *43*
\catcodetable@string *43*
\catcoding *1197*
\cdot 30:401
\cdotp 30:509, 30:515
\cdots 30:515
center (env.) 37:444
\center 37:444
\centering 37:444, 37:449, 37:450, 37:468, 37:470, 37:485, 37:487
\centerline 40:614
\changes 22:36, 29:667, *76*
\chaptermark *946*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\char . . . 21:391, 21:394, 21:430, 21:433, 21:444, 21:451, 21:477, 21:481, 21:486, 21:489, 21:491, 21:493, 21:733, 21:761, 21:764, 21:797, 21:804, 21:811, 21:834, 21:837, 21:866, 21:896, 21:1006, 21:1041, 21:1165, 21:1167, 21:1169, 21:1216, 29:626, 29:633, 33:631, 33:638, 33:640, 33:760, 37:576, 37:743, 38:251, 42:232, 42:282, 42:296, 42:304, 42:307, 42:448, 42:562, 42:567, 42:575, 42:579, 42:615, 42:616, 42:618, 42:631, 42:632, 42:635, 42:662, 06:842, 06:857, *1230*

char commands:

\char_generate:nn 05:145, 07:1666, 07:1962
\char_set_catcode_active:N . 07:2232
\char_set_catcode_active:n . 07:2409
\char_set_catcode_escape:N . . 09:443
\char_set_catcode_group_begin:N 09:444
\char_set_catcode_group_end:N 09:445
\char_set_catcode_other:N . . 07:1952
\char_set_catcode_other:n . . 07:1955
\char_set_catcode_parameter:N 09:446
\char_set_catcode_parameter:n 07:1956
\char_set_lccode:nn . 07:2237, 07:2419
\char_value_catcode:n 09:375

\chardef 12:2, 04:22, 04:26, 04:38, 04:47, 04:48, 20:52, 20:122, 04:89, 21:18, 04:157, 04:224, 24:14, 41:4, 41:9, 50:1254, 50:1385, 50:1502, 02:10, 02:16, 02:17, 02:18, 02:19, 02:20, 57:42, 57:44, 57:48, 57:67, 57:171, 57:172, 57:173, 57:174, 57:175, 57:176, 57:177, 02:58, 02:64, 02:66, 02:73, 02:79, 02:82, 02:84, 02:94, 02:96, 02:97, 02:98, 02:99, 02:108, 02:114, 02:115, 02:128, 02:130, 02:194, 01:48, 02:253, 02:257, 02:259, 01:54, 02:283, 01:55, 02:298, 02:529, 02:530, 02:531, *1210*

\charsubdef 57:353
\charzero 04:224
\check 30:529
\CheckCommand 06:187, *1214*
\CheckEncodingSubset 33:16, 33:61, 33:109, 33:110, 33:111, 33:156, 33:158, 33:352, 33:621, 33:830, 33:879, 33:935, 33:936, 33:1004, 33:1121, 33:1124, 33:1138, *695*
\chi 30:295
\choose 38:57
\circ 30:398
\circle 42:450, 42:604, 42:806, 42:823, *1197*
\citation 47:11, 47:39, 47:67, 47:84
\cite *937*
\cite 47:12, *938*
\clap 40:618, *1251*
class/.../after *1037*
class/.../before *1037*
class/after *1037*
class/before *1037*
\ClassError 14:84
\ClassInfo 14:84
\ClassNote 14:136, *1258*
\ClassNoteNoLine 14:136
\ClassWarning 14:84
\ClassWarningNoLine 14:84
\cleaders 30:555, 30:558, 02:549
\cleardoublepage 54:140
\ClearHookNext 08:2768, *199*
\ClearHookRule 08:2805, 08:2910, *203*
\clearpage 20:297, 20:324, 20:328, 20:359, 20:382, 20:385, 20:406, 20:424, 37:17, 37:83, 37:147, 37:240, 54:127, 54:140, 54:145, 54:202, 54:409, 54:412, 54:416, 54:457, 54:463, 54:2209, 54:2226, *1039*
\cline 41:367, *1269*

clist commands:

\clist_clear:N 51:65
\clist_gclear:N 08:1591, 08:1676
\clist_gput_left:Nn . 08:1495, 08:1530
\clist_gput_right:Nn 08:1497, 08:1534
\clist_if_empty:NTF 11:534, 11:563, 08:1932, 08:2033
\clist_if_exist:NTF 08:121
\clist_if_in:NnTF . 11:266, 11:570, 51:51, 51:52, 51:127, 51:150, 51:152
\clist_map_function:nN 36:59
\clist_map_inline:Nn 11:536, 51:108, 51:139, 51:204
\clist_map_inline:nn 51:239, 08:917, 08:926
\clist_map_variable:NNn 51:82
\clist_new:N 11:32, 51:54, 51:55, 51:56, 08:123, 08:140
\clist_put_right:Nn . 51:33, 51:53, 51:121, 51:128, 51:135, 51:160, 51:169
\clist_remove_all:Nn 11:572, 51:35, 51:134, 51:161
\clist_set:Nn 11:524
\clist_use:Nn 08:1934, 08:2035
\l_tmpa_clist *346*

\clubpenalty 20:8, 20:25, 20:95, 20:152, 24:690,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

39:128, 39:218, 39:220, 44:100, 44:106, 44:130, 44:135, 02:325, *1233*
\clubsuit 30:338
cmd internal commands:
__cmd_add_arg:n 07:1602, 07:1613, 07:1618, 07:1619, 07:1653, 07:1741, 07:1748, 07:1823, 07:1836, 07:1837, 07:1911, 07:1968, 07:1975, 07:2007, 07:2007, 07:2012, *156*
__cmd_add_arg_spec:n . . . 07:590, 07:604, 07:664, 07:738, 07:738, 07:778
__cmd_add_arg_spec_mandatory:n . 07:636, 07:646, 07:652, 07:738, 07:764
__cmd_add_default: 07:846, 07:884, 07:901, 07:952, 07:955, 07:962, 07:972, 07:1039, 07:1048, *136*
__cmd_add_default:n 07:853, 07:893, 07:909, 07:952, 07:952, 07:969, 07:983, *136*
__cmd_add_default_E:nn 07:861, 07:952, 07:967, 07:1017
__cmd_add_expandable_grabber:nn 07:996, 07:1009, 07:1020, 07:1040, 07:1050, 07:1057, 07:1057
__cmd_add_expandable_type_+:w 07:974
__cmd_add_expandable_type_D:w 07:979, 07:979
__cmd_add_expandable_type_D_aux:NN 07:979, 07:985, 07:1002
__cmd_add_expandable_type_D_aux:NNN 07:979, 07:986, 07:989
__cmd_add_expandable_type_D_aux:NNNn 07:979, 07:980, 07:981, 07:1045
__cmd_add_expandable_type_E:w 07:1015, 07:1015
__cmd_add_expandable_type_E_aux:n 07:1015, 07:1019, 07:1031
__cmd_add_expandable_type_m:w 07:1037, 07:1037
__cmd_add_expandable_type_R:w 07:1044, 07:1044
__cmd_add_expandable_type_t:w 07:1046, 07:1046
__cmd_add_grabber:N 07:847, 07:854, 07:867, 07:886, 07:894, 07:902, 07:910, 07:924, 07:924, *138*
__cmd_add_type_!:w 07:818
__cmd_add_type_+:w 07:811
__cmd_add_type_=: 07:833
__cmd_add_type_>:w 07:825
__cmd_add_type_b:w . . 07:843, 07:843
__cmd_add_type_D:w . . 07:850, 07:850
__cmd_add_type_E:w . . 07:858, 07:858
__cmd_add_type_m:w . . 07:882, 07:882
__cmd_add_type_R:w . . 07:890, 07:890
__cmd_add_type_t:w . . 07:898, 07:898
__cmd_add_type_v:w . . 07:906, 07:906
__cmd_all_m_check:n 07:96, 07:96, 07:116
__cmd_all_m_check_aux:n 07:96, 07:97, 07:98
__cmd_allowed_token_check:N 07:588, 07:600, 07:621, 07:642, 07:680, 07:680
\l__cmd_arg_spec_tl 07:11, 07:90, 07:447, 07:535, 07:568, 07:622, 07:755, *113*
__cmd_arg_to_keyvalue:nn 07:840, 07:2291, 07:2291
__cmd_arg_to_keyvalue_auxi:nnn 07:2291, 07:2306, 07:2308
__cmd_arg_to_keyvalue_auxii:Nnnn 07:2291, 07:2311, 07:2314
__cmd_arg_to_keyvalue_auxiii:nnn 07:2291, 07:2317, 07:2320
__cmd_arg_to_keyvalue_auxiv:Nnnn 07:2291, 07:2323, 07:2326
__cmd_arg_to_keyvalue_auxv:nn 07:2291, 07:2312, 07:2318, 07:2324, 07:2330, 07:2332
__cmd_arg_to_keyvalue_braces:nnn 07:2291, 07:2293, 07:2296
__cmd_arg_to_keyvalue_loop:w 07:2291, 07:2334, 07:2337, 07:2349, 07:2351, 07:2366, 07:2387
__cmd_arg_to_keyvalue_loop_group:n . . 07:2291, 07:2343, 07:2348
__cmd_arg_to_keyvalue_loop_N_type:N . . . 07:2291, 07:2340, 07:2352
__cmd_arg_to_keyvalue_loop_space:w . . 07:2291, 07:2344, 07:2350
__cmd_arg_to_keyvalue_math:w 07:2291, 07:2365, 07:2369, 07:2388, 07:2391, 07:2393
__cmd_arg_to_keyvalue_math_group:n . . 07:2291, 07:2375, 07:2390
__cmd_arg_to_keyvalue_math_N_type:N . . . 07:2291, 07:2372, 07:2380
__cmd_arg_to_keyvalue_math_space:w . . 07:2291, 07:2376, 07:2392
__cmd_arg_to_keyvalue_set_default:nn 07:2291, 07:2355, 07:2383, 07:2394, *177*
__cmd_arg_to_keyvalue_set_keyvalue:nn 07:2291, 07:2359, 07:2396, *177*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\l__cmd_args_i_tl .. 07:13, 07:327, 07:333, 07:338, 07:339, 07:341, *113*
\l__cmd_args_ii_tl 07:14, 07:337, 07:339, 07:341, 07:379, 07:384, 07:391, *113*
__cmd_args_process: 07:317, 07:377, 07:377, *137*
__cmd_args_process_aux:n 07:377, 07:390, 07:394
__cmd_args_process_loop:nn 07:377, 07:383, 07:386
\l__cmd_args_tl ... 07:12, 07:280, 07:305, 07:322, 07:327, 07:333, 07:352, 07:381, 07:384, 07:397, 07:398, 07:1538, 07:1539, 07:1544, 07:1547, 07:1719, 07:1753, 07:2009, *125*
__cmd_bad_arg_spec:wn ... 07:505, 07:510, 07:519, 07:528, 07:574, 07:587, 07:597, 07:598, 07:603, 07:614, 07:620, 07:641, 07:732, 07:732
__cmd_bad_def:wn 07:461, 07:469, 07:498, 07:533, 07:564, 07:579, 07:661, 07:669, 07:677, 07:696, 07:705, 07:717, 07:732, 07:737, 07:748, 07:775, *145*
__cmd_bool_reverse:N 07:2205, 07:2205, 07:3001
__cmd_break_point:n 07:93, 07:95, 07:95, 07:732, 07:737, 07:1107, 07:1114, 07:1325
__cmd_cant_copy:nwn 07:1104, 07:1114, 07:1114, 07:1230, 07:1293, 07:1323
__cmd_check_definable:nNTF 07:2406, 07:2406, 07:2806, 07:2819, 07:2832, 07:2837, 07:2925, 07:2938, 07:2951, 07:2959
__cmd_check_definable_aux:nN ... 07:2406, 07:2407, 07:2410, *180*
__cmd_check_end:n 07:1289, 07:1291, 07:1297
__cmd_check_end:Nn 07:1276, 07:1277, 07:1289, 07:1289, 07:1435
__cmd_check_end:w 07:1289, 07:1299, 07:1302
__cmd_chk_if_free_cs:N 07:3006, 07:3007
__cmd_cmd_if_xparse:NTF *1257*
__cmd_cmd_if_xparse_aux:N . 07:2473
__cmd_cmd_type_cases:NnnnnnTF 07:1098, 07:1317, 07:2473, 07:2473, 07:2496, *145*
__cmd_copy:NN 07:1091, 07:1093, 07:1093, 07:1128, 07:1143, 07:1189, 07:1307, *145*
__cmd_copy_command:nnNN 07:1099, 07:1129, 07:1129, *146*
__cmd_copy_command:NnNNnnnn ... 07:1129, 07:1134, 07:1136, *146*
__cmd_copy_environment:nnNN ... 07:1102, 07:1263, 07:1263
__cmd_copy_environment:Nnnnnnn . 07:1263, 07:1269, 07:1272
__cmd_copy_environment_end:nnNN 07:1103, 07:1274, 07:1274
__cmd_copy_environment_end_-aux:nnNN . 07:1274, 07:1278, 07:1281
__cmd_copy_expandable:nnN 07:1208, 07:1213, 07:1216, 07:1243, 07:1253, 07:1259, 07:1261
__cmd_copy_expandable:nnNN 07:1100, 07:1142, 07:1146, 07:1148, 07:1161, 07:1163, 07:1174
__cmd_copy_expandable:NnNNNNnnn 07:1142, 07:1158, 07:1171, 07:1191, *146*
__cmd_copy_expandable_signature:NnNNNNnnn 07:1154, 07:1169, 07:1208, 07:1208, *146*
__cmd_copy_grabber_⟨*type*⟩:w ... *148*
__cmd_copy_grabber_D:w 07:1233, 07:1233, 07:1246, 07:1247
__cmd_copy_grabber_D_alt:w 07:1233, 07:1245, 07:1248
__cmd_copy_grabber_E:w 07:1233, 07:1249, 07:1255
__cmd_copy_grabber_E_long:w ... 07:1233, 07:1255
__cmd_copy_grabber_m:w 07:1233, 07:1261, 07:1262
__cmd_copy_grabber_m_long:w ... 07:1233, 07:1262
__cmd_copy_grabber_R:w 07:1233, 07:1247
__cmd_copy_grabber_R_alt:w 07:1233, 07:1248
__cmd_copy_grabber_t:w 07:1233, 07:1256
__cmd_copy_optimized:nnNN 07:1101, 07:1177, 07:1177
__cmd_copy_parse_grabber:w 07:1208, 07:1220, 07:1224, *148*
\l__cmd_current_arg_int ... 07:15, 07:25, 07:146, 07:154, 07:183, 07:190, 07:270, 07:351, 07:355, 07:360, 07:361, 07:445, 07:457, 07:480, 07:536, 07:582, 07:607, 07:782, 07:957, 07:964, 07:1211, 07:1219, 07:1237, 07:1241, 07:1242, 07:1252, 07:1444, 07:1510, 07:1519, *149*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

__cmd_declare_cmd:Nnn 07:55, 07:55, 07:2814, 07:2822, 07:2833, 07:2838
__cmd_declare_cmd_aux:Nnn 07:55, 07:58, 07:63, 07:65
__cmd_declare_cmd_code:Nnn 07:91, 07:106, 07:106
__cmd_declare_cmd_code_aux:Nnn 07:106, 07:123, 07:150
__cmd_declare_cmd_code_expandable:Nnn 07:106, 07:122, 07:178
__cmd_declare_cmd_internal:Nnnn 07:55, 07:84, 07:86, 07:263, *122*
__cmd_declare_cmd_optimized:Nnn 07:106, 07:125, 07:128
__cmd_declare_env:nnnn 07:224, 07:224, 07:226, 07:236, 07:238, 07:240, 07:2856, 07:2871, 07:2877, 07:2887, 07:2902, 07:2908, 07:2912, 07:2914, *189*
__cmd_declare_env_internal:nnnn 07:224, 07:234, 07:256, 07:261
__cmd_declare_expandable_-cmd:Nnn 07:55, 07:60, 07:2933, 07:2941, 07:2954, 07:2960
__cmd_defaults: 07:316, 07:324, 07:324
__cmd_defaults_aux: 07:324, 07:328, 07:329, 07:330, 07:335
\l__cmd_defaults_bool 07:16, 07:169, 07:184, 07:207, 07:787, 07:958, *114*
__cmd_defaults_def: 07:324, 07:326, 07:348
__cmd_defaults_def:nn 07:324, 07:353, 07:358
__cmd_defaults_def:nnn 07:324, 07:361, 07:363
__cmd_defaults_error:w 07:324, 07:331, 07:343
\l__cmd_defaults_tl 07:17, 07:170, 07:191, 07:308, 07:316, 07:352, 07:788, 07:959, 07:965, *114*
__cmd_delimiter_check:nnn 07:635, 07:644, 07:721, 07:721
__cmd_end_expandable:NNw 07:404, 07:406, 07:406
__cmd_end_expandable_aux:nNNNN 07:406, 07:409, 07:410
__cmd_end_expandable_aux:w 07:406, 07:407, 07:408
__cmd_end_expandable_defaults:nnnNNn 07:406, 07:413, 07:422, 07:431, 07:441, 07:442, *126*
__cmd_end_expandable_defaults:nnw 07:406, 07:430, 07:434
__cmd_end_expandable_defaults:nw 07:406, 07:437, 07:438, 07:440
\l__cmd_environment_bool 07:18, 07:83, 07:110, 07:157, 07:233, 07:255, 07:288, 07:299, 07:318, 07:474, 07:657, 07:2550, *114*
__cmd_environment_or_command: 07:346, 07:460, 07:492, 07:496, 07:578, 07:660, 07:667, 07:676, 07:691, 07:735, 07:769, 07:1821, 07:1965, 07:1972, 07:2548, 07:2548
\l__cmd_environment_str 07:19, 07:160, 07:228, 07:242, 07:243, 07:244, 07:245, 07:248, 07:252, 07:257, 07:287, 07:290, 07:291, 07:319, 07:2551, *114*
\l__cmd_expandable_aux_name_tl 07:21, 07:22, 07:994, 07:998, 07:1007, 07:1011, *114*
\l__cmd_expandable_bool 07:20, 07:57, 07:62, 07:130, 07:142, 07:182, 07:194, 07:232, 07:254, 07:463, 07:472, 07:529, 07:560, 07:701, 07:711, 07:744, 07:794, *114*
__cmd_expandable_grab_D:nnNNNwNN 07:2013, 07:2037, 07:2043
__cmd_expandable_grab_D:NNNwNNn 07:2013, 07:2014, 07:2017
__cmd_expandable_grab_D:NNNwNNnnn 07:2013, 07:2028, 07:2035, 07:2061, 07:2156, *170*
__cmd_expandable_grab_D:Nw 07:2013, 07:2039, 07:2042
__cmd_expandable_grab_D:w 07:2013, 07:2013
__cmd_expandable_grab_D_-alt:NNwn . 07:2080, 07:2087, 07:2183
__cmd_expandable_grab_D_-alt:NNwNNn 07:2065, 07:2066, 07:2069
__cmd_expandable_grab_D_alt:Nwn 07:2065
__cmd_expandable_grab_D_alt:w 07:2065, 07:2065
__cmd_expandable_grab_E:w 07:2100, 07:2100
__cmd_expandable_grab_E_aux:w 07:2100, 07:2101, 07:2103, 07:2104, 07:2132
__cmd_expandable_grab_E_end:nnw 07:2100, 07:2116, 07:2133
__cmd_expandable_grab_E_-find:nnw . 07:2100, 07:2130, 07:2131

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

__cmd_expandable_grab_E_find:w 07:2100, 07:2121, 07:2129
__cmd_expandable_grab_E_long:w 07:2100, 07:2102
__cmd_expandable_grab_E_-loop:nnnNNw 07:2100, 07:2108, 07:2112, 07:2123
__cmd_expandable_grab_E_-test:nnw . 07:2100, 07:2105, 07:2106
__cmd_expandable_grab_m:w 07:2135, 07:2135, *149*
__cmd_expandable_grab_m_aux:wNn . . 07:2135, 07:2136, 07:2138, 07:2139
__cmd_expandable_grab_m_long:w 07:2135, 07:2137
__cmd_expandable_grab_R:w 07:2141, 07:2141
__cmd_expandable_grab_R_alt:w 07:2168, 07:2168
__cmd_expandable_grab_R_alt_-aux:NNwNNn 07:2168, 07:2169, 07:2172
__cmd_expandable_grab_R_-aux:NNNwNNn 07:2141, 07:2142, 07:2145
__cmd_expandable_grab_t:w 07:2195, 07:2195
__cmd_expandable_grab_t_-aux:NNwn . 07:2195, 07:2196, 07:2197
__cmd_flush_m_args: 07:794, 07:813, 07:820, 07:827, 07:835, 07:845, 07:852, 07:860, 07:892, 07:900, 07:908, 07:913, 07:913, *139*
\l__cmd_fn_code_tl 07:29, 07:307, 07:322, *114*
\l__cmd_fn_tl 07:28, 07:119, 07:306, 07:1540, 07:1607, 07:1631, 07:1637, 07:1647, 07:1657, 07:1666, 07:1671, 07:1675, 07:1706, 07:1732, 07:1740, 07:1742, 07:1747, 07:1749, 07:1760, 07:1761, 07:1766, 07:1767, 07:1772, 07:1773, 07:1778, 07:1779, 07:1784, 07:1785, 07:1790, 07:1791, 07:1796, 07:1797, 07:1802, 07:1803, 07:1833, 07:1839, *156*
\l__cmd_function_tl 07:24, 07:30, 07:88, 07:120, 07:136, 07:139, 07:153, 07:164, 07:165, 07:181, 07:189, 07:199, 07:202, 07:206, 07:208, 07:216, 07:222, 07:468, 07:532, 07:563, 07:704, 07:716, 07:747, 07:2554, *115*
__cmd_get_grabber:NN 07:1033, 07:1049, 07:1062, 07:1062, *142*
__cmd_get_grabber_auxi:NN 07:1062, 07:1065, 07:1068, 07:1076
__cmd_get_grabber_auxii:NN 07:1062, 07:1083, 07:1086
__cmd_grab_b:w 07:1526, 07:1526
__cmd_grab_b_aux:NNw 07:1526, 07:1527, 07:1529, 07:1531, 07:1533, 07:1534
__cmd_grab_b_end:Nw 07:1526, 07:1537, 07:1542
__cmd_grab_b_long:w 07:1526, 07:1528
__cmd_grab_b_long_obey_spaces:w 07:1526, 07:1532
__cmd_grab_b_obey_spaces:w 07:1526, 07:1530
__cmd_grab_D:w 07:1553, 07:1553
__cmd_grab_D_aux:NNnNN . 07:1536, 07:1597, 07:1599, 07:1604, 07:1816
__cmd_grab_D_aux:NNnNNN 07:1555, 07:1560, 07:1565, 07:1570, 07:1576, 07:1582, 07:1588, 07:1594, 07:1597, 07:1597
__cmd_grab_D_call:Nw 07:1601, 07:1661, 07:1661, 07:1818, *157*
__cmd_grab_D_long:w 07:1553, 07:1558
__cmd_grab_D_long_no_strip:w 07:1553, 07:1579
__cmd_grab_D_long_obey_spaces:w 07:1553, 07:1568
__cmd_grab_D_long_obey_spaces_-no_strip:w 07:1553, 07:1591
__cmd_grab_D_nested:NNnN 07:1610, 07:1624, 07:1627
__cmd_grab_D_nested:w 07:1624, 07:1642, 07:1659
__cmd_grab_D_no_strip:w 07:1553, 07:1573
__cmd_grab_D_obey_spaces:w 07:1553, 07:1563
__cmd_grab_D_obey_spaces_no_-strip:w 07:1553, 07:1585
__cmd_grab_E:nnNN 07:1680, 07:1682, 07:1688, 07:1694, 07:1700, 07:1704
__cmd_grab_E:w 07:1680, 07:1680
__cmd_grab_E_finalise: 07:1680, 07:1713, 07:1729, 07:1736
__cmd_grab_E_long:w 07:1680, 07:1686
__cmd_grab_E_long_obey_spaces:w 07:1680, 07:1698
__cmd_grab_E_loop:NnN . . 07:1680, 07:1709, 07:1724, 07:1726, 07:1733
__cmd_grab_E_obey_spaces:w 07:1680, 07:1692
\l__cmd_grab_expandably_bool 07:31, 07:121, 07:448, 07:473, 07:475, 07:537,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

07:567, 07:592, 07:606, 07:628, 07:645, 07:707, 07:750, 07:807, *115*
__cmd_grab_m:w 07:1737, 07:1737
__cmd_grab_m_1:w 07:1751
__cmd_grab_m_2:w 07:1751
__cmd_grab_m_3:w 07:1751
__cmd_grab_m_4:w 07:1751
__cmd_grab_m_5:w 07:1751
__cmd_grab_m_6:w 07:1751
__cmd_grab_m_7:w 07:1751
__cmd_grab_m_8:w 07:1751
__cmd_grab_m_9:w 07:1751
__cmd_grab_m_aux:Nnnnnnnnn 07:1751, 07:1751, 07:1760, 07:1766, 07:1772, 07:1778, 07:1784, 07:1790, 07:1796, 07:1802
__cmd_grab_m_long:w 07:1737, 07:1744
__cmd_grab_R:w 07:1810, 07:1810
__cmd_grab_R_aux:NNnN 07:1810, 07:1811, 07:1813, 07:1814
__cmd_grab_R_long:w 07:1810, 07:1812
__cmd_grab_t:w 07:1826, 07:1826
__cmd_grab_t_aux:NNw 07:1826, 07:1827, 07:1829, 07:1830
__cmd_grab_t_obey_spaces:w 07:1826, 07:1828
__cmd_grab_v:w 07:1842, 07:1842
__cmd_grab_v_aux:w 07:1842, 07:1845, 07:1850, 07:1852, *166*
__cmd_grab_v_aux_abort:n 07:1869, 07:1887, 07:1893, 07:1906, 07:1925, 07:1948, 07:1950, 07:1958, *165*
__cmd_grab_v_aux_catcodes: 07:1884, 07:1916, 07:1950, 07:1950, *164*
__cmd_grab_v_aux_loop:N 07:1879, 07:1885, 07:1889, 07:1903
__cmd_grab_v_aux_loop:NN 07:1879, 07:1892, 07:1895
__cmd_grab_v_aux_loop_end: 07:1879, 07:1900, 07:1908, 07:1939
__cmd_grab_v_aux_put:N 07:1883, 07:1902, 07:1918, 07:1936, 07:1944, 07:1978, 07:1978, 07:1980, 07:1994, 07:1996
__cmd_grab_v_aux_test:N 07:1868, 07:1879, 07:1879
__cmd_grab_v_bgroup: 07:1864, 07:1914, 07:1914
__cmd_grab_v_bgroup_loop: 07:1914, 07:1919, 07:1921, 07:1937, 07:1945
__cmd_grab_v_bgroup_loop:N 07:1914, 07:1924, 07:1927
__cmd_grab_v_group_end: 07:1842, 07:1873, 07:1910, 07:1960, *164*
__cmd_grab_v_long:w 07:1842, 07:1847
__cmd_grab_v_token_if_char:NTF 07:1881, 07:1897, 07:1929, 07:2005, 07:2005, *165*
\g__cmd_grabber_int 07:27, 07:1075, 07:1079, *114*
__cmd_if_recursion_tail_stop_-do:Nn . 07:46, 07:48, 07:2354, 07:2382
\c__cmd_ignore_def_tl .. 07:2546, 07:2562, 07:2569, 07:2583, 07:2604, 07:2633, 07:2640, 07:2647, 07:2655, 07:2663, 07:2672, 07:2679, 07:2686, 07:2693, 07:2700, 07:2712, 07:2719, *183*
\l__cmd_last_delimiters_tl 07:33, 07:446, 07:465, 07:591, 07:605, 07:627, 07:663, 07:713, 07:723, 07:777, *127*
\l__cmd_long_bool 07:34, 07:450, 07:543, 07:544, 07:630, 07:741, 07:752, 07:757, 07:761, 07:783, 07:814, 07:931, 07:940, 07:976, 07:991, 07:1004, 07:1021, 07:1041, 07:1844, 07:1849, 07:1954, *115*
\l__cmd_m_args_int 07:36, 07:786, 07:887, 07:915, 07:918, 07:920, 07:922, *115*
\l__cmd_nesting_a_tl 07:1624
\l__cmd_nesting_b_tl 07:1624
__cmd_new_env:nnnn 07:2840, 07:2844, 07:2858, 07:2889, 07:2915, 07:2916
__cmd_normalize_arg_spec:n 07:89, 07:443, 07:443
__cmd_normalize_arg_spec_loop:n 07:443, 07:455, 07:477, 07:538, 07:583, 07:593, 07:608, 07:631, 07:637, 07:647, 07:653
__cmd_normalize_check_gv:N 07:651, 07:699, 07:699
__cmd_normalize_check_lu:N 07:699, 07:709
__cmd_normalize_E_unique_-check:w 07:585, 07:601, 07:610, 07:615
__cmd_normalize_type_!:w ... 07:526
__cmd_normalize_type_+:w ... 07:526
__cmd_normalize_type_=:w ... 07:526
__cmd_normalize_type_>:w ... 07:526
__cmd_normalize_type_aux:NnNn .. 07:526, 07:542, 07:548, 07:557, 07:572
__cmd_normalize_type_b:w 07:655, 07:655

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

__cmd_normalize_type_D:w 07:506, 07:514, 07:516, 07:585, 07:585
__cmd_normalize_type_d:w 07:501, 07:503
__cmd_normalize_type_E:w 07:511, 07:585, 07:595
__cmd_normalize_type_e:w 07:501, 07:508
__cmd_normalize_type_m:w 07:633, 07:633
__cmd_normalize_type_O:w 07:501, 07:515
__cmd_normalize_type_o:w 07:501, 07:513
__cmd_normalize_type_R:w 07:520, 07:633, 07:639
__cmd_normalize_type_r:w 07:501, 07:517
__cmd_normalize_type_s:w 07:501, 07:522
__cmd_normalize_type_t:w 07:523, 07:585, 07:618
__cmd_normalize_type_v:w 07:633, 07:649
\l__cmd_obey_spaces_bool .. 07:32, 07:449, 07:549, 07:551, 07:624, 07:629, 07:758, 07:762, 07:771, 07:784, 07:821, 07:932, 07:941, *115*
__cmd_peek_cs_check_equal:NNN 07:2510, 07:2524, 07:2530
__cmd_peek_meaning:NTF 07:2501, 07:2510, 07:2510
__cmd_peek_meaning_aux:NNTF 07:2510, 07:2511, 07:2513, 07:2514
__cmd_peek_meaning_remove:NTF 07:1566, 07:1571, 07:1589, 07:1595, 07:1696, 07:1702, 07:1829, 07:2503, 07:2510, 07:2512
__cmd_peek_nonspace:NTF 07:2500, 07:2500
__cmd_peek_nonspace_aux:nNNTF 07:2500, 07:2501, 07:2503, 07:2504, 07:2507
__cmd_peek_nonspace_remove:NTF 07:1556, 07:1561, 07:1577, 07:1583, 07:1684, 07:1690, 07:1817, 07:1827, 07:2500, 07:2502
__cmd_peek_true_remove:NNw 07:2510
__cmd_peek_true_remove:Nw 07:2521, 07:2525, 07:2533, 07:2537
\l__cmd_prefixed_bool 07:37, 07:798, 07:815, 07:822, 07:828, 07:836, 07:885, *115*
__cmd_prepare_signature:N 07:780, 07:793, 07:796, 07:848, 07:856, 07:869, 07:888, 07:896, 07:904, 07:911, 07:977, 07:987, 07:1029, 07:1042, 07:1055, *136*
__cmd_prepare_signature:n 07:90, 07:780, 07:780
__cmd_prepare_signature_-bypass:N 07:780, 07:799, 07:801, 07:816, 07:823, 07:831, 07:841, *137*
\l__cmd_process_all_tl 07:38, 07:174, 07:309, 07:317, 07:382, 07:789, 07:919, 07:943, *125*
\l__cmd_process_one_tl 07:39, 07:790, 07:830, 07:839, 07:865, 07:874, 07:947, 07:950, *116*
\l__cmd_process_some_bool . 07:40, 07:173, 07:791, 07:829, 07:838, *116*
__cmd_provide_env:nnnn 07:2840, 07:2852, 07:2883, 07:2891, 07:2919, 07:2920
__cmd_put_arg_expandable:nw 07:2049, 07:2054, 07:2055, 07:2090, 07:2095, 07:2096, 07:2203, 07:2203, 07:2204
__cmd_renew_env:nnnn 07:2840, 07:2848, 07:2874, 07:2890, 07:2917, 07:2918
__cmd_replicate_processor:nn 07:864, 07:871, 07:871
__cmd_run_code: 07:311, 07:314, 07:314, 07:1534, 07:1542, 07:1550, 07:1553, 07:1558, 07:1563, 07:1568, 07:1574, 07:1580, 07:1586, 07:1592, 07:1680, 07:1686, 07:1692, 07:1698, 07:1722, 07:1737, 07:1744, 07:1755, 07:1757, 07:1763, 07:1769, 07:1775, 07:1781, 07:1787, 07:1793, 07:1799, 07:1810, 07:1812, 07:1830, 07:1852, 07:2010, *122*
\l__cmd_saved_args_tl 07:41, 07:1538, 07:1546, *116*
__cmd_set_environment_end:n 07:224, 07:275, 07:319
__cmd_set_eq_if_exist:NN 07:1093, 07:1111, 07:1113, 07:1132, 07:1151, 07:1152, 07:1153, 07:1166, 07:1167, 07:1168, 07:1266
__cmd_show:N 07:1310, 07:1312, 07:1312, 07:1339, 07:1362, 07:1439, 07:1441, 07:1523, *153*
__cmd_show:n 07:1329, 07:1425, 07:1431
__cmd_show_command:N 07:1318, 07:1329, 07:1329

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

__cmd_show_command:NnNNwN 07:1329, 07:1330, 07:1331
__cmd_show_command_aux:NnNNn 07:1329, 07:1333, 07:1346, 07:1355, 07:1364, 07:1425, *153*
__cmd_show_delim:Nw 07:1471, 07:1471
__cmd_show_delims:Nw 07:1471, 07:1473
__cmd_show_delims_opt:Nw 07:1471, 07:1475
__cmd_show_E:Nw 07:1471, 07:1490
__cmd_show_e:Nw 07:1471, 07:1479
__cmd_show_environment:N 07:1321, 07:1329, 07:1411, 07:1436
__cmd_show_environment:Nnnw 07:1413, 07:1421
__cmd_show_environment_end:N 07:1322, 07:1433
__cmd_show_expandable:N 07:1319, 07:1329, 07:1336
__cmd_show_expandable:NnNNNNnN 07:1329, 07:1337, 07:1342, 07:1344, 07:1351, 07:1353, 07:1359
__cmd_show_opt:Nw .. 07:1471, 07:1477
__cmd_show_optimized:N 07:1320, 07:1329, 07:1374
__cmd_show_optimized:NN 07:1376, 07:1380
__cmd_show_optimized_aux:N 07:1382, 07:1405
__cmd_show_prefix:Nw 07:1471, 07:1503
__cmd_show_processor:Nw 07:1471, 07:1505
\c__cmd_show_type_!_tl 07:1465
\c__cmd_show_type_+_tl 07:1465
\c__cmd_show_type_>_tl 07:1465
\c__cmd_show_type_D_tl 07:1465
\c__cmd_show_type_d_tl 07:1465
\c__cmd_show_type_E_tl 07:1465
\c__cmd_show_type_e_tl 07:1465
\c__cmd_show_type_O_tl 07:1465
\c__cmd_show_type_R_tl 07:1465
\c__cmd_show_type_r_tl 07:1465
\c__cmd_show_type_t_tl 07:1465
\l__cmd_signature_tl 07:42, 07:167, 07:210, 07:792, 07:855, 07:868, 07:895, 07:903, 07:917, 07:926, 07:1059, 07:1537, 07:1606, 07:1712, 07:1722, 07:1739, 07:1746, 07:1755, 07:1759, 07:1765, 07:1771, 07:1777, 07:1783, 07:1789, 07:1795, 07:1801, 07:1832, 07:1854, 07:2010, *116*
__cmd_single_token_check:n 07:588, 07:589, 07:599, 07:621, 07:642, 07:643, 07:671, 07:671
\l__cmd_some_long_bool 07:44, 07:114, 07:143, 07:204, 07:212, 07:453, 07:742, 07:753, *116*
\l__cmd_some_obey_spaces_bool 07:43, 07:452, 07:552, 07:766, *116*
\l__cmd_some_short_bool 07:45, 07:113, 07:203, 07:214, 07:454, 07:754, *116*
__cmd_split_add_item:n 07:1504, 07:1507, 07:1506, 07:1512, *155*
__cmd_split_argument:nnn 07:2246, 07:2246, 07:3002
__cmd_split_argument_aux:n 07:2246, 07:2262, 07:2281
__cmd_split_argument_aux:nnnn 07:2246, 07:2249, 07:2253
__cmd_split_argument_aux:wn 07:2246, 07:2282, 07:2283
__cmd_split_end_item:n . 07:1463, 07:1472, 07:1474, 07:1476, 07:1478, 07:1484, 07:1495, 07:1507, 07:1514, *155*
__cmd_split_list:nn 07:2211, 07:2213, 07:2248, 07:3003
__cmd_split_list_multi:nn 07:2211, 07:2218, 07:2221, 07:2223, 07:2230, 07:2243
\l__cmd_split_list_seq 07:2211, 07:2225, 07:2227, *175*
__cmd_split_list_single:Nn 07:2211, 07:2219, 07:2233
\l__cmd_split_list_tl 07:2212, 07:2235, 07:2241, 07:2243, *175*
__cmd_split_N_head_apply:Nn 07:2291, 07:2311, 07:2323, 07:2398
__cmd_split_N_head_apply_-aux:NNw .. 07:2291, 07:2399, 07:2400
__cmd_split_signature:n 07:1366, 07:1442, 07:1442, *152*
__cmd_split_signature_loop:Nw .. 07:1447, 07:1449, 07:1449, 07:1463, 07:1472, 07:1474, 07:1476, 07:1478, 07:1486, 07:1499, 07:1504, 07:1506
__cmd_split_start_item: 07:1452, 07:1483, 07:1494, 07:1507, 07:1507, *154*
__cmd_start:nNNnnn 07:163, 07:283, 07:294, 07:2481, *138*
__cmd_start_aux:NNnnnn 07:289, 07:300, 07:303, 07:303, 07:313
__cmd_start_env:nnnnn 07:159, 07:283, 07:283, 07:2484, *181*
__cmd_start_expandable:nNNNNn .. 07:197, 07:401, 07:401, 07:2482, *146*
__cmd_start_optimized: .. 07:106, 07:135, 07:149, 07:1185, 07:2483, *181*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\l__cmd_suppress_strip_bool 07:35, 07:451, 07:558, 07:566, 07:785, 07:837, 07:934, 07:942, *115*
__cmd_tl_mapthread_function:NNN 07:352, 07:380, 07:2447, 07:2447
__cmd_tl_mapthread_function:nnN 07:428, 07:1497, 07:2447, 07:2460, *155*
__cmd_tl_mapthread_loop:w 07:2447, 07:2450, 07:2462, 07:2466, 07:2471
__cmd_tmp:w 07:49, 07:52, 07:338, 07:354, 07:396, 07:398, 07:501, 07:525, 07:1064, 07:1070, 07:1088, 07:1295, 07:1305, 07:1465, 07:1468, 07:1470, 07:1488, 07:1492, 07:1498, 07:1502, 07:2015, 07:2034, 07:2067, 07:2086, 07:2143, 07:2167, 07:2170, 07:2194, 07:2542, *169*
\l__cmd_tmp_prop 07:49, 07:1708, 07:1711, 07:1717, *117*
\l__cmd_tmpa_tl ... 07:50, 07:350, 07:355, 07:365, 07:1033, 07:1035, 07:1049, 07:1052, 07:1206, 07:1212, 07:1227, 07:1235, 07:1251, 07:1258, 07:1276, 07:1279, 07:1370, 07:1435, 07:1436, 07:1445, 07:1516, 07:1645, 07:1648, 07:2516, 07:2541, 07:2544, *117*
\l__cmd_tmpb_tl .. 07:51, 07:1018, 07:1023, 07:1034, 07:1277, 07:1279, 07:1446, 07:1452, 07:1509, 07:1513, 07:1517, 07:1518, 07:1717, 07:1718, 07:1720, 07:2517, 07:2528, 07:2534, *154*
__cmd_token_if_cs:NTF 07:1669, 07:2438, 07:2438, 07:2523, *133*
__cmd_trim_spaces:n 07:2289, 07:2289, 07:3004
__cmd_use_i_delimit_by_q_-recursion_stop:nw . 07:46, 07:2358
\l__cmd_v_arg_tl 07:1841, 07:1858, 07:1877, 07:1911, 07:1966, 07:1973, 07:1982, 07:1998, *164*
\l__cmd_v_nesting_int .. 07:1913, 07:1917, 07:1933, 07:1934, 07:1943, *166*
\cmrFont 33:1249, 33:1252
\colon 30:510
\color *410*
\columnbreak *952*
\columnsep 20:27, 20:97, 20:154, 54:81, 54:204
\columnseprule ... 54:82, 54:2264, 54:2298
\columnwidth 20:24, 20:27, 20:28, 20:30, 20:94, 20:97, 20:98, 20:100, 20:151, 20:154, 20:155, 20:158, 40:444, 40:483, 40:501, 40:518, 45:99, 45:168, 45:470, 45:489, 45:507, 54:80, 54:146, 54:147, 54:148, 54:203, 54:204, 54:205, 54:206, 54:207, 54:1847, 54:1849, 54:2262, 54:2266, 54:2294, 54:2300
\cong 30:464
\contentsline 44:164, 44:171, 44:177, 44:184, *908*
\coprod 30:348
\copyright .. 21:285, 21:316, 29:649, *1218*
\coremissesfalse 33:1256
\coremissestrue 33:1234
\cos 38:12
\cosh 38:14
\cot 38:18
\coth 38:19
\countdef 04:75, 04:85, 04:174, 04:190, 04:198, 04:206, 04:225, 34:3, 02:37, 57:75, 02:38, 02:39, 02:41, 02:51, 02:90, 01:50
counter 36:242, *751*
\counterwithin *512*
\counterwithin 22:123, *1258*
\counterwithout *512*
\counterwithout 22:123, *1258*
\CountZero 04:225
\cr 21:500, 21:506, 21:516, 21:522, 33:101, 33:105, 33:927, 33:931, 38:188, 38:192, 38:378, 38:424, 38:540, 41:192, 41:203, 41:210, 41:219, 41:224, 41:230, 41:377, 42:141, 42:143, 42:148, 42:154, 02:446, *1166*
\crcr 21:327, 21:362, 21:363, 21:390, 21:394, 21:397, 21:476, 21:480, 21:484, 21:486, 21:489, 21:732, 21:760, 21:764, 21:767, 21:834, 21:837, 21:895, 21:1253, 29:651, 30:344, 30:345, 30:347, 30:466, 30:469, 30:473, 30:537, 30:538, 30:539, 30:540, 30:541, 30:542, 30:544, 30:545, 30:546, 30:547, 30:548, 30:550, 33:106, 33:932, 38:168, 38:170, 38:171, 38:172, 38:188, 38:190, 38:191, 38:192, 38:210, 38:211, 41:171, 41:172, 42:141, 55:237, 02:536, *1171*
create commands:
create_callback 04:773
\create_callback *46*
\CS *82*
cs commands:
\cs:w ... 08:2206, 08:2225, 08:2231, 08:2245, 08:2255, 08:2277, 08:2331, 08:2418, 08:2483, 08:2533, 08:2916, 08:2936, 08:2945, 08:2946, 28:319,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

28:322, 51:180, 52:211, 52:225, 52:237, 52:238, 07:1672, 07:2487, 08:381, 08:1066, 08:1123, 08:1274, 08:1414, 08:1521, 08:1522, 08:1583, 08:1603, 08:1606, 08:1667, 08:1692, 08:1695, 08:1713, 08:1714, 08:1738, 08:1739, 08:1740, 08:1747, 08:1754, *236*
`\cs_argument_spec:N` 05:141, *1268*
`\cs_end:` 08:2099, 08:2205, 08:2209, 08:2213, 08:2225, 08:2230, 08:2231, 08:2245, 08:2251, 08:2265, 08:2278, 08:2283, 08:2331, 08:2417, 08:2418, 08:2483, 08:2533, 08:2916, 08:2936, 08:2946, 28:319, 28:322, 51:180, 52:211, 52:225, 52:237, 52:238, 07:1672, 07:2489, 08:381, 08:767, 08:1066, 08:1123, 08:1277, 08:1417, 08:1521, 08:1522, 08:1583, 08:1603, 08:1606, 08:1667, 08:1692, 08:1695, 08:1713, 08:1714, 08:1729, 08:1739, 08:1740, 08:1747, 08:1754
`\cs_generate_from_arg_count:NNnn` 10:81, 10:100, 11:397, 11:459, 11:818, 07:138, 07:152, 07:180, 07:188, 07:268, 07:354, 07:357, 07:396, *1266*
`\cs_generate_variant:Nn` ... 09:20, 11:64, 11:349, 11:502, 11:703, 11:734, 11:807, 20:528, 36:37, 36:52, 36:62, 36:79, 36:96, 36:116, 36:117, 36:230, 48:264, 51:51, 51:52, 51:53, 51:177, 52:461, 52:469, 57:596, 07:236, 07:313, 07:357, 07:1113, 07:1410, 07:2012, 07:2204, 07:2230, 07:2889, 07:2890, 07:2891, 08:37, 08:38, 08:39, 08:40, 08:41, 08:42, 08:43, 08:53, 08:54, 08:57, 08:66, 08:1080, 08:1103, 08:1633, 08:1716
`\cs_gset:Npn` 08:2288, 08:2358, 11:811, 28:285, 28:324, 36:24, 52:242, 53:173, 08:1079, 08:1099, 08:1629, 08:2060
`\cs_gset:Npx` 52:213, 08:1047
`\cs_gset_eq:NN` 08:2194, 08:2295, 08:2296, 08:2297, 08:2298, 08:2562, 09:56, 09:539, 09:567, 11:642, 28:283, 28:291, 28:316, 28:330, 05:136, 05:137, 05:138, 05:140, 05:141, 05:142, 05:143, 05:144, 05:145, 53:52, 53:151, 53:166, 53:167, 53:172, 53:181, 53:185, 53:321, 53:387, 07:3006, 08:770, 08:1424, 08:1450, 08:1465, 08:1466, 08:1469
`\cs_gset_nopar:Npn` 08:2944
`\cs_gset_nopar:Npx` 08:50, 08:52
`\cs_gset_protected:Npn` 08:2139, 08:2142, 08:2170, 08:2175, 08:2193, 08:2352, 08:2382, 08:2778, 08:2921, 09:231, 09:423, 09:545, 11:399, 11:828, 51:62, 08:88, 08:91, 08:100, 08:133, 08:160, 08:188, 08:191, 08:197, 08:221, 08:226, 08:251, 08:310, 08:328, 08:581, 08:584, 08:602, 08:628, 08:677, 08:685, 08:1448, 08:1509
`\cs_gset_protected:Npx` 10:21, 10:23, 48:318, 53:20, 08:20, 08:1161
`\cs_if_eq:NNTF` 51:97, 51:99, 51:165, 07:1070, 07:1728
`\cs_if_exist:N` *1032*
`\cs_if_exist:NTF` 08:2464, 08:2493, 08:2501, 09:73, 09:81, 10:76, 10:98, 10:119, 11:54, 11:60, 11:73, 11:86, 11:395, 11:453, 11:488, 11:640, 11:663, 11:679, 36:67, 36:157, 05:139, 51:167, 52:93, 52:438, 52:442, 55:8, 07:67, 07:229, 07:245, 07:1073, 07:1112, 07:2808, 07:2821, 07:2833, 07:2860, 07:2866, 07:2876, 07:2885, 07:2897, 07:2900, 07:2907, 07:2912, 07:2927, 07:2940, 07:2953, 08:1051, 08:1116, *1053*
`\cs_if_exist_p:N` . 07:74, 07:75, 08:277
`\cs_if_exist_use:NTF` 11:728, 05:184, 53:313, 53:314, 53:318, 53:319, 07:481, 07:1229, 08:1283, 08:1327, 08:1352
`\cs_if_free:NTF` 36:8, 51:106
`\cs_meaning:N` 10:64
`\cs_new:Npn` 08:2203, 08:2228, 08:2236, 08:2249, 08:2257, 08:2266, 08:2275, 08:2310, 08:2491, 08:2499, 08:2513, 08:2531, 08:2540, 08:2545, 08:2786, 08:2787, 08:2788, 08:2789, 08:2793, 08:2794, 09:303, 09:313, 09:323, 09:325, 09:327, 09:338, 09:361, 09:362, 09:365, 09:485, 09:593, 10:83, 10:142, 11:350, 11:351, 11:454, 11:508, 11:908, 11:1186, 11:1188, 11:1190, 16:85, 16:88, 16:113, 16:125, 20:519, 36:63, 36:65, 36:89, 36:97, 36:104, 36:167, 36:168, 36:182, 36:183, 36:197, 36:198, 05:182, 48:250, 48:260, 48:261, 48:262, 48:263, 51:174, 51:176, 52:30, 52:38, 52:44, 52:57, 52:95, 52:100, 52:105, 52:112,

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

52:127, 52:234, 52:240, 52:411, 52:413, 52:415, 52:418, 52:422, 52:436, 52:452, 52:462, 52:582, 53:54, 53:59, 53:77, 53:141, 53:146, 53:163, 53:178, 53:183, 53:189, 53:192, 53:206, 53:257, 53:310, 53:323, 53:337, 53:340, 53:350, 53:392, 53:511, 53:512, 53:513, 55:14, 55:28, 55:128, 55:134, 55:186, 55:189, 55:232, 07:96, 07:98, 07:149, 07:401, 07:406, 07:408, 07:410, 07:422, 07:434, 07:440, 07:1136, 07:1191, 07:1272, 07:1297, 07:1302, 07:1659, 07:2013, 07:2017, 07:2035, 07:2042, 07:2043, 07:2065, 07:2069, 07:2087, 07:2100, 07:2102, 07:2104, 07:2106, 07:2112, 07:2129, 07:2131, 07:2133, 07:2135, 07:2137, 07:2139, 07:2141, 07:2145, 07:2168, 07:2172, 07:2195, 07:2197, 07:2203, 07:2281, 07:2283, 07:2308, 07:2314, 07:2320, 07:2326, 07:2398, 07:2400, 07:2447, 07:2460, 07:2466, 07:2548, 07:2962, 07:2979, 07:2980, 07:2984, 07:2985, 07:2986, 08:45, 08:46, 08:330, 08:336, 08:351, 08:361, 08:362, 08:364, 08:378, 08:384, 08:904, 08:906, 08:908, 08:1202, 08:1214, 08:1219, 08:1227, 08:1236, 08:1238, 08:1245, 08:1272, 08:1279, 08:1281, 08:1396, 08:1412, 08:1546, 08:1547, 08:1718, 08:1719, 08:2056, 08:2063

`\cs_new_eq:NN` 08:2265, 08:2807, 08:2808, 08:2809, 08:2810, 08:2880, 08:2882, 09:13, 09:14, 09:440, 10:7, 10:8, 10:138, 10:139, 10:140, 10:141, 10:145, 10:170, 10:171, 10:172, 10:173, 10:174, 10:175, 10:176, 10:177, 11:356, 11:357, 11:358, 11:373, 16:135, 20:797, 20:798, 36:166, 36:181, 36:196, 05:162, 05:163, 05:164, 05:165, 05:181, 48:305, 48:321, 48:322, 48:363, 48:365, 48:475, 48:477, 51:8, 52:252, 52:253, 52:509, 52:511, 52:513, 52:515, 52:517, 52:519, 52:521, 52:523, 52:525, 52:527, 53:7, 53:152, 53:347, 53:349, 53:411, 53:414, 53:415, 53:510, 55:3, 55:4, 55:123, 57:624, 57:625, 07:52, 07:95, 07:1247, 07:1248, 07:1255, 07:1262, 07:2802, 07:2803, 07:2981, 07:2982, 07:2983, 07:2989, 07:2990, 07:2991, 07:3001, 07:3002, 07:3003, 07:3004, 07:3005, 08:7, 08:23, 08:34, 08:63, 08:1186, 08:1369, 08:1376, 08:1419, 08:1793, 08:1794, 08:1795, 08:1796, 08:1797, 08:1798

`\cs_new_protected:Npe` 11:275

`\cs_new_protected:Npn` . . 08:2074, 08:2092, 08:2097, 08:2106, 08:2124, 08:2130, 08:2144, 08:2157, 08:2184, 08:2188, 08:2198, 08:2211, 08:2220, 08:2234, 08:2240, 08:2263, 08:2270, 08:2293, 08:2316, 08:2326, 08:2337, 08:2343, 08:2363, 08:2393, 08:2395, 08:2405, 08:2547, 08:2553, 08:2559, 08:2730, 08:2731, 08:2732, 08:2755, 08:2766, 08:2796, 08:2797, 08:2798, 08:2799, 08:2811, 08:2818, 08:2825, 08:2832, 08:2839, 08:2841, 08:2843, 08:2850, 08:2857, 08:2864, 08:2871, 09:18, 09:23, 09:25, 09:34, 09:36, 09:46, 09:54, 09:63, 09:77, 09:97, 09:117, 09:130, 09:135, 09:144, 09:149, 09:156, 09:296, 09:367, 09:404, 09:441, 09:469, 09:498, 10:9, 10:14, 10:19, 10:27, 10:53, 10:72, 10:73, 10:95, 10:116, 10:132, 11:42, 11:52, 11:58, 11:65, 11:71, 11:96, 11:104, 11:120, 11:128, 11:136, 11:146, 11:162, 11:172, 11:182, 11:188, 11:203, 11:219, 11:241, 11:253, 11:270, 11:296, 11:320, 11:352, 11:354, 11:359, 11:361, 11:363, 11:365, 11:367, 11:369, 11:371, 11:374, 11:393, 11:401, 11:412, 11:414, 11:428, 11:432, 11:446, 11:503, 11:522, 11:540, 11:561, 11:578, 11:585, 11:593, 11:598, 11:607, 11:613, 11:618, 11:627, 11:632, 11:645, 11:654, 11:673, 11:686, 11:698, 11:704, 11:710, 11:715, 11:735, 11:743, 11:748, 11:759, 11:765, 11:785, 11:802, 11:808, 11:814, 11:825, 11:831, 11:843, 11:853, 11:872, 11:874, 11:885, 11:890, 11:896, 11:902, 11:909, 11:911, 11:913, 11:922, 11:931, 11:940, 11:948, 11:1156, 11:1158, 11:1160, 11:1162, 11:1164, 11:1166, 11:1168, 11:1170, 11:1172, 11:1174, 11:1176, 11:1178, 11:1180, 11:1182, 11:1184, 11:1192, 11:1193, 11:1194, 16:93, 28:278, 36:6, 36:17, 36:22, 36:38, 36:43, 36:48, 36:50, 36:53, 36:128, 36:142, 36:149, 36:199, 36:206, 36:214, 48:7,

48:17, 48:65, 48:73, 48:117, 48:122, 48:147, 48:156, 48:194, 48:205, 48:213, 48:216, 48:248, 48:306, 48:311, 48:316, 48:324, 48:352, 48:358, 48:381, 48:412, 51:22, 51:28, 51:44, 51:58, 51:60, 51:95, 51:104, 51:116, 51:132, 51:137, 51:146, 51:158, 51:163, 51:178, 51:182, 51:200, 51:214, 52:49, 52:62, 52:70, 52:79, 52:208, 52:222, 52:392, 52:397, 52:420, 52:547, 53:8, 53:13, 53:18, 53:343, 53:473, 55:5, 55:6, 55:15, 55:16, 55:17, 55:18, 55:124, 55:155, 55:158, 55:178, 55:182, 55:192, 55:195, 55:198, 55:201, 55:204, 55:221, 57:577, 57:605, 57:610, 57:615, 57:623, 07:55, 07:60, 07:65, 07:86, 07:106, 07:128, 07:150, 07:178, 07:226, 07:240, 07:261, 07:275, 07:283, 07:303, 07:314, 07:324, 07:335, 07:343, 07:348, 07:358, 07:363, 07:377, 07:386, 07:394, 07:443, 07:477, 07:503, 07:508, 07:513, 07:515, 07:517, 07:522, 07:526, 07:540, 07:546, 07:555, 07:572, 07:585, 07:595, 07:610, 07:618, 07:633, 07:639, 07:649, 07:655, 07:671, 07:680, 07:699, 07:709, 07:721, 07:732, 07:737, 07:738, 07:764, 07:780, 07:796, 07:801, 07:811, 07:818, 07:825, 07:833, 07:843, 07:850, 07:858, 07:871, 07:882, 07:890, 07:898, 07:906, 07:913, 07:924, 07:952, 07:962, 07:967, 07:974, 07:979, 07:981, 07:989, 07:1002, 07:1015, 07:1031, 07:1037, 07:1044, 07:1046, 07:1057, 07:1062, 07:1068, 07:1086, 07:1093, 07:1111, 07:1114, 07:1129, 07:1148, 07:1163, 07:1177, 07:1208, 07:1216, 07:1224, 07:1233, 07:1245, 07:1249, 07:1256, 07:1261, 07:1263, 07:1274, 07:1281, 07:1289, 07:1312, 07:1329, 07:1331, 07:1336, 07:1344, 07:1353, 07:1364, 07:1374, 07:1380, 07:1411, 07:1421, 07:1431, 07:1433, 07:1442, 07:1449, 07:1471, 07:1473, 07:1475, 07:1477, 07:1479, 07:1490, 07:1503, 07:1505, 07:1507, 07:1512, 07:1514, 07:1526, 07:1528, 07:1530, 07:1532, 07:1534, 07:1542, 07:1553, 07:1558, 07:1563, 07:1568, 07:1573, 07:1579, 07:1585, 07:1591, 07:1597, 07:1604, 07:1627, 07:1680, 07:1686, 07:1692, 07:1698, 07:1704, 07:1726, 07:1736, 07:1737, 07:1744, 07:1757, 07:1763, 07:1769, 07:1775, 07:1781, 07:1787, 07:1793, 07:1799, 07:1810, 07:1812, 07:1814, 07:1826, 07:1828, 07:1830, 07:1842, 07:1847, 07:1852, 07:1873, 07:1879, 07:1889, 07:1895, 07:1908, 07:1921, 07:1927, 07:1950, 07:1958, 07:1980, 07:1996, 07:2005, 07:2007, 07:2205, 07:2213, 07:2223, 07:2233, 07:2246, 07:2253, 07:2289, 07:2291, 07:2296, 07:2332, 07:2337, 07:2348, 07:2350, 07:2352, 07:2369, 07:2380, 07:2390, 07:2392, 07:2394, 07:2396, 07:2406, 07:2410, 07:2438, 07:2473, 07:2494, 07:2500, 07:2502, 07:2504, 07:2510, 07:2512, 07:2514, 07:2530, 07:2537, 07:2804, 07:2817, 07:2830, 07:2835, 07:2842, 07:2846, 07:2850, 07:2854, 07:2858, 07:2874, 07:2883, 07:2895, 07:2905, 07:2911, 07:2913, 07:2923, 07:2936, 07:2949, 07:2957, 08:8, 08:13, 08:18, 08:47, 08:49, 08:51, 08:55, 08:58, 08:64, 08:69, 08:71, 08:73, 08:104, 08:148, 08:172, 08:174, 08:176, 08:201, 08:203, 08:205, 08:231, 08:266, 08:268, 08:285, 08:290, 08:292, 08:385, 08:394, 08:399, 08:407, 08:431, 08:433, 08:449, 08:457, 08:467, 08:477, 08:488, 08:494, 08:500, 08:516, 08:529, 08:555, 08:572, 08:632, 08:651, 08:659, 08:666, 08:698, 08:704, 08:710, 08:720, 08:762, 08:764, 08:772, 08:774, 08:777, 08:779, 08:960, 08:962, 08:997, 08:1034, 08:1046, 08:1049, 08:1078, 08:1091, 08:1093, 08:1095, 08:1097, 08:1114, 08:1138, 08:1145, 08:1303, 08:1310, 08:1341, 08:1363, 08:1370, 08:1377, 08:1383, 08:1388, 08:1393, 08:1394, 08:1422, 08:1475, 08:1555, 08:1637, 08:1720, 08:1727, 08:1736, 08:1743, 08:1750, 08:1757, 08:1765, 08:1771, 08:1782, 08:1799, 08:1806, 08:1818, 08:1823, 08:1828, 08:1830, 08:1834, 08:1952, *1082*

`\cs_new_protected:Npx` . . . 09:475, 48:100, 07:294, 07:1805, 07:1914

`\cs_new_protected_nopar:Npn` 57:620, 07:1661, 07:1751

`\cs_parameter_spec:N` 05:139, 05:140, 07:1402, 08:1230, *1268*

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

\cs_prefix_spec:N 09:207, 09:282, 05:138
\cs_replacement_spec *253*
\cs_replacement_spec:N 05:143, 07:1371, 07:1407, 07:1417, 08:37, 08:41, 08:1205, 08:1440, 08:1462, 08:1876, 08:1889, 08:1898
\cs_set:Npe 07:1382
\cs_set:Npn 08:2551, 08:2557, 08:2930, 09:199, 09:274, 09:456, 09:515, 09:551, 10:102, 11:328, 11:812, 16:37, 16:68, 07:190, 07:269, 07:354, 07:396, 07:1064, 08:1147, 08:1158, 08:1165, 08:1174, 08:1873, *156*
\cs_set:Npx 52:211, 52:225, 07:219, 07:992, 07:1005
\cs_set_eq:NN 09:171, 09:172, 09:192, 09:193, 09:198, 09:246, 09:247, 09:267, 09:268, 09:273, 09:452, 09:453, 09:511, 09:512, 09:547, 09:548, 09:606, 11:681, 16:91, 16:92, 16:136, 16:137, 16:138, 16:140, 16:141, 16:142, 16:176, 16:177, 16:178, 16:181, 28:294, 28:333, 36:231, 48:99, 48:251, 48:252, 48:253, 51:85, 53:24, 53:44, 53:83, 53:95, 53:125, 53:132, 53:417, 53:419, 53:421, 53:423, 53:425, 55:118, 55:121, 07:271, 07:272, 07:1088, 07:1112, 07:1131, 07:1150, 07:1165, 07:1179, 07:1240, 07:1265, 07:1270, 07:1285, 07:1287, 07:1760, 07:1766, 07:1772, 07:1778, 07:1784, 07:1790, 07:1796, 07:1802, 07:1952, 07:2542, 08:1494, 08:1495, 08:1496, 08:1497, 08:1527, 08:1529, 08:1531, 08:1533, 08:1820, 08:1825
\cs_set_nopar:Npe 07:131, 07:1182
\cs_set_nopar:Npn 57:579, *156*
\cs_set_nopar:Npx 07:195, 07:216, 07:221, 07:266, 07:277, 07:993, 07:1006, 07:1156, 07:1170, 07:1283, 08:48
\cs_set_protected:Npe 11:665
\cs_set_protected:Npn .. 08:2078, 08:2109, 09:173, 09:248, 11:324, 11:829, 28:311, 28:343, 51:5, 51:64, 51:78, 51:188, 53:46, 53:135, 53:412, 57:594, 07:154, 07:501, 07:1295, 07:1465, 07:1488, 07:1492, 07:1529, 07:1533, 07:1560, 07:1570, 07:1582, 07:1594, 07:1689, 07:1701, 07:1713, 07:1747, 07:1813, 07:1833, 07:2015, 07:2067, 07:2143, 07:2170, 08:420
\cs_set_protected_nopar:Npe 07:132, 07:1181
\cs_set_protected_nopar:Npn 07:1527, 07:1531, 07:1555, 07:1565, 07:1576, 07:1588, 07:1683, 07:1695, 07:1740, 07:1811
\cs_set_protected_nopar:Npx 07:155, 07:195, 07:1133, 07:1156, 07:1268
\cs_show:N 11:912, *375*
\cs_to_str:N 09:112, 09:133, 09:147, 09:152, 05:136, 07:74, 07:75, 07:88, 07:1105, 07:1107, 07:1325, 07:1377, 07:1416, 07:1418, 07:1435, 07:1672, 07:2489, *236*
\cs_undefine:N 08:2303, 08:2389, 08:2527, 09:59, 09:210, 09:494, 09:502, 52:227, 07:2915, 07:2916, 07:2917, 07:2918, 07:2919, 07:2920, 07:2995, 07:2996, 07:2997, 07:3007, 08:79, 08:80, 08:90, 08:132, 08:190, 08:196, 08:271, 08:649, 08:1084, 08:1085, 08:1086, 08:1087, 08:1107, 08:1108, 08:1109, 08:1110, 08:1134, 08:1190, 08:1210, 08:1268, 08:1289, 08:1290, 08:1301, 08:1395

cs\check@icr commands:
\cs_gset_eq:NN *76*
\csc 38:21
\csname *200*
\csname\endcsname *1051*
\cup 30:380
\CurrentFile 50:938, 52:3, 52:67, 52:84, 52:166, 52:171, 52:174, 52:377, 52:381, 52:555, 52:556, *1046*
\CurrentFilePath 52:3, 52:67, 52:83, 52:165, 52:378, 52:381, *1052*
\CurrentFilePathUsed 52:3, 52:66, 52:81, 52:163, 52:375, 52:378, *1037*
\CurrentFileUsed 50:938, 52:3, 52:66, 52:82, 52:164, 52:375, 52:377, 52:554, 52:555, *1037*
\currentgrouplevel 39:148, 50:780
\CurrentOption 21:1536, 21:1539, 21:1544, 21:1560, 50:14, 50:520, 50:531, 50:544, 50:545, 50:550, 50:563, 50:564, 50:566, 50:580, 50:581, 50:584, 50:598, 50:603, 50:604, 50:617, 50:622, 50:623, 50:636, 50:638, 50:644, 50:646, 50:658, 50:659, 50:660, 50:668, 50:669, 50:670, 50:988, 50:1053, 50:1172, 50:1173, 50:1183, 51:82, 51:83, *1200*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

CurrentOption commands:
\CurrentOption: 50:543, 50:562, 50:579, 50:597, 50:602, 50:616, 50:621, 50:657, 50:667, 50:1182
\currsubencoding 33:1248, 33:1259, 33:1415, 33:1417, 33:1418
\CYRA 21:1508
\cyra 21:1508, 21:1563
\CYRABHCH 21:1508
\cyrabhch 21:1508
\CYRABHCHDSC 21:1508
\cyrabhchdsc 21:1508
\CYRABHDZE 21:1509
\cyrabhdze 21:1508
\CYRABHHA 21:1509
\cyrabhha 21:1509
\CYRAE 21:1509
\cyrae 21:1509
\CYRB 21:1509
\cyrb 21:1509
\CYRBYUS 21:1510
\cyrbyus 21:1509
\CYRC 21:1510
\cyrc 21:1510
\CYRCH 21:1510
\cyrch 21:1510
\CYRCHLDSC 21:1510
\cyrchldsc 21:1510
\CYRCHRDSC 21:1511
\cyrchrdsc 21:1510
\CYRCHVCRS 21:1511
\cyrchvcrs 21:1511
\CYRD 21:1511
\cyrd 21:1511
\CYRDELTA 21:1511
\cyrdelta 21:1511
\CYRDJE 21:1512
\cyrdje 21:1512
\CYRDZE 21:1512
\cyrdze 21:1512
\CYRDZHE 21:1512
\cyrdzhe 21:1512
\CYRE 21:1512
\cyre 21:1512
\CYREPS 21:1513
\cyreps 21:1512
\CYREREV 21:1513
\cyrerev 21:1513
\CYRERY 21:1513
\cyrery 21:1513
\CYRF 21:1513
\cyrf 21:1513
\CYRFITA 21:1514
\cyrfita 21:1513
\CYRG 21:1514
\cyrg 21:1514
\CYRGDSC 21:1514
\cyrgdsc 21:1514
\CYRGDSCHCRS 21:1514
\cyrgdschcrs 21:1514
\CYRGHCRS 21:1515
\cyrghcrs 21:1515
\CYRGHK 21:1515
\cyrghk 21:1515
\CYRGUP 21:1515
\cyrgup 21:1515
\CYRH 21:1515
\cyrh 21:1515
\CYRHDSC 21:1516
\cyrhdsc 21:1516
\CYRHHCRS 21:1516
\cyrhhcrs 21:1516
\CYRHHK 21:1516
\cyrhhk 21:1516
\CYRHRDSN 21:1517
\cyrhrdsn 21:1516
\CYRI 21:1517
\cyri 21:1517
\CYRIE 21:1517
\cyrie 21:1517
\CYRII 21:1517
\cyrii 21:1517
\CYRISHRT 21:1517
\cyrishrt 21:1517
\CYRISHRTDSC 21:1518
\cyrishrtdsc 21:1518
\CYRIZH 21:1518
\cyrizh 21:1518
\CYRJE 21:1518
\cyrje 21:1518
\CYRK 21:1518
\cyrk 21:1518
\CYRKBEAK 21:1519
\cyrkbeak 21:1519
\CYRKDSC 21:1519
\cyrkdsc 21:1519
\CYRKHCRS 21:1519
\cyrkhcrs 21:1519
\CYRKHK 21:1520
\cyrkhk 21:1519
\CYRKVCRS 21:1520
\cyrkvcrs 21:1520
\CYRL 21:1520
\cyrl 21:1520
\CYRLDSC 21:1520
\cyrldsc 21:1520
\CYRLHK 21:1521
\cyrlhk 21:1520

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\CYRLJE 21:1521
\cyrlje 21:1521
\CYRM 21:1521
\cyrm 21:1521
\CYRMDSC 21:1521
\cyrmdsc 21:1521
\CYRMHK 21:1521
\cyrmhk 21:1521
\CYRN 21:1522
\cyrn 21:1522
\CYRNDSC 21:1522
\cyrndsc 21:1522
\CYRNG 21:1522
\cyrng 21:1522
\CYRNHK 21:1522
\cyrnhk 21:1522
\CYRNJE 21:1523
\cyrnje 21:1522
\CYRNLHK 21:1523
\cyrnlhk 21:1523
\CYRO 21:1523
\cyro 21:1523
\CYROTLD 21:1523
\cyrotld 21:1523
\CYRP 21:1523
\cyrp 21:1523
\CYRPHK 21:1524
\cyrphk 21:1524
\CYRQ 21:1524
\cyrq 21:1524
\CYRR 21:1524
\cyrr 21:1524
\CYRRDSC 21:1524
\cyrrdsc 21:1524
\CYRRHK 21:1525, *1238*
\cyrrhk 21:1524, *1238*
\CYRRHOOK *1238*
\cyrrhook *1238*
\CYRRTICK 21:1525
\cyrrtick 21:1525
\CYRS 21:1525
\cyrs 21:1525
\CYRSACRS 21:1525
\cyrsacrs 21:1525
\CYRSCHWA 21:1526
\cyrschwa 21:1526
\CYRSDSC 21:1526
\cyrsdsc 21:1526
\CYRSEMISFTSN 21:1526
\cyrsemisftsn 21:1526
\CYRSFTSN 21:1527
\cyrsftsn 21:1527
\CYRSH 21:1527
\cyrsh 21:1527
\CYRSHCH 21:1527
\cyrshch 21:1527
\CYRSHHA 21:1527
\cyrshha 21:1527
\CYRT 21:1528
\cyrt 21:1528
\CYRTDSC 21:1528
\cyrtdsc 21:1528
\CYRTETSE 21:1528
\cyrtetse 21:1528
\CYRTSHE 21:1528
\cyrtshe 21:1528
\CYRU 21:1529
\cyru 21:1529
\CYRUSHRT 21:1529
\cyrushrt 21:1529
\CYRV 21:1529
\cyrv 21:1529
\CYRW 21:1529
\cyrw 21:1529
\CYRY 21:1529
\cyry 21:1529
\CYRYA 21:1530
\cyrya 21:1530
\CYRYAT 21:1530
\cyryat 21:1530
\CYRYHCRS 21:1530
\cyryhcrs 21:1530
\CYRYI 21:1530
\cyryi 21:1530
\CYRYO 21:1531
\cyryo 21:1530
\CYRYU 21:1531
\cyryu 21:1531
\CYRZ 21:1531
\cyrz 21:1531
\CYRZDSC 21:1531
\cyrzdsc 21:1531
\CYRZH 21:1531
\cyrzh 21:1531
\CYRZHDSC 21:1532
\cyrzhdsc 21:1532

D

\d 21:235, 21:395, 21:482, 21:765, 21:1249, 21:1463, 21:1464, 21:1465, 21:1466, 21:1469, 21:1470, 21:1471, 21:1472, 21:1473, 21:1474, 21:1475, 21:1476, 21:1477, 21:1478, 21:1479, 21:1480, 21:1481, 21:1482, 21:1483, 21:1484, 21:1485, 21:1486, 21:1487, 21:1488, 21:1489, 21:1490, 21:1491, 21:1492, 21:1493, 21:1494, 21:1495,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

21:1496, 21:1497, 21:1498, 21:1499, 21:1500, 21:1501, 21:1502, *1222*
\dag 21:312, *1218*
\dagger 21:312, 22:213, 22:219, 22:227, 22:228, 30:382
\dashbox 42:309, 42:807, 42:824
\dashv 30:410
\date *898*
\date 44:9, 44:25
\day 01:169, 50:1298, 50:1431, 50:1520, 02:384
\dblfigrule 54:771, 54:2363, *1209*
\dblfloatpagefraction 45:287, 45:301, 54:2340
\dblfloatsep 54:757, 54:769, 54:1644, 54:1770, 54:2347
\dbltextfloatsep 54:224, 54:232, 54:773, 54:1643, 54:1769, 54:2347
\dbltopfraction . . 45:284, 45:298, 54:2339
\ddag 21:313, *1218*
\ddagger 21:313, 22:214, 22:220, 22:227, 22:229, 30:381
\ddot 30:525
\ddots 30:520
\deadcycles 20:335, 20:391, 20:430, 37:33, 37:98, 37:174, 37:240, 54:301, *1234*
debug commands:
\debug_resume: 11:102, 11:118, 11:126, 11:134, *224*
\debug_suspend: 11:98, 11:106, 11:122, 11:130, *224*
\DebugHooksOff 08:2798, 08:2906, *206*
\DebugHooksOn 08:2798, 08:2905, *304*
\DebugMarksOff 48:321, *946*
\DebugMarksOn 48:321, *946*
\DebugShipoutsOff . . 53:414, 53:446, *1064*
\DebugShipoutsOn . . . 53:414, 53:445, *1064*
\DebugSocketsOff 10:170, 10:191, *335*
\DebugSocketsOn 10:170, 10:190, *335*
\DebugTablesOff 55:117
\DebugTablesOn 55:117
declare commands:
declare_callback_rule 04:864
\declare_callback_rule *46*
\DeclareAccent *1202*
\DeclareCaseChangeEquivalent 57:571, *1262*
\DeclareCommandCopy 06:472, 06:473, 06:475, *1252*
\DeclareCurrentRelease . . 33:813, 50:1763
\DeclareDefaultHookRule 08:2802, 08:2909, *211*
\DeclareDocumentCommand 07:2804
\DeclareDocumentEnvironment . . . 07:2840
\DeclareEmphSequence *652*
\DeclareEmphSequence 29:541, 29:577, 29:578, 29:590, *653*
\DeclareEncoding *1202*
\DeclareEncodingSubset 33:40, 33:397, 33:398, 33:399, 33:400, 33:401, 33:402, 33:403, 33:404, 33:405, 33:406, 33:407, 33:408, 33:409, 33:410, 33:411, 33:412, 33:413, 33:414, 33:415, 33:416, 33:417, 33:418, 33:419, 33:420, 33:421, 33:422, 33:423, 33:424, 33:425, 33:426, 33:427, 33:428, 33:429, 33:430, 33:431, 33:432, 33:433, 33:434, 33:435, 33:436, 33:437, 33:438, 33:439, 33:440, 33:441, 33:442, 33:443, 33:444, 33:445, 33:446, 33:447, 33:448, 33:449, 33:450, 33:451, 33:452, 33:453, 33:454, 33:455, 33:456, 33:457, 33:458, 33:459, 33:460, 33:461, 33:462, 33:463, 33:464, 33:465, 33:466, 33:467, 33:468, 33:469, 33:470, 33:471, 33:472, 33:473, 33:474, 33:475, 33:476, 33:477, 33:478, 33:479, 33:480, 33:481, 33:482, 33:483, 33:484, 33:485, 33:486, 33:487, 33:488, 33:489, 33:490, 33:491, 33:492, 33:493, 33:494, 33:495, 33:496, 33:497, 33:498, 33:499, 33:500, 33:501, 33:502, 33:503, 33:504, 33:505, 33:506, 33:507, 33:508, 33:509, 33:510, 33:511, 33:512, 33:513, 33:514, 33:515, 33:516, 33:517, 33:518, 33:519, 33:520, 33:521, 33:522, 33:523, 33:524, 33:525, 33:526, 33:527, 33:528, 33:529, 33:530, 33:531, 33:532, 33:533, 33:534, 33:535, 33:536, 33:537, 33:538, 33:539, 33:540, 33:541, 33:542, 33:543, 33:544, 33:545, 33:546, 33:547, 33:548, 33:549, 33:550, 33:551, 33:552, 33:553, 33:554, 33:555, 33:556, 33:557, 33:558, 33:559, 33:560, 33:561, 33:562, 33:563, 33:564, 33:565, 33:566, 33:567, 33:568, 33:569, 33:570, 33:571, 33:572, 33:573, 33:574, 33:575, 33:620, 33:818, 33:819, 33:820, 33:821, 33:862, 33:869, 33:870, 33:871, 33:872, 33:1148, 33:1149, 33:1150, 33:1151, 33:1152, 33:1153, 33:1154, 33:1155, 33:1156,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

33:1157, 33:1158, 33:1159, 33:1160, 33:1161, 33:1162, 33:1163, 33:1164, 33:1165, 33:1166, 33:1167, 33:1168, 33:1169, 33:1170, 33:1171, 33:1172, 33:1173, 33:1174, 33:1175, 33:1176, 33:1177, 33:1178, 33:1179, 33:1180, 33:1181, 33:1182, 33:1183, 33:1184, 33:1185, 33:1186, 33:1187, 33:1188, 33:1189, 33:1190, 33:1191, 33:1192, 33:1193, 33:1194, 33:1195, 33:1196, 33:1197, 33:1198, 33:1199, 33:1200, 33:1201, 33:1202, 33:1203, 33:1204, 33:1205, 33:1206, 33:1207, 33:1208

`\DeclareEncodingsubset` *727*

`\DeclareEnvironmentCopy` 06:513, 06:514, 06:516, *1264*

`\DeclareErrorFont` 24:502, 28:383, 29:735, 30:57

`\DeclareExpandableDocumentCommand` 07:2923

`\DeclareFixedFont` 24:75, 33:1249, 33:1251, *728*

`\DeclareFontEncoding` 21:377, 21:463, 21:716, 21:738, 21:744, 21:830, 21:1038, 24:118, 30:133, 30:134, 30:135, 30:136, *1202*

`\DeclareFontEncodingDefaults` 24:168, 27:90, 27:91, 30:36

`\DeclareFontFamily` 24:93, 27:85, 27:86, *539*

`\DeclareFontFamilySubstitution` . 24:462

`\DeclareFontSeriesChangeRule` . . 25:3, 25:4, 25:6, 25:7, 25:8, 25:9, 25:10, 25:11, 25:12, 25:13, 25:14, 25:15, 25:16, 25:17, 25:18, 25:19, 25:20, 25:21, 25:22, 25:23, 25:24, 25:25, 25:26, 25:27, 25:28, 25:29, 25:30, 25:31, 25:32, 25:33, 25:34, 25:35, 25:36, 25:37, 25:38, 25:39, 25:40, 25:41, 25:42, 25:43, 25:44, 25:45, 25:46, 25:47, 25:48, 25:49, 25:50, 25:51, 25:52, 25:53, 25:54, 25:55, 25:56, 25:57, 25:58, 25:59, 25:60, 25:61, 25:62, 25:63, 25:64, 25:65, 25:66, 25:67, 25:68, 25:69, 25:70, 25:71, 25:72, 25:73, 25:74, 25:75, 25:76, 25:77, 25:78, 25:79, 25:80, 25:81, 25:82, 25:83, 25:84, 25:85, 25:86, 25:87, 25:88, 25:89, 25:90, 25:91, 25:92, 25:93, 25:94, 25:95, 25:96, 25:97, 25:98, 25:99, 25:100, 25:101, 25:102, 25:103, 25:104, 25:105, 25:106, 25:107, 25:108, 25:109, 25:110, 25:111, 25:112, 25:113, 25:114, 25:115, 25:116, 25:117, 25:118, 25:119, 25:120, 25:121, 25:122, 25:123, 25:124, 25:125, 25:126, 25:127, 25:128, 25:129, 25:130, 25:131, 25:132, 25:133, 25:134, 25:135, 25:136, 25:137, 25:138, 25:139, 25:140, 25:141, 25:142, 25:143, 25:144, 25:145, 25:146, 25:147, 25:148, 25:149, 25:150, 25:151, 25:152, 25:153, 25:154, 25:155, 25:156, 25:157, 25:158, 25:159, 25:160, 25:161, 25:162, 25:163, 25:164, 25:165, 25:166, 25:167, 25:168, 25:169, 25:170, 25:171, 25:172, 25:173, 25:174, 25:175, 25:176, 25:177, 25:178, 25:179, 25:180, 25:181, 25:182, 25:183, 25:184, 25:185, 25:186, 25:187, 25:188, 25:189, 25:190, 25:191, 25:192, 25:193, 25:194, 25:195, 25:196, 25:197, 25:198, 25:199, 25:200, 25:201, 25:202, 25:203, 25:204, 25:205, 25:206, 25:207, 25:208, 25:209, 25:210, 25:211, 25:212, 25:213, 25:214, 25:215, 25:216, 25:217, 25:218, 25:219, 25:220, 25:221, 25:222, 25:223, 25:224, 25:225, 25:226, 25:227, 25:228, 25:229, 25:230, 25:231, 25:232, 25:233, 25:234, 25:235, 25:236, 25:237, 25:238, 25:239, 25:240, 25:241, 25:242, 25:243, 25:244, 25:245, 25:246, 25:247, 25:248, 25:249, 25:250, 25:251, 25:252, 25:253, 25:254, 25:255, 25:256, 25:257, 25:258, 25:259, 25:260, 25:261, 25:262, 25:263, 25:264, 25:265, 25:266, 25:267, 25:268, 25:269, 25:270, 25:271, 25:272, 25:273, 25:274, 25:275, 25:276, 25:277, 25:278, 25:279, 25:280, 25:281, 25:282, 25:283, 25:284, 25:285, 25:286, 25:287, 25:288, 25:289, 25:290, 25:291, 25:292, 25:293, 25:294, 25:295, 25:296, 25:297, 25:298, 25:299, 25:300, 25:301, 25:302, 25:303, 25:304, 25:305, 25:306, 25:307, 25:308, 25:309, 25:310, 25:311, 25:312, 25:313, 25:314, 25:315, 25:316, 25:317, 25:318, 25:319, 25:320, 25:321, 25:322, 25:323, 25:324, 25:325, 25:326, 25:327, 25:328, 25:329, 25:330, 25:331, 25:332,

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

25:333, 25:334, 25:335, 25:336, 25:337, 25:338, 25:339, 25:340, 25:341, 25:342, 25:343, 25:344, 25:345, 25:346, 25:347, 25:348, 25:349, 25:350, 25:351, 25:352, 25:353, 25:354, 25:355, 25:356, 25:357, 25:358, 25:359, 25:360, 25:361, 25:362, 25:363, 25:364, 25:365, 25:366, 25:367, 25:368, 25:369, 25:370, 25:371, 25:372, 25:373, 25:374, 25:375, 25:376, 25:377, 25:378, 25:379, 25:380, 25:381, 25:382, 25:386, 25:388, *551*
`\DeclareFontSeriesDefault` *635*
`\DeclareFontSeriesDefault` 29:34, 29:35, 29:69, 29:71, 29:72, 29:82, 29:93, 29:102, 29:104, *642*
`\DeclareFontShape` 24:18, 24:471, 24:472, 24:473, 24:474, 24:475, 24:476, 24:477, 24:478, 24:479, 24:480, 24:481, 24:482, 24:483, 24:484, 24:485, 24:486, 24:487, 24:488, 24:489, 24:490, 24:491, 27:25, 27:27, 27:81, 27:82, *1196*
`\DeclareFontShapeChangeRule` 25:523, 25:524, 25:541, 25:542, 25:543, 25:544, 25:545, 25:546, 25:547, 25:548, 25:549, 25:550, 25:551, 25:552, 25:553, 25:554, 25:555, 25:556, 25:557, 25:558, 25:559, 25:560, 25:561, 25:562, 25:563, 25:564, 25:565, 25:566, 25:567, 25:568, 25:569, 25:570, 25:571, 25:572, 25:573, 25:574, 25:575, 25:576, 25:577, 25:578, 25:579, 25:580, 25:581, 25:582, 25:583, 25:584, 25:585, 25:586, 25:587, 25:588, 25:589, 25:590, 25:591, 25:592, 25:593, 25:594, 25:595, 25:596, 25:597, 25:601, 25:603
`\DeclareFontSubstitution` 21:745, 21:831, 24:141, 30:26, 30:34, 30:37, 30:38, 30:137, 30:138, 30:139, 30:140
`\DeclareHookRule` . . . 08:2800, 08:2908, 37:50, 37:51, 37:115, 37:116, *204*
`\DeclareInstance` 11:1156, *348*
`\DeclareInstanceCopy` 11:1156, *348*
`\DeclareKeys` 51:184, *1027*
`\DeclareLowercaseMapping` . . . 57:571, *1266*
`\DeclareMathAccent` 28:788, 30:523, 30:524, 30:525, 30:526, 30:527, 30:528, 30:529, 30:530, 30:531, 30:532, 30:533, 30:534, 30:535
`\DeclareMathAlphabet` 27:119, 27:123, 27:125, 27:132, 28:614, 28:777, 30:158, 30:159, 30:160, 30:161
`\DeclareMathAlphabetCharacter` . . 28:963
`\DeclareMathDelimiter` 28:965, 30:262, 30:263, 30:264, 30:265, 30:266, 30:267, 30:270, 30:272, 30:273, 30:570, 30:572, 30:574, 30:576, 30:578, 30:581, 30:583, 30:585, 30:587, 30:589, 30:591, 30:593, 30:595, 30:597, 30:599, 30:601, 30:603, 30:605, 30:607, 30:609, 30:611, 30:613, 30:615, 30:617, 30:619, *679*
`\DeclareMathOperator` *1222*
`\DeclareMathRadical` 28:1100, 30:536
`\DeclareMathSizes` 24:205, 24:211, 24:233, 30:164, 30:165, 30:166, 30:167, 30:168, 30:169, 30:170, 30:171, 30:172, 30:173, 30:174, 30:175, *1220*
`\DeclareMathSizes*` 24:205
`\DeclareMathSymbol` 28:901, 28:964, 28:981, 30:176, 30:177, 30:178, 30:179, 30:180, 30:181, 30:182, 30:183, 30:184, 30:185, 30:186, 30:187, 30:188, 30:189, 30:190, 30:191, 30:192, 30:193, 30:194, 30:195, 30:196, 30:197, 30:198, 30:199, 30:200, 30:201, 30:202, 30:203, 30:204, 30:205, 30:206, 30:207, 30:208, 30:209, 30:210, 30:211, 30:212, 30:213, 30:214, 30:215, 30:216, 30:217, 30:218, 30:219, 30:220, 30:221, 30:222, 30:223, 30:224, 30:225, 30:226, 30:227, 30:228, 30:229, 30:230, 30:231, 30:232, 30:233, 30:234, 30:235, 30:236, 30:237, 30:238, 30:239, 30:240, 30:241, 30:242, 30:243, 30:244, 30:245, 30:246, 30:247, 30:248, 30:249, 30:250, 30:251, 30:252, 30:253, 30:254, 30:255, 30:256, 30:257, 30:258, 30:268, 30:269, 30:271, 30:275, 30:276, 30:277, 30:278, 30:279, 30:280, 30:281, 30:282, 30:283, 30:284, 30:285, 30:286, 30:287, 30:288, 30:289, 30:290, 30:291, 30:292, 30:293, 30:294, 30:295, 30:296, 30:297, 30:298, 30:299, 30:300, 30:301, 30:302, 30:303, 30:304, 30:305, 30:306, 30:307, 30:308, 30:309, 30:310, 30:311, 30:312, 30:313,

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

30:314, 30:315, 30:316, 30:317, 30:318, 30:319, 30:320, 30:321, 30:322, 30:323, 30:324, 30:325, 30:326, 30:327, 30:328, 30:329, 30:330, 30:331, 30:332, 30:334, 30:335, 30:336, 30:337, 30:338, 30:339, 30:340, 30:341, 30:348, 30:349, 30:350, 30:351, 30:352, 30:353, 30:354, 30:356, 30:357, 30:358, 30:359, 30:360, 30:361, 30:363, 30:364, 30:365, 30:366, 30:367, 30:368, 30:371, 30:372, 30:373, 30:374, 30:377, 30:378, 30:379, 30:380, 30:381, 30:382, 30:383, 30:384, 30:385, 30:386, 30:387, 30:388, 30:389, 30:390, 30:391, 30:392, 30:393, 30:394, 30:395, 30:396, 30:397, 30:398, 30:399, 30:400, 30:401, 30:402, 30:403, 30:404, 30:405, 30:406, 30:407, 30:408, 30:409, 30:410, 30:411, 30:412, 30:413, 30:414, 30:415, 30:416, 30:417, 30:418, 30:421, 30:422, 30:425, 30:426, 30:427, 30:428, 30:429, 30:430, 30:431, 30:432, 30:433, 30:434, 30:435, 30:436, 30:437, 30:439, 30:440, 30:441, 30:442, 30:443, 30:444, 30:445, 30:448, 30:449, 30:450, 30:452, 30:453, 30:454, 30:455, 30:456, 30:457, 30:458, 30:459, 30:460, 30:461, 30:462, 30:484, 30:486, 30:508, 30:509, 30:510, 30:560, 30:561, 30:562, 30:563, 30:621, 30:622, 30:623, *675*

`\DeclareMathVersion` . . . 28:396, 29:2, 29:3

`\DeclareOldFontCommand` . . 32:125, 32:141

`\DeclareOption` *977*

`\DeclareOption` 21:1535, 26:29, 26:37, 26:45, 26:53, 26:56, 26:60, 33:818, 33:819, 33:820, 33:821, 33:822, 33:824, 33:826, 33:828, 33:829, 33:869, 33:870, 33:871, 33:872, 33:873, 33:875, 33:876, 50:496, 50:1193, *1224*

`\DeclareOption*` *977*

`\DeclareOption*` 50:496

`\DeclarePreloadSizes` 24:185, 27:95, 27:96, 31:19, 31:21, 31:22, 31:23, 31:25, 31:26, 31:27, 31:28, 31:29, 31:30, 31:34, 31:38, 31:43, 31:45, 31:49, 31:50, 31:53, 31:54, 31:57, 31:58, 31:64

`\DeclareProtectedCommand` *1211*

`\DeclareRelease` . . 33:811, 33:812, 50:1669

DeclareRelease commands:

`\DeclareRelease:` 50:1672

`\DeclareRobustCommand` 14:4, 14:11, 14:30, 14:57, 18:7, 18:8, 18:9, 18:10, 18:11, 18:69, 18:93, 18:377, 18:453, 18:467, 18:512, 18:537, 19:2, 19:3, 19:13, 20:469, 20:616, 21:150, 21:158, 21:307, 21:310, 21:311, 21:312, 21:313, 21:314, 21:316, 21:318, 21:320, 22:237, 23:32, 23:33, 23:34, 24:251, 24:279, 24:280, 24:281, 24:286, 24:298, 24:308, 24:316, 24:318, 24:336, 24:666, 24:675, 25:394, 25:398, 25:407, 25:409, 25:419, 25:526, 25:531, 25:536, 25:615, 25:618, 25:626, 25:627, 25:633, 25:713, 26:115, 26:164, 29:4, 29:7, 29:10, 29:13, 29:16, 29:19, 29:22, 29:25, 29:28, 29:254, 29:277, 29:307, 29:328, 29:352, 29:363, 29:380, 29:383, 29:405, 29:410, 29:415, 29:448, 29:453, 29:458, 29:474, 29:477, 29:480, 29:483, 29:486, 29:500, 29:549, 29:550, 29:585, 29:591, 29:613, 29:615, 29:624, 29:626, 29:633, 29:649, 29:657, 29:675, 29:691, 29:708, 30:342, 30:343, 30:344, 30:355, 30:362, 30:419, 30:420, 30:451, 30:463, 30:467, 30:470, 30:475, 30:477, 30:479, 30:482, 30:485, 30:487, 30:488, 30:490, 30:492, 30:494, 30:496, 30:498, 30:500, 30:502, 30:504, 30:506, 30:512, 30:514, 30:516, 30:519, 30:537, 30:540, 30:543, 30:547, 30:551, 30:554, 30:557, 30:564, 30:567, 30:624, 30:625, 30:626, 30:631, 30:633, 30:635, 30:637, 32:3, 32:126, 33:4, 33:11, 33:610, 33:1133, 35:178, 35:189, 37:269, 37:450, 37:455, 37:460, 37:470, 37:474, 37:478, 37:576, 37:580, 38:3, 38:4, 38:5, 38:6, 38:7, 38:8, 38:9, 38:10, 38:11, 38:12, 38:13, 38:14, 38:15, 38:16, 38:17, 38:18, 38:19, 38:20, 38:21, 38:22, 38:23, 38:24, 38:25, 38:26, 38:27, 38:28, 38:29, 38:30, 38:31, 38:32, 38:33, 38:34, 38:35, 38:39, 38:41, 38:42, 38:43, 38:44, 38:45, 38:46, 38:47, 38:48, 38:49, 38:50, 38:51, 38:52, 38:81, 38:82, 38:83, 38:84,

38:126, 38:167, 38:169, 38:173, 38:218, 38:220, 38:222, 38:224, 38:227, 38:228, 38:230, 38:237, 38:239, 38:240, 38:251, 38:274, 38:276, 38:292, 38:303, 38:353, 38:354, 38:355, 38:429, 38:463, 38:487, 40:7, 40:24, 40:159, 40:172, 40:195, 40:196, 40:212, 40:223, 40:302, 40:534, 40:552, 40:560, 40:615, 40:616, 40:617, 40:618, 40:619, 40:620, 41:139, 41:142, 41:154, 42:124, 42:127, 42:130, 44:7, 44:8, 44:9, 44:10, 44:14, 44:222, 06:233, 45:402, 45:423, 47:16, 47:28, 48:482, 48:483, 49:23, 49:34, 49:51, 49:59, 49:82, 49:84, 49:86, 49:93, 50:1128, 50:1129, 50:1130, 50:1136, 50:1137, 50:1793, 52:148, 52:184, 53:448, 54:87, 06:833, 06:834, 06:840, 06:855, *104*

`\DeclareSizeFunction` 26:417, 26:490, 26:491, 26:502, 26:503, 26:507, 26:508, 26:514, 26:515, 26:543, 26:557, 26:558, 26:565, 26:566

`\DeclareSymbolFont` 27:136, 28:455, 30:148, 30:149, 30:150, 30:151

`\DeclareSymbolFontAlphabet` 28:1174, 30:155, 30:156, 30:157

`\DeclareTemplateCode` 11:994, 11:1035, 11:1061, 11:1156, *346*

`\DeclareTemplateCopy` 11:1156, *345*

`\DeclareTemplateInterface` .. 11:1156, *344*

`\DeclareTextAccent` 21:64, 21:378, 21:379, 21:380, 21:381, 21:382, 21:383, 21:384, 21:385, 21:386, 21:387, 21:388, 21:464, 21:465, 21:466, 21:467, 21:468, 21:469, 21:470, 21:471, 21:472, 21:473, 21:474, 21:741, 21:746, 21:747, 21:748, 21:749, 21:750, 21:751, 21:752, 21:753, 21:754, 21:755, 21:756, 21:838, 21:839, 21:840, 21:841, 21:842, 21:843, 21:844, 21:845, 21:846, 21:847, 21:848, 21:849, 21:850, 21:851, 21:852

`\DeclareTextAccentDefault` 21:185, 21:226, 21:227, 21:228, 21:229, 21:230, 21:231, 21:232, 21:233, 21:234, 21:235, 21:236, 21:237, 21:238, 21:239, 21:279, 21:282, 33:687, 33:688, 33:937, 33:938, 33:939, 33:940, 33:941, 33:942, 33:943, 33:944, 33:945, 33:946, 33:947, 33:948, 33:949, *1217*

`\DeclareTextCommand` 21:3, 21:58, 21:65, 21:83, 21:389, 21:392, 21:395, 21:409, 21:410, 21:411, 21:414, 21:415, 21:422, 21:424, 21:426, 21:428, 21:434, 21:436, 21:438, 21:445, 21:475, 21:478, 21:482, 21:485, 21:487, 21:490, 21:492, 21:494, 21:510, 21:568, 21:569, 21:570, 21:730, 21:757, 21:759, 21:762, 21:765, 21:798, 21:805, 21:832, 21:835, 21:865, 21:893, 21:1053, 21:1080, 33:376, 33:377, 33:378, 33:379, 33:380, 33:381, 33:382, 33:383, 33:384, 33:385, 33:625, 33:632, 33:639, 33:759, *1218*

`\DeclareTextCommandDefault` ... 21:57, 21:186, 21:188, 21:283, 21:286, 21:287, 21:288, 21:290, 21:292, 21:296, 21:300, 21:301, 21:303, 21:304, 21:305, 21:306, 21:326, 21:355, 33:155, 33:157, 33:160, 33:162, 33:164, 33:166, 33:168, 33:170, 33:172, 33:174, 33:176, 33:178, 33:180, 33:182, 33:184, 33:186, 33:188, 33:190, 33:193, 33:194, 33:195, 33:196, 33:197, 33:198, 33:199, 33:200, 33:201, 33:202, 33:203, 33:204, 33:205, 33:206, 33:207, 33:208, 33:210, 33:212, 33:214, 33:216, 33:218, 33:220, 33:222, 33:224, 33:226, 33:228, 33:230, 33:232, 33:234, 33:236, 33:238, 33:240, 33:242, 33:244, 33:246, 33:248, 33:250, 33:252, 33:254, 33:256, 33:258, 33:260, 33:262, 33:264, 33:266, 33:268, 33:270, 33:272, 33:274, 33:276, 33:278, 33:280, 33:282, 33:284, 33:286, 33:288, 33:290, 33:292, 33:294, 33:296, 33:298, 33:300, 33:302, 33:304, 33:306, 33:309, 33:311, 33:313, 33:315, 33:317, 33:319, 33:321, 33:323, 33:325, 33:327, 33:329, 33:331, 33:333, 33:335, 33:337, 33:339, 33:341, 33:343, 33:345, 33:347, 33:349, 33:351, 33:353, 33:355, 33:357, 33:359, 33:361, 33:363, 33:365, 33:654, 33:662, 33:664, 33:670, 33:678, 33:1003, 33:1005, 33:1006, 33:1008, 33:1010, 33:1012, 33:1014, 33:1016, 33:1018, 33:1020, 33:1022, 33:1024, 33:1026, 33:1028, 33:1030, 33:1032, 33:1034, 33:1036,

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

33:1038, 33:1040, 33:1042, 33:1044, 33:1046, 33:1048, 33:1050, 33:1052, 33:1054, 33:1056, 33:1058, 33:1060, 33:1062, 33:1064, 33:1066, 33:1068, 33:1070, 33:1072, 33:1074, 33:1076, 33:1078, 33:1080, 33:1082, 33:1084, 33:1086, 33:1088, 33:1090, 33:1092, 33:1094, 33:1096, 33:1098, 33:1100, 33:1102, 33:1104, 33:1106, 33:1108, 33:1110, 33:1112, 33:1114, 33:1116, 33:1118, 33:1120, 33:1123, *1217*

`\DeclareTextComposite` . 21:76, 21:452, 21:453, 21:587, 21:588, 21:589, 21:590, 21:591, 21:592, 21:593, 21:594, 21:595, 21:596, 21:597, 21:598, 21:599, 21:600, 21:601, 21:602, 21:603, 21:604, 21:605, 21:606, 21:607, 21:608, 21:609, 21:610, 21:611, 21:612, 21:613, 21:614, 21:615, 21:616, 21:617, 21:618, 21:619, 21:620, 21:621, 21:622, 21:623, 21:624, 21:625, 21:626, 21:627, 21:628, 21:629, 21:630, 21:631, 21:632, 21:633, 21:634, 21:635, 21:636, 21:637, 21:638, 21:639, 21:640, 21:641, 21:642, 21:643, 21:644, 21:645, 21:646, 21:647, 21:648, 21:649, 21:650, 21:651, 21:652, 21:653, 21:654, 21:655, 21:656, 21:657, 21:658, 21:659, 21:660, 21:661, 21:662, 21:663, 21:664, 21:665, 21:666, 21:667, 21:668, 21:669, 21:670, 21:671, 21:672, 21:673, 21:674, 21:675, 21:676, 21:677, 21:678, 21:679, 21:680, 21:681, 21:682, 21:683, 21:684, 21:685, 21:686, 21:687, 21:688, 21:689, 21:690, 21:691, 21:692, 21:693, 21:694, 21:695, 21:696, 21:697, 21:812, 21:813, 21:814, 21:815, 21:816, 21:817, 21:818, 21:819, 21:820, 21:821, 21:822, 21:823, 21:824, 21:825, 21:826, 21:827, *1226*

`\DeclareTextCompositeCommand` . 21:76, 21:431, 21:454, 21:455, 21:456, 21:457, 21:459, 21:698, 21:699, 21:701, 21:704, 21:705, 21:706, 21:707, 21:708, 21:709, 21:710, 21:711, 21:712, 21:795, 21:1059, *1218*

`\DeclareTextFontCommand` 32:1, 32:15, 32:16, 32:17, 32:18, 32:19, 32:20, 32:21, 32:22, 32:23, 32:24, 32:29, 32:30, 32:31, 32:42, 32:140, *1235*

`\DeclareTextFoo` *1202*

`\DeclareTextGlyph` *1209*

`\DeclareTextSymbol` 21:3, 21:398, 21:399, 21:400, 21:401, 21:402, 21:403, 21:404, 21:405, 21:406, 21:407, 21:408, 21:412, 21:413, 21:416, 21:417, 21:418, 21:419, 21:420, 21:421, 21:526, 21:527, 21:528, 21:529, 21:530, 21:531, 21:532, 21:533, 21:534, 21:535, 21:536, 21:537, 21:538, 21:539, 21:540, 21:541, 21:543, 21:544, 21:545, 21:546, 21:547, 21:548, 21:549, 21:550, 21:551, 21:552, 21:553, 21:554, 21:555, 21:556, 21:557, 21:558, 21:559, 21:560, 21:561, 21:562, 21:563, 21:564, 21:565, 21:566, 21:567, 21:571, 21:572, 21:573, 21:574, 21:575, 21:576, 21:577, 21:578, 21:579, 21:580, 21:581, 21:582, 21:583, 21:584, 21:585, 21:586, 21:717, 21:718, 21:719, 21:720, 21:721, 21:722, 21:723, 21:724, 21:725, 21:726, 21:727, 21:728, 21:729, 21:739, 21:740, 21:768, 21:769, 21:770, 21:771, 21:772, 21:773, 21:774, 21:776, 21:777, 21:778, 21:779, 21:780, 21:781, 21:782, 21:783, 21:784, 21:785, 21:786, 21:787, 21:788, 21:789, 21:790, 21:791, 21:792, 21:793, 21:794, 21:853, 21:854, 21:855, 21:856, 21:857, 21:858, 21:859, 21:860, 21:861, 21:862, 21:863, 21:864, 21:876, 21:877, 21:878, 21:879, 21:880, 21:881, 21:882, 21:883, 21:884, 21:885, 21:886, 21:887, 21:888, 21:889, 21:890, 21:891, 21:892, 21:899, 21:900, 21:901, 21:902, 21:903, 21:904, 21:905, 21:906, 21:907, 21:908, 21:909, 21:910, 21:911, 21:912, 21:913, 21:914, 21:915, 21:916, 21:917, 21:918, 21:919, 21:920, 21:921, 21:922, 21:923, 21:924, 21:925, 21:926, 21:927, 21:928, 21:929, 21:930, 21:931, 21:932, 21:933, 21:934, 21:935, 21:936, 21:937, 21:938, 21:939, 21:940, 21:941, 21:942, 21:943, 21:944, 21:945, 21:946, 21:947, 21:948, 21:949, 21:950, 21:951, 21:952, 21:953, 21:954, 21:955, 21:956, 21:957,

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

21:958, 21:959, 21:960, 21:961, 21:962, 21:963, 21:964, 21:965, 21:966, 21:967, 21:968, 21:969, 21:970, 21:971, 21:972, 21:973, 21:974, 21:975, 21:976, 21:977, 21:978, 21:1079, 33:368, 33:369, 33:370, 33:371, 33:372, 33:373, 33:374, 33:375, 33:386, 33:387, 33:388, 33:389, 33:390, 33:391, 33:392, 33:393, 33:394, 33:395, 33:589, 33:590, 33:591, 33:592, 33:593, 33:594, 33:595, 33:596, *1218*

`\DeclareTextSymbolDefault` 21:185, 21:240, 21:241, 21:242, 21:243, 21:244, 21:245, 21:246, 21:247, 21:248, 21:249, 21:250, 21:251, 21:252, 21:253, 21:254, 21:255, 21:256, 21:257, 21:258, 21:259, 21:260, 21:261, 21:262, 21:263, 21:264, 21:265, 21:266, 21:267, 21:268, 21:269, 21:270, 21:271, 21:272, 21:273, 21:274, 21:275, 21:276, 21:277, 21:278, 21:280, 21:281, 21:291, 33:114, 33:116, 33:118, 33:120, 33:121, 33:122, 33:123, 33:124, 33:125, 33:126, 33:127, 33:128, 33:129, 33:130, 33:131, 33:132, 33:133, 33:134, 33:135, 33:136, 33:137, 33:138, 33:139, 33:140, 33:141, 33:142, 33:143, 33:144, 33:145, 33:146, 33:147, 33:148, 33:149, 33:150, 33:151, 33:152, 33:153, 33:154, 33:577, 33:578, 33:579, 33:580, 33:581, 33:582, 33:583, 33:584, 33:597, 33:598, 33:599, 33:600, 33:601, 33:602, 33:603, 33:604, 33:623, 33:624, 33:642, 33:643, 33:644, 33:645, 33:646, 33:647, 33:648, 33:650, 33:950, 33:951, 33:952, 33:953, 33:954, 33:955, 33:956, 33:957, 33:958, 33:959, 33:960, 33:961, 33:962, 33:963, 33:964, 33:965, 33:966, 33:967, 33:968, 33:969, 33:970, 33:971, 33:972, 33:973, 33:974, 33:975, 33:976, 33:977, 33:978, 33:979, 33:980, 33:981, 33:982, 33:983, 33:984, 33:985, 33:986, 33:987, 33:988, 33:989, 33:990, 33:991, 33:992, 33:993, 33:994, 33:995, 33:996, 33:997, 33:998, 33:999, 33:1000, 33:1001, 33:1002, *1217*

`\DeclareTitlecaseMapping` . . . 57:571, *1266*

`\DeclareUnicode...` *500*

`\DeclareUnicodeAccent` 21:1042, 21:1237, 21:1238, 21:1239, 21:1240, 21:1241, 21:1242, 21:1243, 21:1244, 21:1245, 21:1246, 21:1247, 21:1248, 21:1249, 21:1250, 21:1251, *500*

`\DeclareUnicodeCharacter` . . . 57:384, *1247*

`\DeclareUnicodeCommand` 21:1080, 21:1081, 21:1082, 21:1083, 21:1164, 21:1166, 21:1168, 21:1215, 21:1252, *1260*

`\DeclareUnicodeComposite` 21:1057, 21:1256, 21:1257, 21:1258, 21:1259, 21:1260, 21:1261, 21:1262, 21:1263, 21:1264, 21:1265, 21:1266, 21:1267, 21:1268, 21:1269, 21:1270, 21:1271, 21:1272, 21:1273, 21:1274, 21:1275, 21:1276, 21:1277, 21:1278, 21:1279, 21:1280, 21:1281, 21:1282, 21:1283, 21:1284, 21:1285, 21:1286, 21:1287, 21:1288, 21:1289, 21:1290, 21:1291, 21:1292, 21:1293, 21:1294, 21:1295, 21:1296, 21:1297, 21:1298, 21:1299, 21:1300, 21:1301, 21:1302, 21:1303, 21:1304, 21:1305, 21:1306, 21:1307, 21:1308, 21:1309, 21:1310, 21:1311, 21:1312, 21:1313, 21:1314, 21:1315, 21:1316, 21:1317, 21:1318, 21:1319, 21:1320, 21:1321, 21:1322, 21:1323, 21:1324, 21:1325, 21:1326, 21:1327, 21:1328, 21:1329, 21:1330, 21:1331, 21:1332, 21:1333, 21:1334, 21:1335, 21:1336, 21:1337, 21:1338, 21:1339, 21:1340, 21:1341, 21:1342, 21:1343, 21:1344, 21:1345, 21:1346, 21:1347, 21:1348, 21:1349, 21:1350, 21:1351, 21:1352, 21:1353, 21:1354, 21:1355, 21:1356, 21:1357, 21:1358, 21:1359, 21:1360, 21:1361, 21:1362, 21:1363, 21:1364, 21:1365, 21:1366, 21:1367, 21:1368, 21:1369, 21:1370, 21:1371, 21:1372, 21:1373, 21:1374, 21:1375, 21:1376, 21:1377, 21:1378, 21:1379, 21:1380, 21:1381, 21:1382, 21:1383, 21:1384, 21:1385, 21:1386, 21:1387, 21:1388, 21:1389, 21:1390, 21:1391, 21:1392, 21:1393, 21:1394, 21:1395, 21:1396, 21:1397, 21:1398, 21:1399, 21:1400, 21:1401, 21:1402, 21:1403, 21:1404, 21:1405, 21:1406, 21:1407, 21:1408, 21:1409, 21:1410, 21:1411, 21:1412, 21:1413, 21:1414, 21:1415, 21:1416, 21:1417, 21:1418, 21:1419, 21:1420, 21:1421, 21:1422,

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

21:1423, 21:1424, 21:1425, 21:1426, 21:1427, 21:1428, 21:1429, 21:1430, 21:1431, 21:1432, 21:1433, 21:1434, 21:1435, 21:1436, 21:1437, 21:1438, 21:1439, 21:1440, 21:1441, 21:1442, 21:1443, 21:1444, 21:1445, 21:1446, 21:1447, 21:1448, 21:1449, 21:1450, 21:1451, 21:1452, 21:1453, 21:1454, 21:1455, 21:1456, 21:1457, 21:1458, 21:1459, 21:1460, 21:1461, 21:1462, 21:1463, 21:1464, 21:1465, 21:1466, 21:1467, 21:1468, 21:1469, 21:1470, 21:1471, 21:1472, 21:1473, 21:1474, 21:1475, 21:1476, 21:1477, 21:1478, 21:1479, 21:1480, 21:1481, 21:1482, 21:1483, 21:1484, 21:1485, 21:1486, 21:1487, 21:1488, 21:1489, 21:1490, 21:1491, 21:1492, 21:1493, 21:1494, 21:1495, 21:1496, 21:1497, 21:1498, 21:1499, 21:1500, 21:1501, 21:1502, *500*

`\DeclareUnicodeSymbol` 21:1079, 21:1084, 21:1085, 21:1086, 21:1087, 21:1088, 21:1089, 21:1090, 21:1091, 21:1092, 21:1093, 21:1094, 21:1095, 21:1096, 21:1097, 21:1098, 21:1099, 21:1100, 21:1101, 21:1102, 21:1103, 21:1104, 21:1106, 21:1107, 21:1108, 21:1109, 21:1110, 21:1111, 21:1112, 21:1113, 21:1114, 21:1115, 21:1116, 21:1117, 21:1118, 21:1119, 21:1120, 21:1122, 21:1123, 21:1124, 21:1125, 21:1126, 21:1127, 21:1128, 21:1129, 21:1130, 21:1131, 21:1132, 21:1133, 21:1134, 21:1135, 21:1136, 21:1137, 21:1138, 21:1139, 21:1140, 21:1141, 21:1142, 21:1143, 21:1144, 21:1145, 21:1146, 21:1147, 21:1148, 21:1149, 21:1150, 21:1151, 21:1152, 21:1153, 21:1154, 21:1155, 21:1156, 21:1157, 21:1158, 21:1159, 21:1160, 21:1161, 21:1162, 21:1163, 21:1170, 21:1171, 21:1172, 21:1173, 21:1174, 21:1175, 21:1176, 21:1177, 21:1178, 21:1179, 21:1180, 21:1181, 21:1182, 21:1183, 21:1184, 21:1185, 21:1186, 21:1187, 21:1188, 21:1189, 21:1190, 21:1191, 21:1192, 21:1193, 21:1194, 21:1195, 21:1196, 21:1197, 21:1198, 21:1199, 21:1200, 21:1201, 21:1202, 21:1203, 21:1204, 21:1205, 21:1206, 21:1207, 21:1208, 21:1209, 21:1210, 21:1211, 21:1213, 21:1214, 21:1226, 21:1227, 21:1228, 21:1229, 21:1230, 21:1231, 21:1232, 21:1233, 21:1234, 21:1235, 21:1236, *1260*

`\DeclareUnknownKeyHandler` . . 51:186, *1028*

`\DeclareUnknownKeysHandler` *1261*

`\DeclareUppercaseMapping` . . . 57:571, *1266*

`\def` *1001*

`default` (plug) 55:40

`\defaultencoding` *1214*

`\defaulthyphenchar` . 06:843, 06:858, 02:378

`\defaultscriptratio` . 24:649, 24:656, *1207*

`\defaultscriptscriptratio` 24:650, 24:656, *1207*

`\defaultskewchar` 02:379

`\deg` 38:34

`\delcode` 28:1098

`\delimiter` 28:1011, 28:1081, 28:1092, *1245*

`\delimiterfactor` 02:382

`\delimitershortfall` 02:397

`\Delta` 30:305

`\delta` 30:278

`\depth` 40:32, 40:35

`\det` 38:30

`\Details` 33:1625, 33:1626

`\detokenize` 03:96, 03:97, 03:182, 14:249, 14:250, 20:254, 20:505, 21:998, 21:1040, 25:434, 29:563, 33:44, 37:205, 50:564, 50:598, 50:638, 50:902, 50:1269, 50:1273, 50:1401, 50:1402, 50:1406, 06:574, 06:592, 06:656, *109*

`\DH` 21:528, 21:1128, 57:650, *1213*

`\dh` 21:538, 21:1134, 57:650, *1213*

`\Diamond` 29:727

`\diamond` 30:387

`\diamondsuit` 30:339

`\dim` 38:28

`dim` commands:

- `\dim_eval:n` 11:364, 05:164
- `\dim_new:N` 11:33, 53:199, 53:200, 53:201, 53:202
- `\dim_set:Nn` 53:193, 53:194, 53:195, 53:196
- `\dim_set_eq:NN` 48:67, 48:69
- `\dim_use:N` 53:511, 53:512, 53:513
- `\c_max_dim` 48:67, 48:69, 48:99, 53:213, 53:239, 53:264, 53:291, *955*
- `\l_tmpa_dim` *346*
- `\c_zero_dim` 16:102, 53:219, 53:220, 53:221, 53:227, 53:245, 53:246, 53:247, 53:270, 53:271, 53:272, 53:299, 53:300, 53:301, 53:329, 53:331, 53:332

`\dimen` *1232*

`\dimendef` 04:226, 02:42, 02:43, 02:44, 02:52, 02:91

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

\dimenzero 04:226
\dimeval *77*
\dimeval 05:158, 05:174, *1260*
\dimexpr 21:503, 21:519, 21:869, 21:872, 21:1219, 21:1222, 42:13
\directlua 04:2, 04:14, 04:29, 04:213, 04:231, 21:1011, 04:256, 04:261, 04:265, 01:12, 01:15, 06:34, 01:20, 01:23, 50:1254, 50:1385, 06:754, 06:839, 01:28, 57:98, 57:100, 57:108, 57:113, 57:115, 02:65, 02:81, 02:105, 02:256, 02:353, 02:363
disable commands:
disable_callback 04:975
\disable_callback *46*
\DisableGenericHook 08:2738, 08:2744, 08:2892, *306*
\DisableHook 08:2843, *195*
\DiscardShipoutBox 53:82, 53:411, 53:444, 53:500, *1063*
\discretionary 38:251, 06:841, 06:856, 06:866
\displayindent 02:403
\displaylines 38:203
displaymath (env.) 38:345
\displaymath 38:347
\displaystyle 30:539, 30:542, 30:546, 30:548, 38:62, 38:210, 38:372, 38:375, 38:428, 38:470, 38:482, 38:510, 38:535, 38:538
\displaywidowpenalty 02:327
\displaywidth 38:210, 38:371, 38:457, 38:513, 02:402
\div 30:390
\DJ 21:529, 21:1138, 57:650, *1213*
\dj 21:539, 21:1139, 57:650, *1213*
\do . . . 01:110, 13:3, 13:7, 13:16, 13:26, 20:66, 20:69, 20:72, 20:135, 20:138, 20:190, 20:193, 20:233, 20:305, 20:367, 20:414, 20:587, 20:604, 20:755, 20:761, 32:90, 37:541, 37:562, 37:722, 37:732, 37:738, 37:744, 40:78, 40:97, 06:69, 41:244, 41:269, 42:104, 42:116, 42:123, 42:203, 42:337, 42:339, 42:362, 42:365, 42:394, 42:396, 42:417, 42:422, 42:477, 42:505, 42:534, 42:730, 42:786, 45:65, 45:134, 47:36, 47:65, 47:82, 50:232, 50:249, 50:543, 50:562, 50:579, 50:597, 50:602, 50:616, 50:621, 50:657, 50:667, 50:1182, 50:1219, 50:1301, 50:1354, 50:1434, 50:1523, 50:1808, 02:13, 02:14, 07:1952, 01:58, 01:59, *446*
\DocInput 26:8, 30:5, 31:5, 56:4
\doclearpage *1239*
\document *442*
document (env.) 37:8
\document 16:143, 20:9, 37:258, 47:64, 47:81, *1240*
\documentclass 04:16, 26:2, 30:2, 31:2, 50:674, 50:681, 50:740, 50:743, 50:975, 50:1070, 50:1202, 56:2, *975*
\DocumentMetadata 17:6, 17:11, 17:15, 17:18, 17:61, *418*
\documentstyle 50:679, 50:1202
\doesglyphexist 33:1218, 33:1244, 33:1262, 33:1263, 33:1264, 33:1265, 33:1266, 33:1267, 33:1268, 33:1271, 33:1272, 33:1273, 33:1274, 33:1277, 33:1278, 33:1279, 33:1280, 33:1283, 33:1284, 33:1287, 33:1288, 33:1291, 33:1292, 33:1293, 33:1294, 33:1295, 33:1296, 33:1297, 33:1298, 33:1301, 33:1302, 33:1305, 33:1306, 33:1307, 33:1308, 33:1309, 33:1310, 33:1311, 33:1312, 33:1313, 33:1314, 33:1315, 33:1316, 33:1317, 33:1318, 33:1319, 33:1320, 33:1321, 33:1322, 33:1323, 33:1324, 33:1325, 33:1326, 33:1327, 33:1328, 33:1329, 33:1330, 33:1331, 33:1332, 33:1333, 33:1334, 33:1335, 33:1336, 33:1337, 33:1338, 33:1339, 33:1340, 33:1341, 33:1342, 33:1343, 33:1344, 33:1345, 33:1346, 33:1347, 33:1348, 33:1349, 33:1350, 33:1351, 33:1352, 33:1353, 33:1354, 33:1355, 33:1356, 33:1357, 33:1358, 33:1359, 33:1360, 33:1361, 33:1362, 33:1363, 33:1364, 33:1365, 33:1366, 33:1367, 33:1368, 33:1369, 33:1370, 33:1373, 33:1377, 33:1378, 33:1379, 33:1380, 33:1381, 33:1382, 33:1383, 33:1384, 33:1385, 33:1386, 33:1387, 33:1388, 33:1389, 33:1390, 33:1391, 33:1392, 33:1393, 33:1394, 33:1395, 33:1396, 33:1397, 33:1398, 33:1399, 33:1400, 33:1401, 33:1402, 33:1403, 33:1404, 33:1405, 33:1406, 33:1407, *729*
\dollar *1214*
\dospecials 01:110, 37:541, 37:562, 37:722, 37:732, 50:1301, 50:1434, 50:1523, 02:13, 07:1953, 01:58, *1231*
\dot 30:532
\doteq 30:476
\dotfill 06:881, 06:902, 02:546
\dots 21:320, 21:322, *1221*
\doublehyphendemerits 02:334

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\doublerulesep 41:311, 41:338, 41:362
\Downarrow 30:593
\downarrow 30:587
\downbracefill 30:545, 30:564
\dump 57:714

E

\E 50:1308, 50:1311, 50:1339, 50:1441, 50:1444, 50:1471, 50:1530, 50:1533, 50:1561
\edef *646*
\EditInstance 11:1156, *349*
\EditTemplateDefaults 11:1156, *349*
\egroup 02:450, *1208*
\eject 02:513
\ell 30:318
\else *1205*
else commands:
\else: 08:2252, 08:2280, 08:2422, 08:2438, 08:2485, 08:2520, 08:2535, 09:347, 16:26, 16:32, 16:64, 52:241, 06:721, 07:1303, 08:767, 08:1041, 08:1198, 08:1260, 08:1401, 08:1408, 08:1416, 08:1732
\em 29:550, 29:577, 32:42, *653*
\emergencystretch 49:88, 49:94, *1203*
\emforce *652*
\emforce 29:546, 29:573, 29:582, *653*
\eminnershape . 29:554, 29:560, 29:577, *652*
\emph 32:42, *1250*
\empty 02:448
\emptyset 30:325
\emreset *653*
\emreset 29:546, 29:549, 29:580, *653*
\emrest 29:549
\ENC-cmd *1213*
\encodingdefault 21:990, 04:260, 04:275, 04:283, 21:1536, 21:1573, 21:1574, 28:388, 29:658, 29:676, 29:692, 29:709, 30:58, *642*
\EncodingSpecific *1202*
\EncodingSpecificAccent *1202*
\EncodingSpecificAccentedLetter .. *1202*
\EncodingSpecificCommand *1202*
\end 14:250, 26:9, 30:6, 31:6, 37:268, 37:296, 37:310, 37:346, 37:362, 37:376, 37:384, 37:520, 37:521, 38:491, 38:500, 39:112, 06:23, 44:15, 44:17, 50:1316, 50:1320, 50:1327, 50:1449, 50:1453, 50:1459, 50:1538, 50:1542, 50:1548, 53:396, 06:719, 06:776, 56:5, 07:1416, 07:1536, 07:1551, 01:53, *221*
\end\verbvisiblespace 37:343
\endarray 41:171
\endcenter 37:445
\endcsname *200*
\enddisplaymath 38:348
\enddocument 37:8, 37:77, 37:78, 37:141, 37:143, 53:352, *1227*
\endenumerate 39:264
\endeqnarray 38:380, 38:427
\endequation 38:351
\endfilecontents 50:1206
\endflushleft 37:495
\endflushright 37:497
\endgraf 16:140, 16:178, 40:138, 40:144, 02:445, *1206*
\endgroup *1194*
\EndIncludeInRelease 08:2136, 08:2143, 08:2167, 08:2183, 08:2190, 08:2195, 08:2217, 02:579, 08:2237, 08:2267, 08:2285, 08:2290, 08:2300, 08:2304, 08:2307, 08:2334, 08:2349, 08:2360, 08:2378, 08:2390, 08:2402, 08:2412, 08:2442, 08:2461, 08:2524, 02:613, 08:2528, 02:621, 08:2610, 08:2613, 08:2727, 08:2733, 08:2740, 08:2745, 08:2752, 08:2756, 08:2763, 08:2767, 08:2790, 08:2795, 02:646, 09:31, 09:45, 02:686, 02:691, 09:228, 09:295, 02:701, 02:706, 09:420, 09:439, 09:491, 09:495, 09:542, 09:569, 09:572, 09:584, 03:71, 14:157, 14:161, 14:204, 14:210, 16:44, 16:75, 17:43, 17:54, 18:21, 18:31, 18:65, 18:78, 18:86, 18:91, 18:104, 18:110, 18:152, 18:166, 18:178, 18:189, 18:206, 18:218, 18:250, 18:266, 18:283, 18:319, 18:353, 18:375, 18:412, 18:445, 18:481, 18:485, 18:494, 18:498, 18:506, 18:510, 18:520, 18:526, 18:540, 18:547, 20:83, 20:140, 20:195, 20:256, 20:277, 20:289, 20:352, 20:356, 20:435, 20:473, 20:496, 20:531, 20:552, 20:571, 20:578, 20:597, 20:614, 20:627, 20:631, 20:649, 20:660, 20:670, 20:709, 20:736, 21:103, 21:123, 21:168, 21:175, 21:330, 21:352, 21:367, 21:375, 04:234, 04:257, 04:280, 04:284, 22:28, 22:33, 22:57, 22:75, 22:86, 22:91, 22:116, 22:121, 22:146, 22:173, 22:182, 22:224, 22:230, 22:250, 22:253, 23:10, 23:14, 23:25, 23:30, 24:52, 24:73, 24:231, 24:248,

24:294, 24:304, 24:314, 24:366, 24:375, 24:455, 24:460, 24:495, 24:500, 24:517, 24:535, 24:574, 24:607, 24:724, 24:736, 25:384, 25:390, 25:403, 25:415, 25:422, 25:505, 25:520, 25:599, 25:611, 25:622, 25:629, 25:636, 25:709, 25:727, 25:734, 25:738, 25:741, 26:157, 26:160, 26:176, 26:549, 26:555, 27:21, 27:143, 28:77, 28:105, 28:179, 28:209, 28:239, 28:271, 28:306, 28:349, 28:410, 28:416, 28:421, 28:504, 28:511, 28:556, 28:563, 28:837, 28:879, 28:892, 28:899, 28:1087, 28:1095, 29:67, 29:100, 29:122, 29:227, 29:248, 29:300, 29:347, 29:376, 29:387, 29:434, 29:468, 29:494, 29:530, 29:539, 29:576, 29:588, 29:595, 29:629, 29:635, 29:670, 29:686, 29:703, 29:718, 30:88, 30:98, 30:116, 30:125, 30:640, 30:647, 32:33, 32:40, 05:10, 05:18, 05:27, 35:28, 35:45, 35:61, 35:75, 35:82, 35:110, 35:114, 35:138, 35:147, 35:158, 35:168, 35:180, 35:191, 35:202, 37:75, 37:139, 37:178, 05:110, 37:186, 37:192, 37:225, 37:238, 37:266, 37:277, 37:308, 37:333, 05:131, 37:341, 37:359, 37:374, 37:382, 37:387, 37:401, 37:410, 37:420, 37:427, 37:437, 37:442, 37:466, 37:482, 37:491, 05:147, 37:510, 37:516, 37:546, 37:567, 37:589, 05:156, 37:599, 37:616, 37:628, 37:632, 37:640, 37:644, 37:665, 37:678, 37:693, 37:705, 37:727, 37:735, 38:86, 38:95, 05:168, 38:117, 38:124, 38:143, 38:148, 38:156, 38:160, 05:176, 38:175, 38:183, 38:232, 38:249, 38:279, 38:287, 38:316, 38:343, 05:191, 38:407, 38:416, 05:197, 38:448, 38:453, 38:473, 38:485, 38:494, 38:503, 05:203, 05:208, 06:12, 39:134, 06:18, 39:139, 39:154, 39:162, 40:13, 40:22, 40:57, 40:65, 40:90, 40:107, 40:121, 40:132, 40:141, 40:146, 40:152, 40:163, 40:170, 40:227, 40:234, 40:260, 40:271, 40:307, 40:315, 40:354, 40:375, 40:399, 40:417, 40:494, 40:510, 40:527, 40:536, 40:541, 40:564, 40:571, 40:605, 40:612, 06:60, 06:64, 41:64, 41:69, 41:156, 41:164, 41:226, 41:231, 42:15, 42:19, 42:38, 42:47, 42:68, 42:76, 42:88, 42:96, 42:111, 42:121, 42:150, 42:155, 42:171, 42:182, 42:249, 42:263, 42:368, 42:425, 42:462, 42:468, 42:499, 42:529, 42:554, 42:570, 42:581, 42:594, 42:602, 42:623, 42:640, 42:650, 42:654, 42:744, 42:799, 42:818, 42:835, 43:33, 43:39, 43:56, 43:63, 44:19, 44:29, 44:167, 44:173, 44:178, 44:192, 44:198, 44:224, 44:246, 45:104, 06:224, 45:172, 06:229, 45:231, 45:246, 45:293, 45:306, 45:351, 45:368, 45:411, 45:417, 45:426, 45:430, 45:439, 45:445, 45:449, 45:481, 45:498, 45:516, 45:558, 45:564, 47:24, 47:32, 47:76, 47:92, 01:290, 06:324, 49:47, 49:68, 49:75, 50:21, 50:26, 06:352, 50:49, 50:61, 50:84, 50:93, 50:99, 50:111, 50:142, 50:149, 50:174, 50:181, 50:199, 50:210, 50:244, 50:261, 50:272, 50:281, 06:380, 50:312, 06:385, 50:342, 50:359, 50:371, 50:392, 50:405, 50:418, 50:428, 50:464, 50:480, 50:489, 50:523, 01:300, 50:533, 06:404, 50:573, 50:589, 50:611, 50:626, 50:640, 50:647, 50:662, 06:418, 50:671, 50:705, 50:712, 50:729, 50:736, 50:789, 50:796, 50:834, 50:861, 50:889, 50:1045, 50:1102, 50:1132, 50:1139, 50:1157, 50:1167, 06:459, 06:469, 50:1342, 50:1473, 50:1563, 06:501, 06:510, 06:537, 06:543, 52:14, 52:23, 52:88, 52:140, 52:179, 52:191, 06:557, 52:200, 06:562, 52:245, 52:256, 52:263, 52:298, 52:321, 52:342, 52:356, 52:361, 52:384, 52:404, 52:429, 52:474, 52:494, 52:501, 52:531, 52:536, 01:25, 53:429, 53:466, 53:483, 53:487, 06:663, 54:53, 54:62, 54:180, 06:678, 54:198, 54:367, 54:372, 54:420, 54:466, 54:654, 54:712, 54:815, 06:729, 54:834, 06:733, 54:897, 54:915, 54:957, 54:978, 06:765, 54:1220, 06:774, 54:1389, 54:1471, 54:1565, 54:1687, 54:1814, 54:1924, 54:1932, 06:824, 54:1966, 54:1995, 06:831, 54:2215, 54:2233, 54:2280, 06:853, 54:2324, 06:864, 06:867,

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

06:895, 57:15, 57:19, 57:29, 57:33, 57:51, 57:69, 57:79, 57:86, 57:94, 06:916, 57:141, 57:146, 57:198, 57:222, 57:292, 57:297, 57:338, 57:344, 57:440, 57:487, 07:237, 07:260, 02:87, 01:34, 02:101, 02:118, 02:123, 02:133, 02:137, 02:147, 02:150, 07:1142, 07:1145, 07:1160, 07:1173, 07:1176, 02:167, 07:1306, 07:1309, 02:181, 07:1338, 07:1341, 07:1350, 07:1358, 07:1361, 02:185, 07:1521, 07:1525, 02:219, 02:224, 02:232, 02:240, 07:1993, 07:2004, 02:288, 02:300, 07:2892, 07:2922, 07:2992, 07:2999, 02:359, 02:367, 08:85, 08:101, 08:129, 08:145, 08:157, 08:169, 08:185, 08:198, 08:218, 08:228, 08:248, 08:263, 08:281, 08:287, 08:307, 08:325, 08:329, 08:578, 08:629, 08:646, 08:650, 08:673, 08:694, 08:728, 08:786, 08:816, 08:848, 08:875, 08:878, 08:898, 08:901, 08:914, 08:932, 08:937, 08:940, 08:949, 08:954, 08:957, 08:994, 02:471, 08:1031, 08:1081, 08:1088, 08:1104, 08:1111, 08:1131, 08:1135, 02:488, 08:1187, 08:1191, 08:1207, 08:1211, 08:1239, 08:1242, 08:1265, 08:1269, 08:1286, 08:1291, 08:1298, 08:1302, 08:1338, 08:1362, 08:1445, 08:1472, 08:1506, 08:1545, 08:1634, 08:1717, 08:1948, 08:2052, 08:2057, 08:2062, *28*

`\EndIncludeRelease` 03:78
`\endinput` *1207*
`\enditemize` 39:275
`\endline` 38:188, 02:445
`\endlinechar` 20:684, 20:718, 01:188, 06:37, 06:39, 06:44, 50:433, 50:434, 50:435, 57:329, 02:380, 01:76, 01:77, 01:78, *1239*
`\endlist` 39:98, 39:264, 39:275
`\endlrbox` 40:194
`\endmath` 38:346
`\endminipage` 40:460
`\EndModuleRelease` 08:2914, 09:608, 10:193, 03:111, 03:112, 03:152, 11:1199, 16:183, 17:63, 33:809, 36:280, 48:493, 55:249, 07:3012
`\endpicture` 42:49
`\endscname` *1193*
`\endsloppypar` 49:92
`\endtabbing` 41:84
`\endtabular` 41:171
`\endtabular*` 41:171
`\endtrivlist` 37:445, 37:495, 37:497, 37:570, 38:515, 39:100, 39:101, 41:85, 43:66
`\endverbatim` 37:569, 37:587
`\enlargethispage` 54:1859, *953*
`\enlargethispage*` 54:1859
`\enskip` 18:549
`\enspace` 18:537, 18:545
`\ensuremath` 22:226, 38:429, 45:409, 45:416, 45:437, 45:444, *1230*
`enumerate` (env.) 39:255
`\enumerate` 39:255
environments:
- `array` 41:168
- `center` 37:444
- `displaymath` 38:345
- `document` 37:8
- `enumerate` 39:255
- `eqnarray` 38:357, 38:516
- `eqnarray*` 38:425
- `equation` 38:349, 38:504
- `filecontents` 50:1206, *975*
- `flushleft` 37:494
- `flushright` 37:496
- `itemize` 39:266
- `list` 39:34
- `lrbox` *818*
- `math` 38:345
- `minipage` *819*
- `picture` 42:21
- `sloppypar` 49:91
- `tabbing` 41:71
- `tabular` 41:174
- `thebibliography` *937*
- `trivlist` 39:89
- `verbatim` 37:569
- `verbatim*` 37:584

`\epsilon` 30:279
`eqnarray` (env.) 38:357, 38:516
`\eqnarray` 38:362, 38:426
`eqnarray*` (env.) 38:425
`\eqno` 38:351, 38:439, *1217*
`equation` (env.) 38:349, 38:504
`\equation` 38:350
`\equiv` 30:455
`\errhelp` 03:31, 14:39, 14:66, 01:201, 56:12, 57:307, 57:703
`\errmessage` 03:32, 10:72, 14:47, 14:72, 04:63, 01:7, 24:546, 24:581, 26:425, 26:525, 27:65, 01:206, 05:79, 05:94, 56:16, 57:63, 57:309, 01:42, 02:164, 02:178, 02:304

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

\ERROR 07:1638, 07:1648, 07:1657, 07:2029, 07:2042, 07:2062, 07:2081, 07:2157, 07:2184, 08:1718, 08:1719, *169*
\errorcontextlines 02:569, 02:588, 02:604, 02:619, 02:633, 02:657, 02:674, 14:212, 02:389, *1201*
\ERRORmissingcells 55:145
\errorstopmode 57:713, 02:554, *1226*
\ERRORusetaggingsocket 55:26, 55:36
\escapchar *455*
\escapechar 20:454, 04:212, 24:392, 24:616, 26:229, 28:58, 28:86, 28:158, 28:189, 28:219, 28:250, 28:372, 06:126, 06:169, 06:173, 06:181, 06:287, 06:288, 06:312, 06:449, 52:280, 52:303, 52:326, 52:347, 06:572, 06:590, 02:377, *102*
\eta 30:281
\etatcatcode 04:1026
\eTeXversion 01:41
\evensidemargin 54:73, 54:617, 54:676
\everycr 38:205, 38:208, 38:371, 38:532, 02:534, *1198*
\everydisplay 24:350, 24:351, 24:360, 24:371
\everyeof 57:329
\everyjob 03:37, 04:216, 04:262, 04:263, 28:392, 53:29, 53:30, 57:112, 57:331, 57:416, 57:673, 57:674, 57:676, *38*
\everymath . 24:349, 24:351, 24:364, 24:373
\everypar *399*
\everypar .. 16:35, 16:41, 16:72, 16:77, 16:120, 16:130, 16:143, 16:181, 20:43, 20:113, 20:171, 24:669, 24:683, 24:730, 37:241, 37:543, 37:565, 39:130, 39:131, 39:133, 39:137, 39:138, 39:204, 39:221, 40:389, 40:410, 41:81, 44:48, 44:96, 44:107, 44:127, 44:136, 45:187, 54:167, 54:194, 54:1155, 54:1321, *416*
\EveryShipout *1066*
\ExecuteOptions 26:57, 26:70, 33:877, 50:650
\exhyphenpenalty 02:322, 02:507
\exists 30:331
\exp 38:31
exp commands:
\exp:w ... 07:2399, 08:757, 08:811, *246*
\exp_after:wN 08:2102, 08:2231, 08:2415, 08:2418, 08:2482, 08:2944, 09:170, 09:245, 09:299, 09:300, 09:353, 09:471, 11:285, 11:740, 11:873, 11:880, 16:41, 16:72, 28:319, 28:322, 49:41, 51:46, 52:73, 52:236, 52:237, 07:322, 07:338, 07:392, 07:398, 07:412, 07:1134, 07:1154, 07:1158, 07:1169, 07:1171, 07:1220, 07:1269, 07:1299, 07:1330, 07:1337, 07:1413, 07:1547, 07:1607, 07:1631, 07:1647, 07:1665, 07:1671, 07:1675, 07:1706, 07:1740, 07:1747, 07:1760, 07:1766, 07:1772, 07:1778, 07:1784, 07:1790, 07:1796, 07:1802, 07:1833, 07:1961, 07:2402, 07:2403, 07:2449, 07:2450, 07:2451, 07:2452, 07:2453, 07:2454, 07:2455, 07:2486, 08:56, 08:61, 08:380, 08:381, 08:426, 08:756, 08:810, 08:1040, 08:1042, 08:1183, 08:1348, 08:1521, 08:1731, 08:1840, 08:1873, 08:1958, *246*
\exp_args:Nc 08:2172, 09:29, 09:42, 09:74, 09:132, 48:22, 07:263, 07:1065, 07:1076, 07:1376, 07:1417, 07:1436, 07:1540, 08:1204, 08:1217
\exp_args:Ncc 07:138, 07:180
\exp_args:NcV 08:569
\exp_args:Ne 08:2320, 08:2321, 09:505, 11:845, 51:110, 51:141, 52:46, 52:97, 07:2475, 08:714, 08:716
\exp_args:Nee 52:129
\exp_args:Nf 09:161, 09:236, 52:32, 52:34, 52:394, 07:417, 07:2161, 07:2188, 07:2249, 07:2477
\exp_args:NNc 09:146, 09:151
\exp_args:NNe 07:1346, 08:1623
\exp_args:Nne 08:1850
\exp_args:NNNo 07:398, 07:1875
\exp_args:Nnnv 08:257
\exp_args:NNo ... 08:2945, 09:169, 09:244, 08:1064, 08:1121, 08:1712
\exp_args:NNV 09:188, 09:263, 08:1596, 08:1683
\exp_args:NNx 07:1435, 08:1502, 08:1540
\exp_args:No 08:2120, 08:2934, 08:2945, 09:169, 09:188, 09:244, 09:263, 36:41, 36:46, 07:90, 07:319, 07:409, 07:428, 07:437, 07:525, 07:909, 07:1305, 07:2034, 07:2045, 07:2086, 07:2167, 07:2194, 07:2417, 07:2443, 08:1064, 08:1121, 08:1182, 08:2061
\exp_args:Nof 07:2037
\exp_args:Noo 07:1278
\exp_args:NV 09:189, 09:264, 51:192, 52:53, 07:256, 07:361, 07:1502
\exp_args:Nv 08:2100, 08:1249, 08:1990, 08:1998
\exp_args:Nx 36:10, 53:29, 07:2416, 07:2419, 07:2442, 08:432, 08:474, 08:545
\exp_args_generate:n 57:593

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\exp_end: 07:2402, 08:757, 08:811
\exp_last_unbraced:Ne 08:2936, 07:2488, 08:1066, 08:1099, 08:1123, 08:1216
\exp_last_unbraced:Nf 09:110, 08:1229, 08:1897
\exp_last_unbraced:NNf 08:1157
\exp_last_unbraced:NNNNo 09:116, 09:474, 08:383, 08:2055
\exp_last_unbraced:NnNo 07:2261
\exp_last_unbraced:NNo 09:364
\exp_not:N . . 09:108, 09:110, 09:111, 09:112, 09:184, 09:199, 09:206, 09:207, 09:214, 09:219, 09:259, 09:274, 09:281, 09:282, 09:286, 09:290, 09:307, 09:352, 09:358, 09:376, 09:385, 09:459, 09:460, 09:480, 09:485, 09:508, 09:535, 09:536, 09:563, 09:564, 11:277, 11:278, 11:279, 11:280, 11:290, 11:296, 11:299, 11:301, 11:302, 11:306, 11:308, 11:311, 11:316, 11:460, 11:461, 11:476, 11:478, 11:494, 11:667, 11:669, 11:772, 11:780, 11:820, 11:835, 11:860, 11:867, 16:39, 16:40, 16:70, 16:71, 16:79, 16:80, 36:72, 48:101, 48:104, 48:106, 48:111, 48:114, 51:192, 51:193, 51:194, 53:31, 53:95, 53:125, 57:600, 07:120, 07:135, 07:136, 07:164, 07:165, 07:197, 07:199, 07:200, 07:206, 07:208, 07:267, 07:279, 07:296, 07:297, 07:298, 07:300, 07:368, 07:918, 07:928, 07:998, 07:1011, 07:1035, 07:1053, 07:1060, 07:1098, 07:1106, 07:1107, 07:1139, 07:1185, 07:1186, 07:1193, 07:1194, 07:1195, 07:1199, 07:1202, 07:1203, 07:1205, 07:1224, 07:1227, 07:1228, 07:1229, 07:1237, 07:1284, 07:1303, 07:1317, 07:1324, 07:1325, 07:1426, 07:1427, 07:1807, 07:1808, 07:1916, 07:1918, 07:1919, 07:1985, 07:1988, 07:2001, 07:2478, 07:2481, 07:2482, 07:2483, 07:2484, 07:2486, 08:391, 08:1151, 08:1155, 08:1174, 08:1177, 08:1178, 08:1180, 08:1224, 08:1486, 08:1488, 08:1624, 08:1626, *315*
\exp_not:n 08:2934, 09:171, 09:184, 09:187, 09:189, 09:224, 09:246, 09:259, 09:262, 09:264, 09:292, 09:356, 09:386, 09:388, 09:389, 09:391, 09:392, 11:282, 11:463, 11:588, 11:668, 11:773, 11:774, 11:781, 11:819, 11:821, 11:822, 11:835, 11:838, 11:839, 11:860, 11:861, 11:867, 11:868, 36:238, 48:236, 48:237, 48:238, 48:263, 48:328, 48:329, 48:330, 48:332, 48:333, 48:334, 48:336, 48:337, 48:338, 48:340, 48:341, 48:342, 48:344, 48:345, 48:346, 48:348, 48:349, 48:350, 51:176, 52:211, 52:225, 53:30, 07:159, 07:163, 07:167, 07:170, 07:174, 07:191, 07:198, 07:210, 07:280, 07:299, 07:368, 07:371, 07:372, 07:625, 07:759, 07:865, 07:876, 07:878, 07:879, 07:947, 07:999, 07:1012, 07:1023, 07:1035, 07:1052, 07:1138, 07:1140, 07:1193, 07:1206, 07:1238, 07:1273, 07:1531, 07:1533, 07:1546, 07:1616, 07:1617, 07:1660, 07:1720, 07:1917, 07:2052, 07:2053, 07:2093, 07:2094, 07:2276, 07:2285, 07:2302, 07:2316, 07:2328, 07:2478, 08:114, 08:241, 08:548, 08:1048, 08:1072, 08:1171, 08:1175, 08:1181, 08:1182, 08:1630, *124*
\exp_stop_f: 07:1458, 07:2039, 08:1250, 08:1398, 08:1406
expandable commands:
\expandable_grab_⟨*type*⟩:w *147*
\expandafter *1221*
expandafter commands:
\expandafter: 32:88, 41:269
\ExpandArgs *78*
\ExpandArgs 05:177, 05:193, 05:196, 06:534, 06:535, 57:597
\expanded 50:902, 57:111, *236*
\ExplLoaderFileDate 05:201, 05:207
\ExplSyntaxOff 08:2951, 09:381, 09:609, 10:194, 11:1200, 16:184, 17:42, 17:64, 20:529, 20:749, 20:799, 28:304, 28:347, 36:281, 05:146, 05:166, 05:189, 48:494, 49:45, 50:515, 51:257, 52:12, 52:86, 52:138, 52:243, 52:254, 52:402, 52:427, 52:472, 52:529, 52:584, 53:427, 53:474, 53:514, 55:62, 55:239, 57:570, 57:648, 07:3013, *981*
\ExplSyntaxOn 09:3, 09:381, 10:2, 11:6, 16:3, 17:2, 17:22, 20:518, 20:743, 20:796, 28:277, 28:310, 36:2, 05:135, 05:161, 05:180, 48:3, 49:22, 50:513, 51:3, 52:7, 52:29, 52:92, 52:207, 52:251, 52:390, 52:410, 52:435, 52:508,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

52:577, 53:5, 53:472, 53:509, 55:2, 55:99, 07:8, 57:560, 57:571, 08:3, *304*
\extracolsep 41:167
\extrafloats 02:152, 02:189, 02:274

F

\fam 04:22, 04:26, 04:38, 24:15, 02:98, 02:375
\family *1194*
\familydefault 21:1574, 28:389, 29:444, 29:659, 29:677, 29:693, 29:710, 30:127, *642*
\fbox *818*
\fbox 40:212, 40:225, 40:232, *1211*
\fboxrule 40:210, 40:257, 40:269, 40:274, 40:280, 40:282, 40:289, 40:290, 57:151
\fboxsep 40:210, 40:216, 40:249, 40:256, 40:268, 40:275, 40:285, 40:287, 57:150
\fi *1241*
fi commands:
\fi: 08:2103, 08:2207, 08:2215, 08:2232, 08:2254, 08:2282, 08:2419, 08:2424, 08:2440, 08:2487, 08:2522, 08:2537, 09:342, 09:344, 09:345, 09:349, 09:354, 09:361, 16:26, 16:33, 16:65, 16:103, 52:241, 07:876, 07:879, 07:946, 07:1303, 07:1460, 07:2399, 07:2403, 08:769, 08:1043, 08:1200, 08:1223, 08:1263, 08:1402, 08:1410, 08:1416, 08:1734, *319*
\filbreak 02:511
file commands:
\g_file_curr_name_str 08:2781
\file_full_name:n . 20:525, 52:35, *1045*
\file_mdfive_hash:n 20:797
\file_parse_full_name_apply:nN 52:32, 52:47, 52:90, 52:93, 52:95
\l_file_search_path_seq *1037*
\file_size:n 20:798
file internal commands:
__file_parse_full_name_area:nw 52:102, 52:105, 52:109
__file_parse_full_name_auxi:nN 52:97, 52:100
__file_parse_full_name_base:nw 52:108, 52:112, 52:124
__file_parse_full_name_tidy:nnnN 52:119, 52:120, 52:122, 52:127
file/.../after *1036*
file/.../before *1036*
file/after *1036*
file/before *1036*
filecontents (env.) 50:1206, *975*
\filecontents 50:1206
filehook internal commands:
__filehook_clear_replacement_flag: 52:417, 52:420, 52:520
__filehook_drop_extension:N 52:44, 52:49, 52:522
__filehook_drop_extension_aux:nnn 52:54, 52:57
__filehook_file_name_compose:nnn 52:218, 52:406, 52:414, 52:415, 52:425
__filehook_file_parse_full_name:nN 52:28, 52:30, 52:30, 52:53, 52:238, 52:394, 52:412, 52:414, *1042*
__filehook_file_pop: 52:59, 52:70, 52:526
__filehook_file_pop_assign:nnnn 52:59, 52:73, 52:79, 52:528
__filehook_file_push: 52:59, 52:62, 52:524
__filehook_file_replaced 52:417
__filehook_file_subst_begin:nnn 52:412, 52:422, 52:422, *1052*
__filehook_file_subst_cycle_error:NN 52:457, 52:462, 52:462, 52:467, 52:469, *1054*
__filehook_file_subst_loop:NN . . . 52:431, 52:444, 52:452, 52:461, *1053*
__filehook_file_subst_tortoise_hare:nn 52:424, 52:431, 52:434, 52:436, 52:459, *1054*
__filehook_full_name:nn 52:30, 52:34, 52:38
__filehook_if_file_replaced:TF 52:417, 52:418, 52:518, *1053*
__filehook_if_no_extension:nTF 52:44, 52:44, 52:510
\g__filehook_input_file_seq 52:59, *1043*
\l__filehook_internal_tl 52:59
__filehook_log_file_record:n 52:547, 52:547, 52:561, 52:563
\g__filehook_nesting_level_int . . 52:544, 52:549, 52:552, 52:553, 52:559
__filehook_normalize_file_name:w 52:406, 52:413, 52:516
__filehook_resolve_file_subst:w 52:406, 52:409, 52:411, 52:514
__filehook_set_curr_file:nNN 52:392, 52:512
__filehook_set_curr_file_assign:nnnNN 52:395, 52:397
__filehook_subst_add:nn 52:206, 52:208, 52:208, 52:252, *1047*
__filehook_subst_empty_name_chk:NN 52:208, 52:236, 52:240

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

__filehook_subst_file_normalize:Nn 52:208, 52:216, 52:218, 52:230, 52:234
__filehook_subst_remove:n 52:208, 52:222, 52:253
\filename@parse *6*, *1*
\filesize 05:72, 05:118
\fill 18:530
\finalhyphendemerits 37:453, 37:457, 37:463, 02:335, *1253*
\finalstrut *1220*
\FirstMark 48:366, 48:484, *944*
\firstmark 49:80, 54:651, 54:710, 54:2255, *942*
flag internal commands:
\flag␣__filehook_file_replaced *1053*
flag commands:
\flag_clear:n 52:421
\flag_if_raised:nTF 36:109, 36:201, 52:419, 52:440
\flag_new:n 36:88, 52:417
\flag_raise:n 36:110, 52:441
\flat 30:335
\floatingpenalty 45:469, 45:488, 45:506, 02:332
\floatpagefraction 45:278, 54:2336
\floatsep 54:730, 54:748, 54:755, 54:2142, 54:2192, 54:2341
\flushbottom 49:84
flushleft (env.) 37:494
\flushleft 37:494
flushright (env.) 37:496
\flushright 37:496
\fmtname 03:1, 03:41, 03:43, 03:46, 03:48, 03:51, 03:60, 50:749, 50:753, *1199*
\fmtversion 03:1, 03:19, 03:41, 03:43, 03:46, 03:48, 03:51, 03:60, 03:105, 03:118, 03:166, 14:2, 04:276, 21:1538, 29:225, 41:1, 42:1, 50:169, 50:177, 50:766, 50:769, 54:4, 57:657, 57:683, *1199*
\fnsymbol *512*
\fnsymbol 22:189
\font 21:297, 21:298, 21:299, 21:439, 21:446, 21:799, 21:806, 21:866, 21:1003, 21:1004, 21:1060, 21:1165, 21:1167, 21:1169, 21:1216, 24:81, 24:87, 24:89, 26:84, 29:553, 29:586, 29:592, 29:618, 31:8, 31:9, 31:10, 32:85, 33:6, 33:612, 33:626, 33:633, 37:564, 06:842, 06:845, 06:857, 06:860, 02:539, 02:544
\fontdimen 21:297, 21:298, 21:299, 21:439, 21:446, 21:799, 21:806, 29:553, 29:586, 29:592, 32:85, 33:6, 33:612, 33:626, 33:633, 42:126, 42:129, 42:678, 02:539, 02:544
\fontencoding . . 21:867, 04:259, 04:260, 04:282, 04:283, 21:1573, 24:251, 24:286, 24:298, 24:308, 28:388, 29:658, 29:676, 29:692, 30:16, 30:24, 30:75, 30:82, 30:91, 30:93, 33:36, *1212*
\fontfamily . . 24:279, 29:6, 29:9, 29:12, 29:145, 29:156, 29:482, 29:485, 29:488, 29:738, 30:66, 30:77, 30:84, 30:95, 33:28, 33:30, 33:32, 33:34, 33:85, 33:87, 33:89, 33:91, 33:913, *1196*
\FontFamilyToCheck 33:1620, 33:1621, 33:1634
\fontname 21:1004, 24:89
\fontseries 24:279, 25:393, 25:394, 25:405, 25:407, 25:417, 25:419, 29:15, 29:18, 29:258, 29:259, 29:260, 29:261, 29:281, 29:282, 29:283, 29:284, 29:319, 29:320, 29:321, 29:322, 29:339, 29:340, 29:341, 29:342, 29:355, 29:356, 29:357, 29:358, 29:366, 29:367, 29:368, 29:369, 29:382, 29:385, 29:476, 29:479, 29:739, *1251*
\fontseriesforce 25:398, 25:409, 25:420, *560*
\fontshape 21:449, 21:809, 24:279, 25:528, 25:533, 25:538, 25:614, 25:615, 25:624, 25:626, 25:631, 25:633, 25:684, 25:688, 25:691, 25:694, 25:697, 25:700, 25:703, 25:706, 25:713, 29:21, 29:24, 29:27, 29:30, 29:740, 33:636, *1250*
\fontshapeforce 25:618, 25:627, 25:634, 25:714
\fontsize 19:6, 21:302, 21:328, 21:360, 21:868, 21:1218, 21:1254, 24:79, 24:318, 29:641, 29:741, 33:104, 33:672, 33:930, 45:409, 45:416, 45:437, 45:444, *1212*
\fontsubfuzz 26:441, 26:475, 37:53, 37:118, 37:157, *1208*
\FOO *81*
\foo *313*
\footins 45:390, 45:465, 45:484, 45:502, 54:316, 54:317, 54:318, 54:319, 54:377, 54:424, 54:486, 54:494, 54:498, 54:521, *1225*
\footnote 45:452, *1195*
\footnotemark 44:10, 45:518, *1195*
\footnoterule 40:466, 45:394, 54:497, *1225*
\footnotesep 40:490, 40:507, 40:524, 45:451, 45:468, 45:477, 45:487, 45:495, 45:505, 45:513

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\footnotesize 40:482, 40:500, 40:517, 45:466, 45:485, 45:503
\footnotetext 44:12, 45:535, *1195*
\footref 45:547, *1256*
\footskip 54:77, 54:641, 54:700
\forall 30:330
fp commands:
  \fp_eval:n 11:368, 05:162
  \l_tmpa_fp *346*
\fpeval *77*
\fpeval 05:158, 05:170, 05:172, *77*
\frac 38:354, *1201*
\frame 40:196, 40:297
\framebox *818*
\framebox 40:219, *1216*
\frenchspacing 20:46, 20:116, 20:174, 37:569, 37:586, 37:737, 06:882, 06:903, 02:431
\frown 30:458
\fussy 49:93
\futurelet 18:457, 18:465, 32:83, 05:49, 38:256, 41:359, 06:782, 06:796

G

\g_@@_\meta␣{hook}_code_prop *225*
\g_@@_\meta␣{hook}_declared_tl . . . *225*
\g_@@_\meta␣{hook}_parameter_tl . . . *225*
\g_@@_\meta␣{hook}_reversed_tl . . . *225*
\Gamma 30:304
\gamma 30:277
\gcd 38:33
\gdef *982*
gdef commands:
  \gdef_ 38:269
\ge 30:424, 30:426
\Generic* *1219*
\GenericError 14:18, 14:85, 14:111, 14:164, 26:62, *1215*
\GenericInfo . . . 03:106, 03:108, 03:119, 03:122, 03:127, 03:162, 03:163, 03:168, 03:172, 03:176, 14:4, 03:197, 14:104, 14:130, 14:182, 26:31, 26:34, 26:39, 26:75, 05:63, 50:66, 50:74, 50:104, 50:107, 50:1734, *1215*
\GenericWarning 14:11, 14:94, 14:120, 14:141, 14:149, 14:173, 14:195, 26:42, 26:47, 26:50, 26:78, *1215*
\geq 30:422, 30:424
\getanddefinefonts *1193*
\GetFileInfo 30:3
\gets 30:446, 30:448
\gg 30:440
\gloassary *1249*
\global *1233*
\globaldefs 24:618, 26:231, 28:60, 28:89, 28:160, 28:191, 28:221, 28:253, 02:372
\glossary *934*
\glossary 44:182, 46:23, 46:35, 48:253, 49:25, 49:36, 49:53, 49:61, 54:625, 54:684, *943*
\glossaryentry 46:32
\glyphmissingdetails 33:1219, 33:1244, *728*
\goodbreak 06:883, 06:904, 02:511
\grave 30:524
\group *1193*
group commands:
  \group_align_safe_begin/end: . . *164*
  \group_align_safe_begin: 07:286, 07:298, 07:403, *122*
  \group_align_safe_end: 07:320, 07:426, 07:1863, *123*
  \group_begin: 09:201, 09:276, 16:17, 16:23, 16:50, 16:56, 28:293, 28:332, 48:66, 48:220, 51:4, 52:210, 52:224, 53:92, 07:1855, 07:2231, 07:2236, 07:2408, 07:2412, 07:2440, 08:387, 08:1150, 08:1163
  \c_group_begin_token 07:682, 07:693, 07:1861, 07:1942, 07:2520
  \group_end: 09:205, 09:280, 16:19, 16:28, 16:52, 16:60, 28:296, 28:335, 48:71, 48:240, 51:21, 52:220, 52:232, 53:94, 07:1876, 07:2240, 07:2245, 07:2422, 07:2423, 07:2425, 07:2427, 07:2437, 07:2444, 07:2445, 08:390, 08:1154, 08:1173
  \c_group_end_token . . . 07:685, 07:1931
  \group_insert_after:N 28:318, 28:319, 28:322, 28:344, 28:345
\guillemetleft 21:540, 21:773, 21:1104, *1247*
\guillemetright 21:541, 21:774, 21:1120, *1247*
\guillemotleft . . . 21:543, 21:776, 21:1106
\guillemotright . . 21:544, 21:777, 21:1122
\guilsinglleft 21:545, 21:1183
\guilsinglright 21:546, 21:1184

H

\H 14:24, 21:230, 21:385, 21:469, 21:601, 21:609, 21:628, 21:636, 21:753, 21:1246, 21:1385, 21:1386, 21:1413, 21:1414, 33:177, 33:201
\halign 38:127, 38:210, 38:371, 38:532, 02:534, *1193*
\hang *1224*
\hangafter 02:374
\hangindent 44:139, 02:405
\hat 30:530

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\hbadness 24:672, 24:679, 24:715, 24:734, 53:238, 53:240, 53:290, 53:292, 02:318
\hbar 30:342
\hbox *965*
hbox commands:
  \hbox:n 53:382
  \hbox_set:Nn 53:170, 53:217, 53:243, 53:268, 53:297
  \hbox_set_to_wd:Nnn 53:241, 53:293, 53:329
  \hbox_unpack:N 48:389, 48:420, 53:251, 53:295
  \hbox_unpack_drop:N 53:180
\hbox␣to␣ *1209*
\headheight 54:75, 54:630, 54:689
\headsep 54:76, 54:639, 54:698
\heartsuit 30:340
\height . 21:872, 21:1222, 40:31, 40:34, *1200*
\hfil *803*
\hfill *1196*
\hfuzz 24:680, 49:89, 49:90, 49:96, 49:97, 53:237, 53:239, 53:289, 53:291, 02:390
\hglue 02:501
\hideoutput 02:692
\hideskip 02:309, 02:525
\hidewidth 21:327, 21:329, 21:358, 21:362, 21:390, 21:391, 21:394, 21:397, 21:476, 21:477, 21:481, 21:484, 21:486, 21:489, 21:501, 21:506, 21:522, 21:760, 21:761, 21:764, 21:767, 21:834, 21:837, 21:1253, 21:1255, 02:525
\hline 41:358, 41:361, *1197*
\hoffset 02:406
\holdinginserts 02:339
\hom 38:29
hook commands:
  \hook_activate_generic:n 08:2737, 08:2823, 08:2842, 08:2855, 08:288, 08:288, 08:290, 08:308, 08:326, 08:328, *234*
  \hook_debug_off: 08:2799, 08:7, 08:13, *209*
  \hook_debug_on: 08:2798, 08:7, 08:8, *209*
  \hook_disable:n 08:2811, 08:2811
  \hook_disable_generic:n 08:2739, 08:2816, 08:2848, 08:264, 08:264, 08:266, 08:282, 08:285, *207*
  \hook_gclear_next_code:n 08:2184, 08:2184, 08:2769, *208*
  \hook_gput_code:nnn 08:2749, 08:486, 08:486, 08:488, 08:579, 08:581, 08:681, 08:702, *243*
  \hook_gput_code_with_args:nnn 08:2751, 08:486, 08:494, 08:628, *208*
  \hook_gput_next_code:nn 08:2122, 08:2122, 08:2124, 08:2137, 08:2139, 08:2760, 08:690, 08:708, *208*
  \hook_gput_next_code_with_-args:nn 08:2130, 08:2142, 08:2762, *208*
  \hook_gremove_code:nn . . 08:2771, 08:958, 08:958, 08:960, 08:995, *208*
  \hook_gset_rule:nnnn . . . 08:2801, 08:2803, 08:2806, 08:1303, 08:1303, *209*
  \hook_if_empty:n 08:2426, 08:2428, 08:2443, 08:2445
  \hook_if_empty:nTF 08:2426, 08:2807, 08:2808, 08:2809, 53:63, 53:164, 08:1862, 08:1968, *200*
  \hook_if_empty_p:n 08:2426, 53:67, 53:113, 53:377, 08:1917, 08:2017, *209*
  \hook_log:n 08:2797, 08:1818, 08:1818, *209*
  \hook_new:n 08:2714, 48:259, 53:153, 53:154, 53:155, 53:156, 53:157, 53:158, 53:159, 08:67, 08:69, 08:88, 08:223, 08:861, *206*
  \hook_new_pair:nn 08:2718, 16:6, 16:7, 08:199, 08:201, 08:221, *206*
  \hook_new_pair_with_args:nn . . . *207*
  \hook_new_pair_with_args:nnn 08:2726, 08:199, 08:199, 08:203, 08:219, 08:226, *1265*
  \hook_new_reversed:n . . . 08:2716, 08:170, 08:172, 08:188, 08:224, *206*
  \hook_new_reversed_with_args:nn 08:2724, 08:170, 08:170, 08:174, 08:186, 08:197, *207*
  \hook_new_with_args:nn . . 08:2722, 08:67, 08:67, 08:71, 08:86, 08:100, *207*
  \g_hook_patch_action_list_tl 09:6, 09:91, 09:124
  \hook_provide:n 08:2811, 08:2818
  \hook_provide_pair:nn 08:2811, 08:2832
  \hook_provide_reversed:n 08:2811, 08:2825
  \hook_show:n 08:2796, 08:1818, 08:1823, *209*
  \hook_use:n 08:2196, 08:2196, 08:2198, 08:2218, 08:2220, 08:2238, 08:2240, 08:2295, 08:2786, 16:22, 16:31, 16:55, 16:63, 16:98, 16:105, 48:222, 53:62, 53:65, 53:71, 53:75, 53:123, 08:1465, *236*
  \hook_use:nnw 08:2236, 08:2266, 08:2268, 08:2268, 08:2270, 08:2286, 08:2288, 08:2296, 08:2788, *208*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\hook_use_once:n 08:2335, 08:2337, 08:2352, 08:2787, *207*
\hook_use_once:nnw 08:2335, 08:2335, 08:2343, 08:2350, 08:2358, 08:2789, *208*
hook internal commands:
\c__hook_ 08:1295, 08:1299
\g__hook_??_code_prop 08:1292
\c__hook_??_parameter_tl . . . 08:1292
\g__hook_??_reversed_tl 08:1292
\g__hook_⟨*hook*⟩_code_prop *289*
\g__hook_⟨*hook*⟩_labels_clist . . . *229*
\g__hook_⟨*hook*⟩_reversed_tl . . . *225*
__hook_activate_generic:n . . 08:288
__hook_activate_generic:nn 08:2840, 08:291, 08:292, 08:310
__hook_activate_generic_pair:nn . . 08:2811, 08:2837, 08:2841, 08:2869
__hook_activate_generic_-reversed:n 08:2811, 08:2830, 08:2839, 08:2842, 08:2862
\g__hook_all_seq 08:2942, 08:28, 08:108, 08:137, 08:1426, 08:1453
__hook_apply_-rule_->:nnn . 08:1793
__hook_apply_-rule_<-:nnn . 08:1793
__hook_apply_-rule_<:nnn . . 08:1793
__hook_apply_-rule_>:nnn . . 08:1793
__hook_apply_-rule_xE:nnn . 08:1793
__hook_apply_-rule_xW:nnn . 08:1793
__hook_apply_label_pair:nnn 08:1573, 08:1574, 08:1655, 08:1656, 08:1720, 08:1720, *280*
__hook_apply_rule:nnn 08:1730, 08:1736, 08:1736, *272*
__hook_apply_rule:nnnN *272*
__hook_apply_rule_->:nnn . . 08:1771
__hook_apply_rule_<-:nnn . . 08:1771
__hook_apply_rule_<:nnn . . . 08:1743
__hook_apply_rule_>:nnn . . . 08:1743
__hook_apply_rule_xE:nnn . . 08:1757
__hook_apply_rule_xW:nnn . . 08:1757
__hook_braced_cs_parameter:n 08:2937, 08:1067, 08:1124, 08:1149, 08:1160, 08:1212, 08:1212, 08:1214, 08:1240
__hook_braced_hidden_loop:w 08:1212, 08:1216, 08:1219, 08:1225
__hook_braced_parameter:n 09:221, 09:505, 08:1243, 08:1243, 08:1245, 08:1266, 08:1268, 08:1487, 08:1489, 08:1624, 08:1626
__hook_braced_real_loop:w . 08:1243
__hook_chk_args_allowed:nn 08:2160, 08:502, 08:630, 08:630, 08:632, 08:647, 08:649, *282*
__hook_clear_next:n 08:2164, 08:2180, 08:2185, 08:2186, 08:2186, 08:2188, 08:2191, 08:2193, *279*
__hook_clist_gput:Nn 08:1495, 08:1497, 08:1529, 08:1533, 08:1597, 08:1684, 08:1718, 08:1719
__hook_cmd_begindocument_code: . 09:46, 09:54, 09:59, 09:62, 09:606, *327*
__hook_cmd_if_scanable:Nn . . 09:450
__hook_cmd_if_scanable:NnTF 09:408, 09:427, 09:450, *324*
__hook_cmd_patch_xparse:Nnn 09:128, 09:149, 09:149
__hook_cmd_try_patch:nn 09:52, 09:63, 09:63
__hook_code_gset:nn . . . 08:2397, 08:118, 08:1089, 08:1089, 08:1091, 08:1103, 08:1105, 08:1107, 08:1484
__hook_code_gset_aux:nnn 08:1052, 08:1089, 08:1092, 08:1094, 08:1096, 08:1097, 08:1110
__hook_code_gset_auxi:nnnn 08:1032, 08:1053, 08:1078, 08:1080, 08:1087, 08:1118, *253*
__hook_cs_end:w 08:1212, 08:1231, 08:1232, 08:1233, 08:1238
__hook_cs_gput_right:nnn 08:2165, 08:2368, 08:540, 08:1032, 08:1032, 08:1034, 08:1082, 08:1084
__hook_cs_gput_right_fast:nnn 08:1032, 08:1040, 08:1046, 08:1085
__hook_cs_gput_right_slow:nnn 08:1032, 08:1042, 08:1049, 08:1086
__hook_cs_if_empty:N 08:1192, 08:1194, 08:1208, 08:1210
__hook_cs_if_empty:NTF 08:2161, 08:2163, 08:2434, 08:2435, 08:1192, 08:1887, 08:1894
__hook_cs_if_empty_p:N 08:1192
__hook_cs_parameter_count:N 08:1212, 08:1217, 08:1227
__hook_cs_parameter_count:w 08:1212, 08:1229, 08:1236, 08:1237
\l__hook_cur_hook_tl 08:29, 08:1559, 08:1641, 08:1777, 08:1788, *273*
__hook_curr_name_pop: 08:2777, 08:2883, 08:417, 08:449, *238*
__hook_curr_name_push:n 08:2775, 08:2780, 08:417, 08:431, *298*
__hook_curr_name_push_aux:n 08:417, 08:432, 08:433
__hook_currname_or_default: 08:2625, 08:2673,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

08:333, 08:343, 08:347, 08:363, 08:364, 08:364, 08:537, 08:611, *235*
__hook_debug:n 09:19, 09:28, 09:39, 09:48, 09:49, 09:65, 09:69, 08:7, 08:7, 08:20, 08:531, 08:604, 08:1425, 08:1434, 08:1452, 08:1455, 08:1477, 08:1501, 08:1511, 08:1539, 08:1578, 08:1598, 08:1660, 08:1685, 08:1745, 08:1752, 08:1759, 08:1767, 08:1773, 08:1784, *222*
\g__hook_debug_bool 08:6, 08:10, 08:15, 08:21
__hook_debug_gset: 08:7, 08:11, 08:16, 08:18
__hook_debug_label_data:N 08:1578, 08:1620, 08:1661, 08:1709, 08:1806, 08:1806
__hook_debug_print_rules:n 08:2106, 08:2106
__hook_declare_deprecated_-generic:NNn . 08:756, 08:777, 08:810
__hook_declare_deprecated_-generic:NNw . 08:772, 08:778, 08:779
__hook_def_cmd:w 09:13, 09:14, 09:192, 09:198, 09:267, 09:273, 09:299, *317*
\g__hook_delayed_patches_prop 09:17, 09:51, 09:57, 09:58
__hook_deprecated_generic_-warn:n 08:755, 08:762, 08:809, 08:987, 08:1023, 08:1314, 08:1345, 08:1838, 08:1956, *246*
__hook_deprecated_generic_-warn:Nn 08:762
__hook_deprecated_generic_-warn:Nw 08:762
__hook_deprecated_generic_-warn:w 08:763, 08:764
__hook_deprecated_warn:nn 08:2813, 08:2820, 08:2827, 08:2834, 08:2845, 08:2852, 08:2859, 08:2866, 08:2871, 08:2871
__hook_disable:n 09:209, 09:501, 08:264, 08:267, 08:268
__hook_do_deprecated_generic:Nn . . 08:772, 08:772, 08:988, 08:1024, 08:1315, 08:1346, 08:1839, 08:1957
__hook_do_deprecated_generic:Nw 08:772, 08:773, 08:774
__hook_double_hashes:n 09:188, 09:263, 09:325, 09:325, 09:363, 08:549, 08:1064, 08:1073, 08:1121, 08:1178, *257*
__hook_double_hashes:w 09:325, 09:326, 09:327, 09:359, 09:363, 09:366, *319*
__hook_double_hashes_group:n 09:325, 09:333, 09:362
__hook_double_hashes_output:N 09:325, 09:330, 09:338, *319*
__hook_double_hashes_space:w 09:325, 09:334, 09:365
__hook_double_hashes_stop:w 09:325, 09:341, 09:361
\c__hook_empty_tl 08:35, 08:1284
__hook_end_document_label_-check: 08:417, 08:456, 08:457, 08:464
__hook_exp_not:n 09:171, 09:172, 09:176, 09:187, 09:246, 09:247, 09:251, 09:262, *314*
__hook_exp_not:NN 09:13, 09:13, 09:193, 09:199, 09:219, 09:224, 09:268, 09:274, 09:290, 09:292
__hook_file_hook_normalize:n 08:2321, 08:716, 08:899, 08:899, 08:902, 08:904, *249*
\l__hook_front_tl 08:1548, 08:1589, 08:1592, 08:1595, 08:1597, 08:1598, 08:1599, 08:1613, 08:1614, 08:1674, 08:1678, 08:1682, 08:1684, 08:1686, 08:1688, 08:1702, 08:1703
\c__hook_generic_⟨*type*⟩/./⟨*place*⟩_tl *245*
\c__hook_generic_class/./after_-tl 08:915
\c__hook_generic_class/./before_-tl 08:915
\c__hook_generic_cmd/./after_tl 08:915
\c__hook_generic_cmd/./before_tl 08:915
\c__hook_generic_env/./after_tl 08:915
\c__hook_generic_env/./before_tl 08:915
\c__hook_generic_env/./begin_tl 08:915
\c__hook_generic_env/./end_tl 08:915
\c__hook_generic_file/./after_tl 08:915
\c__hook_generic_file/./before_-tl 08:915
\c__hook_generic_include/./after_-tl 08:915
\c__hook_generic_include/./before_-tl 08:915

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\c__hook_generic_include/./end_-tl 08:915
\c__hook_generic_package/./after_-tl 08:915
\c__hook_generic_package/./before_-tl 08:915
__hook_generic_parameter:n 08:1059, 08:1279, 08:1290
__hook_generic_parameter:w 08:1280, 08:1281
\c__hook_generics_file_prop 08:891, 08:938
\c__hook_generics_prop 08:827, 08:859, 08:915, 08:933, 08:935
\c__hook_generics_reversed_ii_-prop 08:836, 08:863, 08:938
\c__hook_generics_reversed_iii_-prop 08:840, 08:867, 08:938
__hook_gput_code:nnn 08:486, 08:491, 08:497, 08:500, 08:583, 08:584, 08:662
__hook_gput_code_store:nnn 08:486, 08:513, 08:516
__hook_gput_next_code:nn 08:2127, 08:2133, 08:2141, 08:2144, 08:2144, 08:669
__hook_gput_next_do:nn . 08:2150, 08:2155, 08:2155, 08:2157, 08:2168, 08:2170, 08:670, 08:691, 08:708, *243*
__hook_gput_next_do:Nnn 08:2172, 08:2175
__hook_gput_undeclared_hook:nnn 08:651, 08:651, 08:663, 08:682, 08:702, *243*
__hook_gremove_code:nn . . 08:958, 08:961, 08:962, 08:989, 08:997, 08:1025
__hook_gset_rule:nnnn 08:1303, 08:1305, 08:1308, 08:1310, 08:1315, 08:1339, 08:1341, 08:1346
__hook_guess_arg_count:NN 09:467, 09:467, 09:469, 09:492, 09:494, 09:500
__hook_guess_arg_count:nw 09:467, 09:480, 09:485, 09:488
__hook_guess_arg_count:wN 09:467, 09:471, 09:475
\c__hook_hash_tl . . . 09:11, 09:254, 09:255, 09:257, 09:344, 08:1224, *257*
\c__hook_hashes_tl . 09:11, 09:179, 09:180, 09:182, 09:345, 08:1151, *314*
\g__hook_hook_curr_name_tl 08:32, 08:366, 08:376, 08:417, 08:429, 08:444, 08:445, 08:452, 08:462, 08:463, 08:484, *237*
__hook_hook_gput_code_do:nnn 08:240, 08:257, 08:486, 08:520, 08:529, 08:591, 08:602, 08:654
__hook_if_cmd_hook:n 08:2509, 08:2511, 08:2525
__hook_if_cmd_hook:nTF 08:2509, 08:2527, 08:2068
__hook_if_cmd_hook:w 08:2512, 08:2513
__hook_if_cmd_hook:wTF 08:2509
__hook_if_cmd_hook_p:n 08:2509
__hook_if_cmd_hook_p:w 08:2509
__hook_if_declared:n 08:2474
__hook_if_declared:nTF . 08:2474, 09:67, 08:75, 08:93, 08:178, 08:207, 08:210, 08:297, 08:315, 08:636, 08:861, 08:1037, 08:1055, 08:2067, *227*
__hook_if_declared_p:n 08:2474
__hook_if_deprecated_generic:n 08:2497
__hook_if_deprecated_generic:nTF 08:2489, 08:753, 08:807, 08:985, 08:1021, 08:1312, 08:1343, 08:1836, 08:1954
__hook_if_deprecated_generic:w 08:2498, 08:2499
__hook_if_deprecated_generic_-p:n 08:2489
__hook_if_disabled:n 08:273
__hook_if_disabled:nTF 08:2146, 08:264, 08:294, 08:312, 08:524, 08:595, 08:1848, 08:1860, 08:1943, 08:1966, 08:2046, *233*
__hook_if_disabled_p:n 08:264
__hook_if_execute_immediately:n 08:2413
__hook_if_execute_immediately:nTF 08:2339, 08:2345, 08:2354, 08:2413, 08:503, 08:586, 08:1319
__hook_if_execute_immediately_-p:n 08:2413
__hook_if_file_hook:w 08:876, 08:879, 08:881
__hook_if_file_hook:wTF 08:2318, 08:712, 08:876, *286*
__hook_if_file_hook_p:w 08:876
__hook_if_generic:n 08:2489
__hook_if_generic:nTF 08:2489, 08:736, 08:794, 08:1058, 08:1846, 08:1962
__hook_if_generic:w 08:2490, 08:2491
__hook_if_generic_p:n 08:2489
__hook_if_generic_reversed:n 08:2529
__hook_if_generic_reversed:nTF . 08:2529, 08:302, 08:320, 08:748, 08:802

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

__hook_if_generic_reversed:w 08:2530, 08:2531
__hook_if_generic_reversed_p:n 08:2529
__hook_if_has_hash:n 09:311
__hook_if_has_hash:nTF 09:169, 09:244, 09:311, *318*
__hook_if_has_hash:w 09:311, 09:312, 09:313, 09:319, 09:321
__hook_if_has_hash_check:w 09:311, 09:318, 09:323
__hook_if_has_hash_p:n 09:311
__hook_if_label_case:nnnnn 08:2083, 08:1412, 08:1412, 08:1571, 08:1653
__hook_if_public_command:N . 09:108
__hook_if_public_command:NTF 09:77, 09:87, *311*
__hook_if_public_command:w 09:77, 09:111, 09:117
__hook_if_replacing_args: . 08:2543
__hook_if_replacing_args:TF 08:2539, 08:2545, 08:2550, 08:2551, 08:2556, 08:2557, 08:2562, 08:505, 08:547, 08:558, 08:634, 08:1071
__hook_if_reversed:n 08:2480
__hook_if_reversed:nTF . 08:2480, 08:1493, 08:1526, 08:1883, 08:1923, 08:1925, 08:1984, 08:2023, 08:2026
__hook_if_reversed_p:n 08:2480
__hook_if_structure_exist:n 08:2468
__hook_if_structure_exist:nTF . . 08:2149, 08:2447, 08:2468, 08:2704, 08:150, 08:162, 08:964, 08:999, *227*
__hook_if_structure_exist_p:n 08:2468
__hook_if_usable:n 08:2462
__hook_if_usable:nTF 08:2371, 08:2462, 08:2810, 08:106, 08:518, 08:532, 08:560, 08:589, 08:605, 08:738, 08:796, 08:829, 08:981, 08:1017, 08:1480, 08:1515, 08:1858, 08:1882, 08:1941, 08:1964, 08:1982, 08:2044, *227*
__hook_if_usable_p:n 08:2462, 08:1916, 08:2016
__hook_if_usable_use:n 08:2305, 08:2320, 08:2323, 08:2326, *286*
__hook_include_legacy_code_-chunk:n 08:126, 08:142, 08:229, 08:229, 08:231, 08:249, 08:251, 08:1479, 08:1514
__hook_init_structure:n 08:2159, 08:119, 08:139, 08:146, 08:146, 08:148, 08:158, 08:160, 08:539, 08:613, 08:653, 08:1325, 08:1350, *226*
__hook_initialize_all: 08:2881, 08:1420, 08:1420, 08:1422, 08:1446, 08:1448, 08:1470
__hook_initialize_hook_code:n 08:2214, 08:2235, 08:2264, 08:1424, 08:1451, 08:1473, 08:1473, 08:1475, 08:1507, 08:1509, *258*
__hook_initialize_single:NNn . . . 08:1499, 08:1537, 08:1553, 08:1553, 08:1555, 08:1633, 08:1635, 08:1637, *263*
\l__hook_label_0_tl 08:1548
__hook_label_if_exist_apply:nnnTF 08:1720, 08:1722, 08:1724, 08:1727, *272*
__hook_label_ordered:nn 08:1404, *263*
__hook_label_ordered:nnTF 08:1367, 08:1374, 08:1381, 08:1404, *262*
__hook_label_ordered_p:nn . 08:1404
__hook_label_pair:nn 08:1366, 08:1373, 08:1380, 08:1385, 08:1390, 08:1395, 08:1396, 08:1396, 08:1802, 08:1803, *263*
\l__hook_labels_int 08:1548, 08:1558, 08:1562, 08:1594, 08:1616, 08:1640, 08:1644, 08:1680, 08:1705, *269*
\l__hook_labels_seq 08:1548, 08:1557, 08:1563, 08:1581, 08:1639, 08:1645, 08:1664, 08:1808
__hook_list_if_rule_exists:nnnTF . . 08:2074, 08:2094, 08:2095, 08:2097
__hook_list_one_rule:nnn 08:2074, 08:2085, 08:2086, 08:2092
__hook_list_rules:nn . . 08:2074, 08:2074, 08:2111, 08:1903, 08:2003, *280*
__hook_log:nN 08:1818, 08:1821, 08:1826, 08:1832, 08:1834, 08:1839, 08:1950, 08:1952, 08:1957
__hook_log_cmd:n 08:1820, 08:1825, 08:1829, 08:1831, 08:1843, 08:1961
__hook_log_line:n 08:1818, 08:1828, 08:1859, 08:1861, 08:1865, 08:1879, 08:1891, 08:1901, 08:1919, 08:1938, 08:1965, 08:1967, 08:1971, 08:1979, 08:1993, 08:2001, 08:2019, 08:2040
__hook_log_line_indent:n 08:1818, 08:1830, 08:1869, 08:1875, 08:1885, 08:1892, 08:1906, 08:1914, 08:1973, 08:1976, 08:1986, 08:1994, 08:2006, 08:2014
__hook_log_next_code:n . 08:1998, 08:2053, 08:2053, 08:2058, 08:2060
__hook_log_next_code:w 08:1897, 08:2056

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

__hook_make_name:n 08:357, 08:363, 08:372, 08:378, 08:378, 08:432, 08:475, *236*
__hook_make_name:w 08:378, 08:380, 08:384
__hook_make_prefixes:w 09:154, 09:207, 09:282, 09:303, 09:308
__hook_make_usable:n 08:97, 08:133, 08:318, 08:800, 08:833
__hook_make_usable:nn 09:211, 09:503, 08:81, 08:102, 08:102, 08:104, 08:130, 08:132, 08:300, 08:743, 08:746, *229*
__hook_misused_if_replacing_-
args:nn .. 08:2539, 08:2540, 08:2546
__hook_msg_pair_found:nnn 08:1745, 08:1752, 08:1759, 08:1767, 08:1775, 08:1786, 08:1799, 08:1799
\g__hook_name_stack_seq 08:32, 08:418, 08:419, 08:423, 08:430, 08:444, 08:451, 08:459, 08:469
__hook_new:n ... 08:89, 08:91, 08:193
__hook_new:nn 08:67, 08:70, 08:72, 08:73, 08:90, 08:181, 08:213
__hook_new_pair:nnn 08:202, 08:204, 08:205
__hook_new_reversed:n 08:189, 08:191
__hook_new_reversed:nn 08:170, 08:173, 08:175, 08:176, 08:190, 08:196, 08:214
__hook_next␣⟨*hook*⟩ *226*
__hook_next_gset:nn 08:2164, 08:2189, 08:2398, 08:154, 08:970, 08:1089, 08:1095, 08:1109
\c__hook_nine_parameters_tl 08:35, 08:114, 08:1060, 08:1175, 08:1874
__hook_normalise_code_pool:n 08:120, 08:1136, 08:1136, 08:1138, 08:1188, 08:1190, *229*
__hook_normalise_cs_args:nn 08:116, 08:117, 08:1112, 08:1112, 08:1114, 08:1132, 08:1134, *229*
__hook_normalise_fn:nn 08:564, 08:1142, 08:1161, 08:1186, *255*
__hook_normalize_hook_args:Nn 08:2127, 08:2133, 08:2140, 08:2185, 08:2340, 08:2346, 08:2355, 08:2840, 08:70, 08:72, 08:89, 08:173, 08:175, 08:189, 08:267, 08:291, 08:385, 08:394, 08:1821, 08:1826
__hook_normalize_hook_args:Nnn 08:202, 08:204, 08:385, 08:399, 08:491, 08:497, 08:582, 08:961
__hook_normalize_hook_args_-
aux:Nn 08:385, 08:385, 08:396, 08:401, 08:409
__hook_normalize_hook_rule_-
args:Nnnnn . 08:385, 08:407, 08:1305
\l__hook_param_text_tl 09:8, 09:175, 09:196, 09:250, 09:271, 09:300, *317*
__hook_parameter:n 08:1100, 08:1158, 08:1270, 08:1270, 08:1272, 08:1287, 08:1289, 08:1853
\c__hook_parameter_cmd 08:950, 08:955
\c__hook_parameter_cmd/./after_-
tl 08:950
\c__hook_parameter_cmd/./before_-
tl 08:950
__hook_parse_dot_label:nN 08:334, 08:336, 08:336
__hook_parse_dot_label:w 08:336, 08:348, 08:351
__hook_parse_dot_label_aux:w 08:336, 08:354, 08:362
__hook_parse_dot_label_cleanup:w 08:336, 08:358, 08:361
__hook_parse_label_default:nN 08:330, 08:330, 08:397, 08:403, 08:404, 08:411, 08:412, 08:414
__hook_patch_check:NNnn 09:77, 09:81, 09:84, 09:87, 09:97
__hook_patch_cmd_or_delay:Nnn .. . 09:29, 09:42, 09:46, 09:46, 09:56, *310*
__hook_patch_command:Nnn 09:56, 09:74, 09:77, 09:77, *310*
__hook_patch_debug:n 09:18, 09:18, 09:79, 09:80, 09:83, 09:86, 09:89, 09:158, 09:233, 09:372, 09:407, 09:410, 09:411, 09:416, 09:426, 09:429, 09:430, 09:435
__hook_patch_DeclareRobustCommand:Nnn 09:126, 09:130, 09:130, *312*
__hook_patch_DeclareRobustCommand_-
aux:Nnn 09:132, 09:135
__hook_patch_expand_redefine:NNnn 09:141, 09:146, 09:151, 09:154, 09:154, 09:156, 09:229, 09:231, 09:370, *312*
__hook_patch_newcommand:Nnn 09:127, 09:140, 09:144, 09:144, *312*
\l__hook_patch_num_args_int 09:7, 09:159, 09:164, 09:167, 09:181, 09:211, 09:220, 09:234, 09:239, 09:242, 09:256, 09:500, 09:503, 09:523, 09:530

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\l__hook_patch_prefixes_tl 09:8, 09:206, 09:281, 09:298, *317*
__hook_patch_required_catcodes: 09:441, 09:441, 09:462, 09:538, 09:566, *325*
__hook_patch_retokenize:Nnnn 09:412, 09:431, 09:496, 09:496, 09:498, 09:543, 09:545, *321*
__hook_post_initialization_defs: . 08:2291, 08:2291, 08:2293, 08:2298, 08:2301, 08:2303, 08:1443
__hook_preamble_hook:n 08:2196, 08:2200, 08:2211, 08:2224, 08:2234, 08:2244, 08:2263, 08:2272, 08:2297, 08:2330, 08:2365, 08:2384, 08:1466, 08:1842, 08:1960, *283*
__hook_print_args:n 08:2063
__hook_print_args:nn 08:1850, 08:2063
__hook_prop_gput_labeled_cleanup:nnn . 08:486, 08:545, 08:555
__hook_prop_gput_labeled_do:Nnn 08:569, 08:572
__hook_prop_gput_labeled_do:Nnnn 08:486
\l__hook_rear_tl 08:1548, 08:1579, 08:1585, 08:1586, 08:1609, 08:1610, 08:1662, 08:1670, 08:1671, 08:1698, 08:1699
__hook_redefine_with_hooks:Nnnn .. 09:154, 09:214, 09:286, 09:296, *316*
\l__hook_replace_text_tl 09:8, 09:176, 09:183, 09:184, 09:185, 09:190, 09:197, 09:224, 09:251, 09:258, 09:259, 09:260, 09:265, 09:272, 09:292, *314*
__hook_replacement_spec:N 08:1196, 08:1202
__hook_replacing_args_false: ... 08:2126, 08:2367, 08:2539, 08:2553, 08:237, 08:490, 08:641, 08:1429, *239*
__hook_replacing_args_reset: ... 08:2128, 08:2134, 08:2370, 08:2539, 08:2559, 08:243, 08:492, 08:498, 08:1432
__hook_replacing_args_true: 08:2132, 08:2539, 08:2547, 08:496, 08:1430
\g__hook_replacing_stack_seq 08:2539
__hook_retokenize_patch:Nnn 09:92, 09:367, 09:367
\l__hook_return_tl 08:2561, 08:2562, 08:25, 08:451, 08:452, 08:459, 08:463, 08:557, 08:565, 08:570, 08:574, 08:575, 08:620, 08:623, 08:977, 08:1012, 08:1595, 08:1596, 08:1682, 08:1683
__hook_rollback_tidying: .. 08:2921
__hook_rule_<_gset:nnn 08:1363
__hook_rule_>_gset:nnn 08:1363
__hook_rule_after_gset:nnn 08:1363, 08:1370, 08:1376
__hook_rule_before_gset:nnn 08:1363, 08:1363, 08:1369, *268*
__hook_rule_gclear:nnn 08:1326, 08:1351, 08:1393, 08:1394, *262*
__hook_rule_incompatible-error_gset:nnn 08:1383
__hook_rule_incompatible-warning_gset:nnn 08:1383
__hook_rule_unrelated_gset:nnn 08:1393, 08:1393, *262*
__hook_rule_voids_gset:nnn 08:1377, 08:1377
__hook_seq_csname:n ... 08:1546, 08:1547, 08:1565, 08:1599, 08:1647, 08:1688, 08:1748, 08:1755, 08:1813
__hook_set_default_hook_label:n 08:2773, 08:467, 08:467
__hook_set_default_label:n 08:467, 08:475, 08:477
__hook_set_normalise_fn:nn 08:562, 08:1136, 08:1140, 08:1145, *255*
__hook_str_compare:nn 08:23, 08:23, 08:1398, 08:1406, 08:1415
__hook_strip_double_slash:n 08:899, 08:905, 08:906
__hook_strip_double_slash:w 08:899, 08:907, 08:908, 08:912
__hook_tl_csname:n 08:1546, 08:1546, 08:1552, 08:1564, 08:1580, 08:1583, 08:1585, 08:1589, 08:1601, 08:1603, 08:1606, 08:1609, 08:1614, 08:1646, 08:1663, 08:1667, 08:1669, 08:1674, 08:1690, 08:1692, 08:1695, 08:1697, 08:1703, 08:1746, 08:1747, 08:1753, 08:1754, 08:1812
__hook_tl_gclear:N 08:2407, 08:2408, 08:2409, 08:64, 08:64, 08:66, 08:244, 08:259, 08:1004, 08:1005, 08:1009, 08:1590, 08:1675
__hook_tl_gput:Nn 08:1494, 08:1496, 08:1527, 08:1531, 08:1596, 08:1623, 08:1683, 08:1712, 08:1718, 08:1718, *270*
__hook_tl_gput_left:Nn 08:58, 08:58, 08:1494, 08:1528
__hook_tl_gput_right:Nn 08:2181, 08:55, 08:55, 08:57, 08:614, 08:1496, 08:1532, 08:1625, 08:1714, 08:1716

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

__hook_tl_gset:Nn 08:2180, 08:2394, 08:2932, 08:49, 08:49, 08:51, 08:53, 08:54, 08:56, 08:60, 08:1365, 08:1372, 08:1379, 08:1384, 08:1389, 08:1519

__hook_tl_gset_eq:NN 08:63, 08:63, 08:65

__hook_tl_set:Nn 08:47, 08:47, 08:1151, 08:1564, 08:1646, *224*

__hook_tmp:w 08:2078, 08:2100, 08:2109, 08:2120, 08:2930, 08:2947, 08:2948, 09:173, 09:179, 09:180, 09:182, 09:248, 09:254, 09:255, 09:257, 09:453, 09:456, 09:460, 09:512, 09:515, 09:536, 09:548, 09:551, 09:564, 08:34, 08:34, 08:420, 08:424, 08:426, 08:1147, 08:1158, 08:1174, 08:1180, 08:1183, 08:1873, 08:1876, *324*

\l__hook_tmpa_bool 08:24, 08:1902, 08:1905, 08:1913, 08:1922, 08:2002, 08:2005, 08:2013, 08:2022, *276*

\l__hook_tmpa_tl .. 09:166, 09:168, 09:170, 09:241, 09:243, 09:245, 09:373, 09:388, 09:391, 09:459, 09:462, 09:504, 09:508, 09:524, 09:531, 09:535, 09:538, 09:563, 09:566, 08:25, 08:430, 08:1159, 08:1181

\l__hook_tmpb_tl .. 09:378, 09:389, 09:392, 08:25, 08:1148, 08:1155, 08:1183

__hook_toplevel␣⟨*hook*⟩ *251*

__hook_toplevel_gset:nn 08:2399, 08:153, 08:969, 08:974, 08:1089, 08:1093, 08:1108

__hook_try_declaring_generic_-hook:nnn 08:526, 08:597, 08:656, 08:657, 08:659, 08:675, 08:677, 08:696, 08:699, *243*

__hook_try_declaring_generic_-hook:nNNnn 08:701, 08:707, 08:710, 08:710, *243*

__hook_try_declaring_generic_-hook:wn . 08:730, 08:733, 08:788, 08:791, 08:818, 08:821, 08:850, 08:853

__hook_try_declaring_generic_-hook:wnTF 08:661, 08:668, 08:679, 08:688, 08:723, 08:729, 08:782, *246*

__hook_try_declaring_generic_-hook_split:nNNnn 08:710, 08:715, 08:718, 08:721

__hook_try_declaring_generic_-next_hook:nn 08:2151, 08:656, 08:666, 08:686, 08:705, *243*

__hook_try_file_hook:n 08:2305, 08:2313, 08:2316, *286*

__hook_try_patch_with_catcodes:Nnnnw ... 09:385, 09:402, 09:402, 09:404, 09:417, 09:421, 09:423, 09:436, *320*

__hook_try_put_cmd_hook:n 09:21, 09:21, 09:23, 09:32, 09:34, 08:742, 08:799, 08:832

__hook_try_put_cmd_hook:w 09:21, 09:24, 09:25, 09:35, 09:36

__hook_unpatchable_cases:n 09:591, 09:593

__hook_update_hook_code:n 08:2162, 08:2178, 09:226, 09:540, 08:82, 08:303, 08:321, 08:521, 08:592, 08:982, 08:1018, 08:1330, 08:1355, 08:1419, 08:1419, 08:1424, 08:1431, 08:1450, 08:1454, *264*

__hook_use:wn .. 08:2247, 08:2261, 08:2305, 08:2305, 08:2308, 08:2310, *286*

__hook_use_end: 08:2255, 08:2258, 08:2265

__hook_use_i_delimit_by_s_-mark:nw 08:45, 08:46, 08:1222

__hook_use_initialized:n 08:2196, 08:2201, 08:2203, 08:2228, 08:2249, 08:2295, 08:2372, 08:2386, 08:1465, *283*

__hook_use_initialized:nnw 08:2268, 08:2273, 08:2275, 08:2296

__hook_use_none_delimit_by_s_-mark:w 08:2415, 08:2482, 09:489, 08:45, 08:45, 08:1317, 08:1323

__hook_use_once:n .. 08:2355, 08:2382

__hook_use_once:nn 08:2340, 08:2346, 08:2361, 08:2361, 08:2363, 08:2380, 08:2389

__hook_use_once_clear:n 08:2369, 08:2387, 08:2391, 08:2391, 08:2395, 08:2403, 08:2405, *288*

__hook_use_once_set:n 08:2366, 08:2385, 08:2391, 08:2393, *288*

__hook_use_undefined:w 08:2253, 08:2257

\g__hook_used_prop 08:31, 08:1425, 08:1437, 08:1452, 08:1458, 08:1502, 08:1540

\l__hook_work_prop 08:30, 08:565, 08:1141, 08:1143, 08:1177, 08:1498, 08:1535, 08:1560, 08:1567, 08:1569, 08:1578, 08:1595, 08:1620, 08:1642, 08:1649, 08:1651, 08:1661, 08:1681, 08:1709, 08:1780, 08:1791, *268*

hook␣?? internal commands:

__hook␣?? *260*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

hook␣⟨*hook*⟩ internal commands:
__hook␣⟨*hook*⟩ *265*
\hookleftarrow 30:487
\hookrightarrow 30:485
hook ?? internal commands:
__hook~?? 08:1292
\hphantom 38:75
\hrule 18:397, 18:405, 18:431, 18:439, 21:289, 21:294, 30:347, 30:625, 40:202, 40:207, 40:280, 40:290, 41:359, 41:376, 42:589, 42:599, 45:395, 02:502, 02:546, *1268*
\hrulefill 06:884, 06:905, 02:546
\hsize *1237*
\hskip *1224*
\hskip... . . . . *1233*
\hspace 18:512, 18:516, 18:522, *439*
\hss *1079*
\Hwithstroke 21:494, 21:1213, *1248*
\hwithstroke 21:510, 21:1214, *1248*
\hyphenation 21:205, *1220*
\hyphenchar 37:564, 06:842, 06:845, 06:854, 06:857, 06:860, 06:865, *1246*
\hyphenpenalty 24:687, 24:719, 02:321

I

\I 50:1335, 50:1467, 50:1557, 50:1577, 57:255, 57:550, 02:437
\i 21:247, 21:402, 21:452, 21:453, 21:454, 21:455, 21:456, 21:457, 21:547, 21:587, 21:588, 21:680, 21:682, 21:684, 21:686, 21:778, 21:1140, 21:1295, 21:1297, 21:1299, 21:1301, 21:1352, 21:1355, 21:1358, 21:1361, 21:1431, 57:259, 57:554, *1239*
\ialign 30:344, 30:466, 30:537, 30:540, 30:544, 30:547, 38:168, 38:170, 38:189, 41:191, 42:141, 02:534, 02:536, *1198*
\IeC 57:357, 57:361, 57:468
\if *1232*
if commands:
\if:w 08:2416, 08:2430, 08:2483, 08:2515, 09:343, 08:1036, 08:1196, 08:1221, *288*
\if_case:w 07:1458, 08:1247, 08:1398, 08:1415
\if_catcode:w 09:352, *319*
\if_charcode:w 08:2533, 07:946
\if_cs_exist:w 08:2099, 08:2205, 08:2213, 08:2230, 08:2251, 08:2278, 08:2417, 08:767, 08:1729
\if_false: 07:876, 07:879, 07:2399, 07:2403
\if_int_compare:w 16:101, 08:1406
\if_meaning:w 09:340, 09:344, 09:345, 52:241, 07:1303
\if_mode... . . . . *413*
\if_mode_horizontal: . . . 16:32, 16:64
\if... . . . . *1027*
\IfBlankF 07:2987
\IfBlankT 07:2987
\IfBlankTF 47:19, 07:2987, *1261*
\IfBold *650*
\IfBooleanF 22:130, 22:138, 07:2962
\IfBooleanT 07:2962
\IfBooleanTF 17:32, 35:67, 35:69, 35:156, 07:2962, *191*
\IfClassAtLeastF 50:283
\IfClassAtLeastT 50:283
\IfClassAtLeastTF *977*
\IfClassAtLeastTF 50:165, 50:293, 50:294, 50:322, 50:323
\IfClassLoadedF 50:283
\IfClassLoadedT 50:283
\IfClassLoadedTF *977*
\IfClassLoadedTF 50:263, 50:289, 50:290, 50:318, 50:319
\IfClassLoadedWithOptionsF 50:283
\IfClassLoadedWithOptionsT 50:301, 50:330
\IfClassLoadedWithOptionsTF *977*
\IfClassLoadedWithOptionsTF 50:263, 50:283
\ifcoremisses 33:1222, 33:1413
\ifcsname 24:340, 25:440, 25:443, 25:653, 25:656, 26:128, 28:153, 29:42, 29:53, 29:75, 29:86, 33:1248, 50:1220, 50:1355, 53:433, 06:740, 06:757, 54:1907, *569*
\ifdefined 02:570, 02:634, 29:623, 05:22, 05:71, 05:72, 05:73, 05:74, 05:117, 05:118, 05:119, 57:319
\IfDocumentMetadataTF 17:5, 17:17
\IfExplAtLeastTF *78*
\IfExplAtLeastTF . . 05:198, 05:205, 05:207
\iff 30:507
\IfFileAtLeastF 50:283
\IfFileAtLeastT 50:283
\IfFileAtLeastTF *977*
\IfFileAtLeastTF 50:165, 50:295, 50:296, 50:324, 50:325, *1264*
\IfFileExists *442*, *978*
\IfFileExists 20:469, 20:481, 20:617, 20:673, 20:698, 01:162, 05:37, 05:68, 05:114, 50:925, 52:149, 52:185, 52:195, 57:651, *1205*
\IfFileLoadedF 50:303
\IfFileLoadedT 50:303

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\IfFileLoadedTF 50:303, *1268*
\iffontchar 21:866, 21:1060, 21:1165, 21:1167, 21:1169, 21:1216, 33:1218, 33:1219
\IfFontSeriesContextTF 29:496, 29:532, 29:534, *650*
\IfFormatAtLeastF 50:283
\IfFormatAtLeastT 50:283
\IfFormatAtLeastTF *977*
\IfFormatAtLeastTF 50:165, 50:297, 50:298, 50:326, 50:327, *1047*
\IfHookEmptyF 08:2807, *205*
\IfHookEmptyT 08:2807, *1270*
\IfHookEmptyTF 08:2807, 08:2912, 37:349, 37:366, *200*
\IfHookExistsTF . . . 08:2810, 08:2911, *299*
\ifincsname 18:475, *1033*
\ifinner 18:239, 18:290, 38:277, 38:285, 38:305, 38:332, 45:57, 45:126, 45:315
\IfInstanceExistsF 11:1186, *348*
\IfInstanceExistsT 11:1186, *348*
\IfInstanceExistsTF 11:1186, *348*
\IfLabelExistsF 36:181, *1270*
\IfLabelExistsT 36:181, *750*
\IfLabelExistsTF . . . 36:181, 36:275, *1268*
\IfLabelExistTF *1268*
\IfMarksEqualF 48:372, *945*
\IfMarksEqualT 48:372, *945*
\IfMarksEqualTF 48:372, 48:487, *945*
\ifmmode *1193*
\IfNoValueF 07:2981
\IfNoValueT 07:2981
\IfNoValueTF 36:120, 57:629, 57:636, 57:643, 07:2981
\ifnum *1064*
\ifodd 28:1156, 42:320, 42:344, 42:378, 42:400, 45:68, 45:137, 54:21, 54:140, 54:614, 54:672, 54:986, 54:989, 54:1022, 54:1025, 54:1136, 54:1139, 54:1298, 54:1301, 54:1578, 54:1581, 54:1699, 54:1702, 54:1822, 54:2077, 54:2085
\IfPackageAtLeastF 50:283
\IfPackageAtLeastT 50:283
\IfPackageAtLeastTF *977*
\IfPackageAtLeastTF 50:165, 50:291, 50:292, 50:320, 50:321
\IfPackageLoadedF 50:283
\IfPackageLoadedT 50:283
\IfPackageLoadedTF *977*
\IfPackageLoadedTF 50:263, 50:287, 50:288, 50:316, 50:317
\IfPackageLoadedWithOptionsF . . . 50:283
\IfPackageLoadedWithOptionsT . . . 50:283
\IfPackageLoadedWithOptionsTF *977*
\IfPackageLoadedWithOptionsTF 50:263, 50:299, 50:300, 50:328, 50:329
\IfPDFManagementActiveTF . . . 57:710, *1192*
\IfPropertyExistsF 36:166, *750*
\IfPropertyExistsT 36:166, *750*
\IfPropertyExistsTF . . 36:166, 36:274, *750*
\IfPropertyExistTF *1268*
\IfPropertyRecordedF 36:196, *1270*
\IfPropertyRecordedT 36:196, *750*
\IfPropertyRecordedTF 36:196, 36:276, *749*
\ifsafesubencodingfound 33:1221, 33:1235, 33:1240, 33:1409
\IfTargetDateBefore 50:1793
\ifthenelse *741*
\IfValueF 07:2984
\IfValueT 07:2984
\IfValueTF 07:2984
\ifvbox 54:321, 54:378, 54:425, 54:506, 54:522
\ifvoid *1234*
\ifx *1251*
\ignorespaces . . 18:50, 18:144, 18:163, 18:175, 18:186, 18:202, 18:215, 18:559, 20:71, 20:139, 20:194, 21:72, 24:292, 24:302, 24:312, 24:358, 28:301, 28:340, 37:339, 37:356, 37:372, 37:381, 37:430, 37:435, 37:441, 38:313, 38:340, 38:445, 38:446, 39:55, 39:241, 40:193, 40:490, 40:507, 40:524, 41:57, 41:62, 41:68, 41:83, 41:92, 41:105, 41:109, 41:116, 41:123, 41:125, 41:134, 41:154, 41:237, 41:301, 41:303, 41:305, 41:332, 42:36, 42:46, 42:66, 42:75, 42:109, 42:120, 42:131, 42:143, 42:148, 42:154, 43:43, 43:45, 44:110, 45:17, 45:24, 45:477, 45:495, 45:513, 47:7, 47:9, 53:330, *1230*
\ignorespacesafterend 37:7
\IJ 21:250, 21:436, 21:550, 21:1141, 57:650, *1241*
\ij 21:249, 21:434, 21:549, 21:1142, 57:650, *1259*
\Im 30:321
\imath 30:316
\immediate *1196*
\in 30:436, 30:468
in commands:
  in_callback 04:959
\in_callback *46*
\include *442*
\include 20:218, 20:266, 20:268, 20:284, 20:286, 20:345, 20:397, *1039*
include/.../after *1039*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

`include/.../before` *1039*
`include/.../end` *1039*
`include/after` *1039*
`include/before` *1039*
`include/end` *1039*
`\IncludeInRelease` . . 08:2122, 08:2137, 08:2155, 08:2168, 08:2186, 08:2191, 08:2196, 08:2218, 08:2238, 02:581, 08:2268, 08:2286, 08:2291, 08:2301, 08:2305, 08:2308, 08:2335, 08:2350, 08:2361, 08:2380, 08:2391, 08:2403, 08:2426, 08:2443, 08:2509, 08:2525, 02:615, 08:2582, 08:2611, 02:622, 08:2719, 08:2728, 08:2734, 08:2741, 08:2746, 08:2753, 08:2757, 08:2764, 08:2784, 08:2791, 02:648, 08:2885, 09:21, 09:32, 09:154, 02:688, 02:693, 09:229, 02:703, 09:402, 09:421, 09:467, 09:492, 09:496, 09:543, 09:570, 09:573, 09:603, 10:179, 03:71, 11:1196, 03:180, 03:184, 03:188, 03:192, 14:138, 14:158, 14:192, 14:205, 16:12, 16:45, 16:167, 17:20, 17:44, 17:58, 18:5, 18:22, 04:3, 18:55, 18:66, 18:82, 18:87, 18:99, 18:105, 18:133, 18:153, 18:167, 18:179, 18:192, 18:207, 18:236, 18:251, 18:267, 18:286, 18:320, 18:354, 18:380, 18:413, 18:471, 18:482, 18:489, 18:495, 18:501, 18:507, 18:515, 18:521, 18:535, 18:541, 20:10, 20:84, 20:142, 20:220, 20:257, 20:278, 20:293, 20:354, 20:403, 20:464, 20:474, 20:500, 20:532, 20:554, 20:572, 20:582, 20:598, 20:623, 20:628, 20:636, 20:650, 20:661, 20:677, 20:710, 21:77, 21:104, 21:148, 21:169, 21:324, 21:332, 21:353, 21:369, 04:235, 04:258, 04:281, 22:24, 22:30, 22:38, 22:59, 22:78, 22:88, 22:94, 22:117, 22:125, 22:147, 22:174, 22:209, 22:225, 22:233, 22:251, 23:5, 23:11, 23:19, 23:26, 24:24, 24:53, 24:210, 24:232, 24:284, 24:295, 24:305, 24:354, 24:367, 24:445, 24:456, 24:464, 24:496, 24:504, 24:518, 24:539, 24:575, 24:662, 24:725, 25:2, 25:385, 25:392, 25:404, 25:416, 25:424, 25:506, 25:522, 25:600, 25:613, 25:623, 25:630, 25:638, 25:710, 25:729, 25:735, 25:739, 26:113, 26:158, 26:161, 26:541, 26:550, 27:2, 27:22, 28:49, 28:78, 28:138, 28:180, 28:210, 28:241, 28:275, 28:307, 28:398, 28:411, 28:417, 28:457, 28:505, 28:532, 28:557, 28:790, 28:838, 28:884, 28:893, 28:1077, 28:1088, 29:33, 29:68, 29:101, 29:126, 29:228, 29:252, 29:301, 29:348, 29:377, 29:395, 29:435, 29:469, 29:498, 29:531, 29:543, 29:577, 29:590, 29:621, 29:630, 29:655, 29:672, 29:688, 29:705, 30:71, 30:89, 30:108, 30:117, 30:629, 30:641, 32:27, 32:34, 33:607, 05:2, 05:11, 05:20, 35:12, 35:29, 35:46, 35:62, 35:76, 35:96, 35:111, 35:115, 35:139, 35:148, 35:159, 35:169, 35:181, 35:192, 05:58, 36:264, 37:10, 37:76, 37:140, 37:182, 37:187, 05:112, 37:199, 37:226, 37:245, 37:267, 37:295, 37:312, 37:334, 37:345, 37:361, 37:375, 37:383, 05:133, 37:394, 37:402, 37:415, 37:421, 37:433, 37:438, 37:448, 37:467, 37:484, 37:500, 37:511, 05:148, 37:525, 37:547, 37:574, 37:590, 37:603, 37:617, 37:629, 37:635, 37:641, 37:648, 37:666, 37:679, 37:695, 37:718, 37:728, 05:159, 38:79, 38:87, 05:169, 38:109, 38:118, 38:137, 38:144, 38:152, 38:157, 38:165, 38:176, 38:216, 38:233, 38:272, 05:178, 38:280, 38:290, 38:317, 38:399, 05:192, 38:408, 38:441, 38:449, 38:462, 38:474, 38:486, 05:199, 38:495, 05:204, 06:5, 06:13, 39:125, 39:135, 39:143, 39:155, 40:4, 40:14, 40:42, 40:58, 40:71, 40:91, 40:112, 40:122, 40:136, 40:142, 40:147, 40:156, 40:164, 40:220, 40:228, 40:241, 40:261, 40:299, 40:308, 40:327, 40:355, 40:378, 40:400, 40:477, 40:495, 40:511, 40:531, 40:537, 40:557, 40:565, 40:595, 40:606, 06:56, 06:62, 41:60, 41:65, 41:137, 41:157, 41:222, 41:227, 42:10, 42:16, 42:26, 42:39, 42:55, 42:69, 42:80, 42:89, 42:100, 42:112, 42:146, 42:151, 42:160, 42:172, 42:235, 42:250, 42:311, 42:369, 42:455, 42:463, 42:472, 42:500, 42:530, 42:557, 42:571, 42:584, 42:595, 42:607, 42:624, 42:644, 42:651, 42:685, 42:745, 42:803, 42:819, 01:274, 43:28, 43:34,

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

43:50, 43:57, 44:5, 44:20, 44:161, 44:168, 44:174, 44:186, 44:193, 44:202, 44:225, 45:35, 45:105, 06:218, 06:225, 45:206, 45:232, 45:280, 45:294, 45:335, 45:352, 45:406, 45:412, 45:420, 45:427, 45:434, 45:440, 45:446, 45:463, 45:482, 45:499, 45:549, 45:559, 47:14, 47:25, 47:61, 47:77, 01:21, 06:284, 01:291, 06:326, 48:479, 49:20, 49:48, 49:69, 50:18, 50:23, 50:36, 06:354, 50:51, 50:63, 50:87, 50:95, 50:101, 50:126, 50:144, 50:167, 50:175, 50:187, 50:200, 50:226, 50:245, 50:265, 50:273, 50:285, 06:382, 50:313, 50:345, 50:361, 50:375, 06:389, 50:393, 50:406, 50:422, 50:442, 50:465, 50:481, 50:510, 50:524, 06:406, 50:556, 50:574, 50:594, 50:612, 50:631, 50:641, 50:651, 50:663, 50:695, 06:422, 50:706, 50:720, 50:730, 50:776, 50:790, 50:805, 50:835, 50:862, 50:894, 50:1046, 50:1126, 50:1133, 50:1144, 50:1158, 50:1208, 06:460, 06:472, 50:1343, 50:1474, 06:502, 06:513, 06:538, 52:5, 52:15, 52:27, 06:546, 52:89, 52:144, 52:181, 52:192, 06:558, 52:205, 52:249, 52:257, 06:565, 52:268, 52:299, 52:322, 52:343, 52:357, 52:365, 52:388, 52:408, 52:433, 52:479, 52:495, 52:506, 52:533, 53:3, 53:430, 53:470, 53:484, 54:24, 54:54, 06:664, 54:153, 54:181, 06:681, 01:26, 54:347, 54:368, 54:373, 54:421, 54:595, 54:655, 54:798, 54:816, 06:730, 54:877, 54:898, 06:736, 54:934, 54:958, 54:1070, 06:766, 54:1221, 54:1390, 54:1472, 54:1566, 54:1688, 06:807, 54:1902, 54:1925, 54:1939, 54:1967, 06:826, 06:837, 54:2198, 54:2216, 54:2235, 54:2281, 06:854, 06:865, 06:876, 06:896, 55:244, 57:8, 57:16, 57:23, 57:30, 57:37, 57:52, 57:71, 57:80, 57:87, 57:106, 57:143, 57:166, 57:199, 57:288, 57:293, 57:314, 57:339, 57:348, 57:441, 02:49, 07:224, 07:238, 02:88, 02:103, 02:119, 02:125, 02:134, 02:139, 02:148, 07:1091, 02:154, 07:1143, 07:1146, 07:1161, 07:1174, 07:1189, 02:168, 07:1307, 07:1310, 02:182, 07:1339, 07:1342, 07:1351, 07:1359, 07:1362, 02:186, 07:1523, 02:220, 02:228, 02:233, 02:244, 07:1978, 07:1994, 02:289, 07:2840, 07:2893, 02:351, 07:2987, 07:2993, 07:3009, 02:361, 08:67, 08:86, 08:102, 08:130, 08:146, 08:158, 08:170, 08:186, 08:199, 08:219, 08:229, 08:249, 08:264, 08:282, 08:288, 08:308, 08:326, 08:486, 08:579, 02:454, 08:630, 08:647, 08:656, 08:674, 08:695, 08:729, 08:787, 08:817, 08:849, 08:876, 08:879, 08:899, 08:902, 08:915, 08:933, 08:938, 08:941, 08:950, 08:955, 08:958, 08:995, 02:472, 08:1032, 08:1082, 08:1089, 08:1105, 08:1112, 08:1132, 08:1136, 08:1188, 08:1192, 08:1208, 08:1212, 08:1240, 08:1243, 08:1266, 08:1270, 08:1287, 08:1295, 08:1299, 08:1308, 08:1339, 08:1420, 08:1446, 08:1473, 08:1507, 08:1553, 08:1635, 08:1832, 08:1950, 02:557, 08:2053, 08:2058, *1257*

`\includeonly` *442*

`\includeonly` 20:218, 20:258, 20:260, 20:279, 20:280, *1039*

`\indent` 18:538, 39:185, 41:81, *1233*

`\IndentBox` 16:91, 16:171, *406*

`\index` *934*

`\index` 44:182, 46:6, 46:18, 48:252, 49:25, 49:36, 49:53, 49:61, 54:624, 54:683, *943*

`\indexentry` 46:15

`\inf` 38:25

`\infty` 30:323

`\initcatcodetable` 04:91

`\input` *442*, *978*

`\input` . . . 04:18, 20:633, 01:158, 01:161, 26:16, 27:106, 29:751, 29:761, 29:771, 30:10, 30:11, 30:12, 30:13, 30:14, 30:23, 30:41, 30:42, 30:46, 30:47, 30:48, 30:49, 30:54, 30:143, 30:144, 30:145, 30:146, 30:661, 30:662, 30:663, 01:218, 33:1126, 05:108, 05:130, 06:22, 50:680, 57:164, 57:178, 57:203, 57:281, 57:325, 57:405, 57:656, 01:52, *1228*

`\input@path` *6*, *1*

`input@path` commands:

`\input@path:` 01:223

`\inputencodingname` 57:382, 57:404, 57:486, *1247*

`\InputIfFileExists` *442*, *978*

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

\InputIfFileExists 17:7, 20:616, 20:639, 20:654, 20:664, 20:674, 20:728, 21:1546, 24:416, 29:743, 29:753, 29:763, 33:842, 33:1209, 50:1011, 50:1077, 52:143, 56:8, 57:275, *1205*
\inputlineno 14:214, 01:311, *1207*
\insert 45:465, 45:484, 45:502, 54:521, 54:522, 54:1887, 02:206, 02:254, 02:279, 02:281, 02:284, 02:299, 02:332, *402*
\InsertMark 48:363, 48:483, *222*
insertmark 48:249, *943*
\int 30:355
int commands:
\int_add:Nn 07:607, 07:1252
\int_case:nnTF 55:19, 55:29
\int_compare:nNnTF 08:2374, 08:2571, 09:164, 09:239, 09:375, 09:487, 10:39, 11:45, 28:287, 28:326, 48:79, 48:254, 53:55, 53:111, 53:142, 53:353, 53:359, 53:367, 55:136, 55:145, 55:168, 55:187, 55:190, 55:215, 55:222, 55:233, 07:457, 07:602, 07:873, 07:915, 07:1934, 07:1987, 07:2255, 07:2257, 08:1583, 08:1605, 08:1616, 08:1666, 08:1694, 08:1705, 08:2065, 08:2071
\int_compare:nTF 11:191
\int_compare_p:nNn . . 08:2918, 08:2919
\int_const:Nn 10:37
\int_decr:N 07:536, 07:582, 07:1933, 08:1594, 08:1680
\int_div_truncate:nn 07:1401
\int_eval:n 11:362, 05:163, 53:355, 07:2269, 08:1248, 08:1602, 08:1691, 08:1747, 08:1754, 08:1852
\int_gadd:Nn 55:125, 55:228
\int_gdecr:N 52:559, 55:199
\int_gincr:N 17:37, 48:255, 52:549, 53:87, 53:102, 55:179, 55:196, 07:1075
\int_gset:Nn 28:292, 28:331, 52:545, 55:138, 55:183, 55:205, 55:206, 55:225
\int_gzero:N 48:256, 55:174, 55:175, 55:180, 55:193, 55:202
\int_if_odd:nTF 48:403, 48:462
\int_if_zero:nTF 10:33
\int_incr:N 07:360, 07:480, 07:887, 07:957, 07:964, 07:1219, 07:1519, 07:1943, 08:1562, 08:1644, *149*
\int_new:N 09:7, 11:34, 17:23, 48:249, 52:544, 53:346, 53:348, 55:38, 55:100, 55:101, 55:116, 07:15, 07:27, 07:36, 07:1913, 08:1549
\int_set:Nn . 09:159, 09:202, 09:234, 09:277, 09:447, 09:448, 09:478, 11:190, 52:212, 52:226, 07:1097, 07:1108, 07:1316, 07:1326, 07:1444, 07:1917, 07:2415, 07:2441, 08:1164
\int_set_eq:NN 48:68
\int_step_function:nN 07:1399
\int_step_inline:nnn 09:167, 09:181, 09:242, 09:256
\int_use:N 09:220, 09:523, 09:530, 10:58, 10:84, 10:103, 17:38, 28:281, 28:282, 28:284, 28:287, 28:314, 28:315, 28:317, 28:326, 36:233, 36:243, 36:244, 48:104, 48:106, 48:225, 52:553, 53:104, 53:116, 53:363, 53:371, 55:20, 55:30, 55:149, 55:159, 55:160, 55:211, 55:212, 07:25, 07:918, 07:1079, 07:1108, 07:1237, 07:1241, 07:1242, 07:1326, 07:1510, *1064*
\int_value:w 53:48, 53:137
\int_zero:N 55:45, 07:351, 07:445, 07:782, 07:786, 07:922, 07:1211, 08:1558, 08:1640
\c_max_int 48:68, 53:214, 53:240, 53:265, 53:292
\l_tmpa_int *346*
\c_zero_int 09:164, 09:239, 53:91
int internal commands:
\g__mark_int 48:225, 48:249, 48:254, 48:255, 48:256
\interdisplaylinepenalty 18:14, 38:55, 38:207, 38:393
\interfootlinepenalty 02:427
\interfootnotelinepenalty 18:19, 45:467, 45:486, 45:504, 02:427
\interlinepenalty 18:11, 24:689, 37:536, 37:539, 37:557, 37:560, 44:67, 44:118, 44:209, 44:232, 45:467, 45:486, 45:504, 54:340, 54:1157, 54:1161, 54:1323, 54:1327, 02:331, *1195*
\inteval *77*
\inteval 05:158, 05:173, *77*
\intextsep . 54:1140, 54:1144, 54:1159, 54:1162, 54:1169, 54:1302, 54:1308, 54:1325, 54:1328, 54:1337, 54:2341
\intop 30:354, 30:355
\iota 30:283
iow commands:
\iow_char:N 08:2594, 08:2596, 08:2598, 08:2603, 08:2606, 08:2608, 08:2618, 08:2619, 08:2659, 08:2667, 08:2679, 08:2684, 08:2689, 08:2875, 08:2877,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

09:581, 16:153, 16:160, 16:161, 16:162, 16:163, 07:468, 07:532, 07:563, 07:635, 07:704, 07:716, 07:747, 07:1118, 07:1124, 07:1918, 07:2722, 07:2800, 08:1777, 08:1788
\iow_log:n 08:1820
\iow_newline: 52:557, 07:1369, 07:1389, 07:1398, 07:1416, 07:1432, 07:1517
\iow_now:Nn 53:362
\iow_term:n 08:2108, 08:2113, 09:19, 09:28, 09:39, 09:48, 09:50, 09:66, 09:70, 10:25, 48:20, 48:90, 48:164, 48:170, 48:176, 48:229, 52:550, 52:583, 07:1432, 08:531, 08:604, 08:1436, 08:1439, 08:1457, 08:1460, 08:1478, 08:1512, 08:1598, 08:1618, 08:1619, 08:1621, 08:1685, 08:1707, 08:1708, 08:1710, 08:1776, 08:1787, 08:1801, 08:1807, 08:1808, 08:1809, 08:1812, 08:1816, 08:1825
\ishortstack 42:132
\itdefault 25:697, 29:30, 30:102
\item . . . 14:282, 37:444, 37:494, 37:496, 37:527, 37:549, 38:469, 38:481, 38:508, 39:165, 39:243, 41:78, 43:53, 43:55, 43:60, 43:62, 47:4, 47:8, *1202*
\itemindent 39:9, 39:42, 39:95, 39:211, 39:232, *803*
\itemitem *1195*
itemize (env.) 39:266
\itemize 39:266
\itemsep 39:1, 39:200, *803*
\iterate 01:65, 01:66, 02:490
\itshape 21:447, 21:807, 25:695, 25:696, 29:28, 29:29, 29:554, 29:587, 29:593, 32:21, 33:634, 43:53, 43:55, 43:60, 43:62, 45:399, *1240*

J

\J 57:257, 57:552
\j 21:248, 21:403, 21:548, 21:779, 21:1150, 21:1365, 21:1451
\jmath 30:317
\jobname *1046*
\Join 29:725
\joinrel 30:478, 30:485, 30:487, 30:489, 30:491, 30:493, 30:495, 30:497, 30:499, 30:503, 30:505
\jot 38:53, 38:204, 38:404, 38:414

K

\k 21:485, 21:590, 21:595, 21:617, 21:622, 21:698, 21:699, 21:757, 21:758, 21:812, 21:814, 21:819, 21:821, 21:1251, 21:1319, 21:1320, 21:1337, 21:1338, 21:1360, 21:1361, 21:1362, 21:1415, 21:1416, 21:1449, 21:1450, 33:181, 33:204, *1240*
\kanjiskip 05:74
\kappa 30:284
\ker 38:27
\kern *1076*
\kern... . . . . *1233*
kernel (refstepcounter/target) (plug) 35:119
kernel (tagsupport/recordtarget) (plug) 55:50
kernel internal commands:
__kernel_chk_if_free_cs:N 07:3006, *193*
__kernel_cmd_if_xparse:NTF 09:128, 07:1128, 07:1439, 07:1441, 07:2473, 07:2494, *144*
__kernel_cs_parameter_spec:N 09:161, 09:236, 09:369, 05:142, 07:2477, *1268*
__kernel_cs_parm_from_arg_count:nnTF 08:109
__kernel_exp_not:w 08:48, 08:50, 08:56, 08:61
\l__kernel_expl_bool 09:380
__kernel_file_name_sanitize:n 52:98
__kernel_msg... . . . . *1257*
__kernel_msg_... . . . . *1259*
__kernel_quark_new_conditional:Nn 11:41
__kernel_quark_new_test:N . . . 07:48
\g__kernel_target_int 17:23, 17:37, 17:38
\kerneltmpDoNotUse . . . 09:440, 09:452, 09:458, 09:463, 09:511, 09:518, 09:539, 09:547, 09:554, 09:567, *323*
keys commands:
\keys_define:nn . . 11:505, 11:526, 11:587, 20:744, 51:70, 51:79, 51:177, 51:185, 51:189, 51:206, 51:241, 57:572
\keys_if_exist:nnTF 51:68, 51:118, 51:124, 51:148
\l_keys_key_str 51:48, 51:49, 51:75, 51:194
\l_keys_key_tl 51:48
\l_keys_path_str 51:47
\keys_set:nn 51:83, 51:256, 57:590
\keys_set_known:nnN 11:1195
\l_keys_usage_load_prop 51:202
\l_keys_usage_preamble_prop . 51:237

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

keys internal commands:
\l__keys_class_only_clist 51:54, 51:121, 51:150
__keys_find_key_module:wNN .. 51:46
\l__keys_forced_global_clist ... 51:49, 51:55, 51:152
\l__keys_no_value_bool 51:24
__keys_option_end: 51:64, 51:78, 51:86
__keys_options:n 51:58, 51:58, 51:198
__keys_options_aux:n 51:58, 51:59, 51:60
__keys_options_class:n 51:100, 51:104, 51:104
__keys_options_class:nn 51:104, 51:120, 51:125, 51:132
__keys_options_class:nnn 51:104, 51:110, 51:116
\l__keys_options_clist 51:56, 51:65, 51:82, 51:135, 51:160, 51:169, *1030*
__keys_options_end: 51:58
__keys_options_expand_module:Nn 51:59, 51:178, 51:178, 51:185, 51:189, 51:256
__keys_options_expand_module:nN 51:178, 51:182
__keys_options_global:n 51:66, 51:95, 51:95
__keys_options_loaded:n 51:87, 51:200, 51:200
__keys_options_loaded:nn 51:200, 51:209, 51:214
\l__keys_options_loading_bool 51:57, 51:81, 51:84, 51:216, *1029*
__keys_options_local: 51:67, 51:163, 51:163
__keys_options_package:n 51:101, 51:137, 51:137
__keys_options_package:nn 51:137, 51:153, 51:155, 51:158
__keys_options_package:nnn 51:137, 51:141, 51:146
\c__keys_props_root_str 51:9, 51:10, 51:22
__keys_remove_equals:n 51:111, 51:142, 51:174, 51:174
__keys_remove_equals:w 51:174, 51:175, 51:176
__keys_scope:N 51:22, 51:33, 51:35, 51:44
__keys_scope:n 51:22, 51:25, 51:26, 51:28
__keys_tmp:nn 51:5, 51:11, 51:13
\l__keys_tmpa_tl 51:202, 51:204

keyval commands:
\keyval_parse:NNn 11:212, 11:610
\keyval_parse:nnn 11:406, 11:527
\KeyValue 11:79, 11:874, 11:1079, 11:1192, *345*
\kill 41:154, 41:162

L

\L 21:242, 21:424, 21:530, 21:771, 21:1143, 50:1332, 50:1464, 50:1554, 50:1576, 57:650
\l 21:251, 21:426, 21:551, 21:780, 21:1144, 57:650
\label 35:94, 35:160, 35:164, 44:182, 48:251, 49:25, 49:36, 49:53, 49:61, 54:623, 54:682, *1235*
label 36:236, *750*
\labelenumi *816*
\labelenumiv *816*
\labelformat *739*
\labelformat 35:150, 35:177, 35:182, 35:188, 35:193, 35:199
\labelitemi *816*
\labelitemii *816*
\labelitemiii *816*
\labelitemiv *816*
\labelsep 39:9, 39:234, 39:240, 43:53, 43:55, 43:60, 43:62, *803*
\labelwidth 39:9, 39:93, 39:233, 39:235, 39:238, *803*
\Lambda 30:307
\lambda 30:285
\land 30:375, 30:377
\langle 30:601
\language 20:52, 20:122, 37:532, 37:724, 54:601, 02:35, 56:10, 02:82, 02:84, 02:99, *1246*
\lastbox 24:707, 38:193, 38:194, 39:132, 39:138, 39:209, 44:99, 44:132, 54:307, *409*
\LastDeclaredEncoding 24:137, 24:140, 57:482, *1237*
\LastMark 48:366, 48:485, *944*
\lastnamedcs 06:758
\lastnodetype 24:700, 24:701, 24:702, 24:706
\lastpenalty 24:703, 32:112, 32:115
\lastskip 18:45, 18:129, 18:141, 18:160, 18:221, 18:222, 18:226, 18:228, 18:229, 18:241, 18:256, 18:273, 18:295, 18:298, 18:328, 18:331, 18:362, 18:365, 18:366, 32:102, 32:105, 39:115, 39:116, 39:174, 39:175, 42:123, 02:514, 02:515, 02:517, 02:519, *1196*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\LaTeX . . . 19:3, 19:15, 50:1294, 50:1427, 50:1516, 53:397, 53:403, 06:837, 02:381
\LaTeXe 19:13
\latexrelease *1057*
\LaTeXReleaseInfo 03:36, 03:37, 03:40, 03:45, 03:50, 37:47, 37:112, *38*
\latexreleaseversion 03:1
\lbrace 21:308, 30:605
\lbrack 02:441
\lccode 14:19, 14:20, 14:21, 14:22, 14:23, 14:24, 21:140, 21:1065, 37:653, 37:670, 37:684, 37:699, 37:742, 57:224, 57:241, 57:249, 57:256, 57:258, 57:259, 57:261, 57:263, 57:264, 57:265, 57:266, 57:536, 57:544, 57:551, 57:553, 57:554, 57:556, 57:558, *1243*
\lceil 30:609
\lcode *1234*
\ldotp 30:508, 30:511, 30:626
\ldots 21:322, 30:512, *1222*
\le 30:423, 30:425
\leaders 30:347, 30:565, 30:566, 30:568, 30:569, 41:376, 42:564, 42:577, 42:589, 42:599, 44:214, 44:237, 02:546
\leadsto 29:728
\leavevmode 18:454, 18:468, 21:75, 21:184, 21:289, 21:290, 21:393, 21:425, 21:429, 21:432, 21:479, 21:763, 21:796, 32:123, 33:98, 33:924, 37:536, 37:557, 37:570, 37:581, 37:611, 37:625, 37:720, 37:730, 37:743, 38:469, 38:481, 38:508, 39:58, 39:103, 40:8, 40:17, 40:24, 40:195, 40:197, 40:213, 40:245, 40:264, 40:331, 40:358, 40:437, 40:544, 40:561, 40:568, 41:178, 42:134, 42:314, 42:373, 44:40, 44:210, 44:222, 44:233, 45:530, 47:34, 54:159, 54:164, 54:186, 54:191, 02:505, 02:532, 02:535, 02:546, 02:548, *1224*
\left 30:632, 30:634, 30:636, 30:638, 30:643, 30:644, 30:645, 30:646, 38:167, 38:173, 38:195
\Leftarrow 30:417, 30:499, 30:505
\leftarrow 30:444, 30:446, 30:487, 30:497, 30:503, 30:557
\leftarrowfill 30:541, 30:557
\lefteqn 38:428
\leftharpoondown 30:460, 30:474
\leftharpoonup 30:459
\lefthyphenmin 56:11, 02:371
\leftline 40:614
\leftmargin 39:9, 39:52, 39:53, 39:94, 39:170, 39:172, *803*
\leftmargini 38:461, 39:17, *803*
\leftmarginii 39:17
\leftmarginiii 39:17
\leftmarginiv 39:17
\leftmarginv 39:17
\leftmarginvi 39:17, *803*
\leftmark 49:77, *947*
\Leftrightarrow 30:416
\leftrightarrow 30:443
\leftskip 24:684, 37:452, 37:458, 37:462, 37:472, 37:476, 37:480, 37:529, 37:551, 39:74, 40:392, 40:413, 44:207, 44:212, 44:230, 44:235, 02:416, 02:527
legacy commands:
\legacy_if:nTF 48:401, 48:422, 48:460, 48:467
\legacyoldstylenums 33:4, 33:617
\leq 30:421, 30:423
\leqno 38:439
\let *1213*
\LetLtxMacro *101*
\lfloor 30:613
\lg 38:4
\lgroup 30:615
\lhd 29:731
\lhook 30:484, 30:485
\lim 38:6
\liminf 38:8
\limits 30:546, 30:550, 38:162, 38:353
\limsup 38:7
\line . 14:269, 42:158, 42:450, 42:808, 42:825
\linebreak *421*
\linebreak 18:9, 18:27
\linepenalty 02:320
\lineskip 30:465, 38:200, 40:394, 40:414, 41:71, 41:198, 42:136, 42:315, 42:374, 53:223, 53:274, 54:626, 54:685, 02:410, 02:435, 02:500, 02:535
\lineskiplimit 30:465, 30:517, 38:202, 38:206, 40:380, 40:395, 40:402, 53:224, 53:275, 54:626, 54:685, 02:396, 02:436, 02:500, 02:537, 02:538, *1246*
\linespread 24:316
\linethickness 42:130, 42:809, 42:826
\linewidth 20:30, 20:100, 20:158, 38:298, 38:324, 38:470, 38:482, 38:509, 38:513, 38:532, 39:9, 39:51, 39:52, 39:54, 40:390, 40:411, 41:36, 45:266, 54:148, 54:207, *802*
\LinkTargetOff 17:20

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\LinkTargetOn 17:20
list (env.) 39:34
\list 39:34, 39:260, 39:271
\listfiles *978*
\listfiles 20:750, *220*
\listparindent 39:9, 39:41, 39:50, *803*
\literal *1202*
\ll 30:441
\llap 39:262, 39:273, 40:618
\lmoustache 30:570
\ln 38:5
\lnot 30:333, 30:334
\LoadClass *976*
\LoadClass 50:686, 50:716, 50:959, 50:1094, 50:1189, 50:1197, 50:1198, *1198*
\LoadClassWithOptions *976*
\LoadClassWithOptions 50:715
\LoadFontDefinitionFile 24:443, 24:469, 24:470, 30:21, 30:27, 30:28, 30:29, 30:33
\LoadPackageWithOptions *1037*
\loccount 04:17
\log 38:3
\loggingall 02:557
\loggingoutput 02:571, 02:589, 02:605, 02:619, 02:553, *1239*
\LogHook 08:2796, *205*
\LogSocket 10:170, 10:184, *335*
\long *1253*
\Longleftarrow 30:499
\longleftarrow 30:496
\Longleftrightarrow 30:505, 30:507
\longleftrightarrow 30:503
\longmapsto 30:501
\Longrightarrow 30:493
\longrightarrow 30:494, 30:501
\loop 04:150, 04:159, 24:698, 41:382, 50:1242, 50:1303, 50:1373, 50:1436, 50:1487, 50:1525, 57:365, 57:376, 57:386, 57:397, 57:427, 57:453, 57:463, 01:65, 02:490, *1220*
\looseness 02:337
\lor 30:376, 30:378
\lower 19:2, 30:465, 40:278, 42:35, 42:45, 42:196, 42:305, 42:306, 42:353, 42:354, 42:409, 42:410
\lowercase 14:26, 21:141, 21:1066, 21:1544, 24:332, 24:415, 37:657, 37:674, 37:688, 37:703, 37:743, *1232*
\lq 02:439
lrbox (env.) *818*
\lrbox 40:183
\ltfilehookdate 52:542
\ltfilehookversion 52:542
lua commands:
\lua_now:n 53:31
\luabytecode 04:202
\luachunk 04:210
\luadef 04:182, 04:186, *42*
\luafunction . . . 04:178, 04:182, 04:186, *42*
luatexbase 04:288
\luatexluafunction 01:21, 01:26
\luatexversion 04:5, 21:994, 01:14, 05:73, 05:119

M

\M 02:437
\Macro *1195*
\mag 02:376
\magstep 02:428
\magstephalf 02:428
\makeat... . . . . *1214*
\makeatletter 09:376, 20:32, 20:102, 20:160, 24:421, 37:27, 37:92, 37:154, 44:151, 50:680, 50:923, 50:1055, 54:2, 06:833, *1236*
\makeatother 09:376, 50:680, 06:833, 57:712, *1211*
\makebox *818*
\makebox 38:298, 38:324, 40:3
\makeglossary *934*
\makeglossary 20:205, 46:20, *1219*
\makeindex *934*
\makeindex 20:204, 46:3, *1219*
\makelabel 39:45, 39:97, 39:229, 39:242, 39:262, 39:273, *802*
\MakeLinkTarget . . 17:20, 43:53, 43:55, *420*
\MakeLowercase 57:571, *1226*
\MakeRobust . . . 28:888, 28:1083, 42:805, 42:806, 42:807, 42:808, 42:809, 42:810, 42:811, 42:812, 42:813, 42:814, 42:815, 42:816, 06:283, 06:878, 06:879, 06:880, 06:881, 06:882, 06:883, 06:884, 06:885, 06:886, 06:887, 06:888, 06:889, 06:890, 06:891, 06:892, 06:893, *1254*
\maketitle *898*
\MakeTitlecase 57:571, *1262*
\MakeUppercase 35:152, 35:154, 35:179, 35:190, 57:571, *1226*
\mapsto 30:451
\mapstochar 30:450, 30:451, 30:501
\marginpar 45:308, *404*
\marginparpush 54:85, 54:1838
\marginparsep 54:84, 54:1849, 54:1851
\marginparwidth 45:341, 45:359, 54:83, 54:1851
\mark . 49:31, 49:42, 49:56, 49:64, 49:82, *942*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

mark commands:
\mark_debug_off: 48:305, 48:311, 48:322, *946*
\mark_debug_on: 48:305, 48:306, 48:321, *946*
\mark_if_eq:nnnn 48:265
\mark_if_eq:nnnnnn 48:272
\mark_if_eq:nnnnnnTF 48:265, *945*
\mark_if_eq:nnnnTF 48:265, 48:373, 48:376, 48:379, *945*
\mark_insert:nn . . 48:174, 48:183, 48:216, 48:216, 48:365, 49:28, 49:29, 49:30, 49:39, 49:40, 49:41, *957*
\mark_new_class:n 48:7, 48:7, 48:16, 48:363, *943*
\mark_use_first:nn 48:260, 48:260, 48:367, *944*
\mark_use_last:nn 48:260, 48:261, 48:369, *944*
\mark_use_top:nn 48:260, 48:262, 48:371, *944*
mark internal commands:
__mark_class_status:nn 48:323, 48:324, 48:355
__mark_debug:n 48:20, 48:90, 48:164, 48:170, 48:176, 48:229, 48:305, 48:305, 48:318, 48:360, 48:397, 48:456, *962*
__mark_debug_gset: 48:305, 48:309, 48:314, 48:316
__mark_error:n 48:205, 48:208, 48:209, 48:210, 48:213
__mark_extract_and_handle_-marks:nn 48:65, 48:65, 48:118, 48:150, *953*
__mark_get_from_splitmarks: 48:151, 48:156, 48:156
__mark_get_marks_for_reinsertion:nNN 48:147, 48:147, *950*
__mark_id:n 48:42, 48:43, 48:44, 48:45, 48:46, 48:47, 48:48, 48:49, 48:50, 48:51, 48:52, 48:53, 48:54, 48:55, 48:56, 48:57, 48:58, 48:59, 48:225, 48:248, 48:248, *951*
__mark_new_class:nn 48:7, 48:14, 48:17
__mark_prepare_and_extract:nn 48:70, 48:73, 48:73, 48:94, *953*
__mark_status:n 48:352, 48:352, 48:360, 48:399, 48:458
__mark_update_dblcol_structures: 48:412, 48:412, 48:478, *949*
__mark_update_singlecol_-structures: 48:381, 48:381, 48:476, *949*
__mark_update_structure_-alias:nn 48:123, 48:194, 48:194, 48:391, 48:392, 48:393, 48:394, 48:424, 48:428, 48:429, *949*
__mark_update_structure_from_-material:nn 48:117, 48:117, 48:384, 48:388, 48:415, 48:419, *949*
__mark_update_structure_from_-splitmarks:n 48:119, 48:122, 48:122, *955*
__mark_update_structure_to_-err:n . . . 48:205, 48:205, 48:425, *949*
__mark_use:n 48:260, 48:261, 48:262, 48:263, 48:264
__mark_vbox_set_split_to_-maxdimen:NN 48:85, 48:99, 48:100, *954*
\markboth 49:21, 49:22, 49:49, 49:51, 49:70, 49:72, *942*
\markright 49:22, 49:59, 49:73, *946*
\marks 04:37, 57:10, 57:12
math (env.) 38:345
\math 38:345
\mathaccemt *1245*
\mathaccent . 28:805, 28:853, 28:887, 28:897
\mathalpha 28:975, 28:1154, 30:176, 30:177, 30:178, 30:179, 30:180, 30:181, 30:182, 30:183, 30:184, 30:185, 30:186, 30:187, 30:188, 30:189, 30:190, 30:191, 30:192, 30:193, 30:194, 30:195, 30:196, 30:197, 30:198, 30:199, 30:200, 30:201, 30:202, 30:203, 30:204, 30:205, 30:206, 30:207, 30:208, 30:209, 30:210, 30:211, 30:212, 30:213, 30:214, 30:215, 30:216, 30:217, 30:218, 30:219, 30:220, 30:221, 30:222, 30:223, 30:224, 30:225, 30:226, 30:227, 30:228, 30:229, 30:230, 30:231, 30:232, 30:233, 30:234, 30:235, 30:236, 30:237, 30:304, 30:305, 30:306, 30:307, 30:308, 30:309, 30:310, 30:311, 30:312, 30:313, 30:314, 30:523, 30:524, 30:525, 30:526, 30:527, 30:528, 30:529, 30:530, 30:532, 30:535
\mathbf 29:14, 29:255, 29:308, 29:353, 29:381, 29:475, 30:158
\mathbin 28:1159, 30:239, 30:240, 30:242, 30:365, 30:366, 30:367, 30:368, 30:371, 30:372, 30:373, 30:374, 30:377, 30:378, 30:379, 30:380, 30:381, 30:382, 30:383, 30:384, 30:385, 30:386, 30:387, 30:388,

30:389, 30:390, 30:391, 30:392, 30:393, 30:394, 30:395, 30:396, 30:397, 30:398, 30:399, 30:400, 30:401, 30:402, 30:403, 30:404, 38:37
\mathcal 30:157
\mathchar 28:916, 28:960, 30:342, 30:343, 30:624, 02:533, *1245*
\mathchardef 12:3, 12:4, 12:5, 12:6, 21:70, 04:227, 28:951, 02:21, 02:22, 02:23, 02:24, 02:107, 02:110, 02:111, *1218*
\mathcharzero 04:227
\mathchoice 38:61
\mathclose 28:1162, 30:238, 30:247, 30:249, 30:252, 30:257, 30:263, 30:265, 30:267, 30:573, 30:600, 30:604, 30:608, 30:612, 30:618, 38:43, 38:46, 38:49, 38:52
\mathcode . . 28:948, 30:259, 30:260, 30:261
\mathdefaultsmode 38:436, 38:437
\mathdollar 21:307, 30:621, *1218*
\mathellipsis 21:321, 30:626, *1218*
\mathfontset *1193*
\mathgroup 24:14, 26:303, 26:309, 26:315, 26:316, 26:327, 30:650, 33:8, 33:14, 33:614, 33:1136, 02:79, *1193*
\mathhexbox 29:652, 02:533, *1225*
\mathindent 38:459, 38:471, 38:483, 38:511, 38:522, *1204*
\mathinner . 30:511, 30:515, 30:520, 30:626
\mathit 25:696, 29:29, 30:160, 30:163, 30:624
\mathnormal 30:156
\mathop 28:1158, 30:348, 30:349, 30:350, 30:351, 30:352, 30:353, 30:354, 30:356, 30:357, 30:358, 30:359, 30:360, 30:361, 30:363, 30:364, 30:544, 30:547, 38:3, 38:4, 38:5, 38:6, 38:7, 38:8, 38:9, 38:10, 38:11, 38:12, 38:13, 38:14, 38:15, 38:16, 38:17, 38:18, 38:19, 38:20, 38:21, 38:22, 38:23, 38:24, 38:25, 38:26, 38:27, 38:28, 38:29, 38:30, 38:31, 38:32, 38:33, 38:34, 38:162, 38:353
\mathopen 28:1161, 30:248, 30:251, 30:256, 30:262, 30:264, 30:266, 30:571, 30:602, 30:606, 30:610, 30:614, 30:616, 38:41, 38:44, 38:47, 38:50
\mathord 28:975, 28:1157, 30:243, 30:250, 30:253, 30:258, 30:270, 30:271, 30:272, 30:274, 30:275, 30:276, 30:277, 30:278, 30:279, 30:280, 30:281, 30:282, 30:283, 30:284, 30:285, 30:286, 30:287, 30:288, 30:289, 30:290, 30:291, 30:292, 30:293, 30:294, 30:295, 30:296, 30:297, 30:298, 30:299, 30:300, 30:301, 30:302, 30:303, 30:315, 30:316, 30:317, 30:318, 30:319, 30:320, 30:321, 30:322, 30:323, 30:324, 30:325, 30:326, 30:327, 30:328, 30:329, 30:330, 30:331, 30:332, 30:334, 30:335, 30:336, 30:337, 30:338, 30:339, 30:340, 30:341, 30:531, 30:533, 30:534, 30:556, 30:557, 30:560, 30:561, 30:562, 30:563, 30:575, 30:577, 30:579, 30:582, 30:584, 30:598, 30:620, 30:621, 30:622, 30:623
\mathpalette 30:464, 30:468, 30:471, 38:60, 38:69, 38:99, 38:129
\mathparagraph 21:310, 22:216, 22:228, 30:621, *1234*
\mathpunct 28:1163, 30:241, 30:245, 30:508, 30:509, 30:510
\mathrel 28:1160, 30:244, 30:246, 30:254, 30:255, 30:268, 30:269, 30:345, 30:405, 30:406, 30:407, 30:408, 30:409, 30:410, 30:411, 30:412, 30:413, 30:414, 30:415, 30:416, 30:417, 30:418, 30:421, 30:422, 30:425, 30:426, 30:427, 30:428, 30:429, 30:430, 30:431, 30:432, 30:433, 30:434, 30:435, 30:436, 30:437, 30:439, 30:440, 30:441, 30:442, 30:443, 30:444, 30:445, 30:448, 30:449, 30:450, 30:452, 30:453, 30:454, 30:455, 30:456, 30:457, 30:458, 30:459, 30:460, 30:461, 30:462, 30:464, 30:468, 30:471, 30:478, 30:480, 30:483, 30:484, 30:486, 30:489, 30:491, 30:586, 30:588, 30:590, 30:592, 30:594, 30:596, 38:42, 38:45, 38:48, 38:51, 38:162, 38:353
\mathring 30:535, *1236*
\mathrm 29:5, 29:406, 29:449, 29:481, 30:155, *1194*
\mathsection 21:311, 22:215, 22:227, 30:621, *1234*
\mathsf 29:8, 29:411, 29:454, 29:484, 30:159, 30:162
\mathsterling 21:319, 30:621, *1218*
\mathstrut . . . 38:84, 38:93, 38:171, 38:172
\mathsurround 02:400, 02:521, *1235*
\mathsymbol 28:953
\mathtt 29:11, 29:416, 29:459, 29:487, 30:161
\mathunderscore 30:621, *1219*
\mathversion . 24:336, 29:614, 29:616, *1212*
\mathversion. *1193*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\matrix 38:169, 38:173, 38:180
\max 38:22
\maxdeadcycles 54:7, 02:373
\maxdepth 18:302, 18:335, 20:60, 20:129, 20:184, 54:92, 54:171, 54:172, 54:510, 54:518, 54:550, 54:719, 54:728, 54:768, 54:995, 57:152, 02:393, *1206*
\maxdimen 02:569, 02:570, 02:588, 02:604, 02:619, 24:670, 24:680, 24:716, 24:731, 26:384, 26:437, 30:465, 42:474, 42:502, 42:532, 42:610, 42:627, 50:1597, 50:1645, 50:1654, 53:351, 53:353, 53:409, 54:293, 54:1857, 54:1877, 54:1882, 54:2203, 54:2243, 54:2244, 54:2246, 57:156, 02:309, 02:394, 02:395, 02:500, 02:538, 02:554, *1195*
\mbox *818*
\mbox 19:13, 21:293, 21:409, 21:568, 21:1165, 29:648, 30:513, 40:11, 40:20, 40:24, 42:52, 45:409, 45:416, 45:437, 45:444, 02:533, *1223*
\mddefault 29:18, 29:284, 29:290, 29:291, 29:292, 29:331, 29:332, 29:333, 29:342, 29:369, 29:385, 29:402, 29:443, 29:479, 30:100, 30:112, 30:114, 30:128, *1259*
\mdseries 29:16, 29:17, 29:221, 29:277, 29:328, 29:329, 29:363, 29:364, 29:383, 29:384, 29:477, 29:478, 29:651, 32:20, *221*
mdseries 29:420
mdseries/defaults 29:420
\meaning 28:627, 28:640, 28:741, 28:806, 28:853, 28:917, 28:1011, 28:1107, 28:1211, 01:203, 01:212, 06:240, 06:302, 06:340, 06:368, 06:451, 01:307, 06:804
\medbreak 06:885, 06:906, 02:515
\mediumseries *642*
\medmuskip 30:652, 38:36, 38:38, 38:224, 38:227, 38:241
\medskip 18:447, 02:518
\medskipamount 18:448, 18:450, 02:517
\medspace 38:214
\MessageBreak 03:101, 14:3, 14:6, 14:13, 14:33, 14:46, 14:60, 14:73, 14:220, 14:222, 14:228, 14:235, 17:13, 21:161, 21:988, 21:1549, 21:1551, 21:1553, 21:1556, 24:34, 24:35, 24:561, 24:595, 25:456, 26:20, 26:21, 26:67, 26:88, 26:327, 26:478, 26:498, 26:530, 26:546, 26:561, 26:574, 27:31, 27:33, 28:280, 28:313, 28:582, 28:591, 28:729, 29:61, 29:94, 32:144, 33:23, 33:79, 33:81, 33:100, 33:848, 33:850, 33:851, 33:852, 33:854, 33:856, 33:857, 33:858, 33:859, 33:860, 33:910, 33:912, 33:919, 33:926, 33:1141, 05:78, 05:81, 05:82, 05:83, 05:84, 05:85, 05:86, 37:54, 05:99, 05:100, 05:101, 05:102, 05:103, 37:119, 37:158, 06:204, 06:291, 06:330, 06:358, 50:353, 50:367, 50:742, 50:753, 50:755, 50:757, 50:768, 50:954, 50:955, 50:956, 50:957, 50:967, 50:968, 50:970, 50:971, 50:972, 50:974, 50:976, 50:1062, 50:1063, 50:1065, 50:1066, 50:1067, 50:1069, 50:1071, 50:1089, 50:1090, 50:1091, 50:1092, 50:1174, 50:1191, 50:1192, 50:1264, 50:1281, 50:1320, 50:1395, 50:1414, 50:1453, 50:1508, 50:1542, 50:1658, 50:1660, 50:1743, 50:1746, 50:1759, 50:1761, 52:488, 53:99, 53:175, 53:371, 53:477, 53:478, 53:479, 53:480, 54:582, 54:1998, 54:2035, 57:300, 57:301, 57:302, 57:304, *1220*
\mho 29:724
\mid 30:409
\min 38:23
minipage (env.) *819*
\minipage 40:424
\mit 29:776
\mkern 30:342, 30:345, 30:347, 30:469, 30:478, 30:520, 30:521, 30:522, 30:552, 30:553, 30:554, 30:555, 30:556, 30:557, 30:558, 30:559, 38:36, 38:37, 38:40, 38:73, 38:74, 44:215, 44:238
mlist commands:
  mlist_to_hlist 04:1008
mode commands:
  \mode_if_horizontal:TF . 16:95, 16:100
  \mode_if_inner:TF 16:96
  \mode_if_math:TF 28:344
  \mode_if_vertical:TF . 16:114, 16:126
\models 30:491, *1239*
module commands:
  module_error 04:344
  module_info 04:344
  module_warning 04:344
\module_error *45*
\module_info *45*
\module_warning *45*
modules 04:297

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\month 03:17, 01:169, 50:1298, 50:1431, 50:1520, 02:385
\moveright 54:629, 54:688
\mp 30:396
\mscount 41:379
msg commands:
\msg_... . . . . *1259*
\msg_error:nn 10:49, 11:267, 16:122, 16:132, 36:14, 08:436, 08:453
\msg_error:nnn 08:2147, 10:30, 10:92, 10:113, 10:130, 11:62, 11:69, 11:185, 11:233, 11:247, 11:291, 11:313, 11:413, 11:426, 11:443, 11:499, 11:537, 11:616, 11:625, 11:731, 11:883, 48:11, 48:214, 48:244, 51:244, 52:75, 07:345, 07:746, 07:2862, 07:2868, 07:2879, 07:2898, 07:2901, 07:2909, 08:76, 08:94, 08:179, 08:208, 08:211, 08:461, 08:525, 08:542, 08:596, 08:616
\msg_error:nnnn 09:43, 09:101, 09:397, 10:78, 10:109, 10:127, 11:56, 11:75, 11:199, 11:555, 11:583, 11:595, 11:684, 11:696, 11:796, 11:900, 11:906, 16:20, 16:33, 16:53, 16:65, 16:108, 36:35, 51:38, 51:74, 51:218, 07:459, 07:467, 07:491, 07:495, 07:531, 07:562, 07:577, 07:659, 07:666, 07:675, 07:703, 07:715, 07:734, 07:768, 07:1115, 07:1820, 07:2429, 07:2433, 07:2810, 07:2824, 07:2929, 07:2943, 08:112, 08:507, 08:640
\msg_error:nnnnn 11:48, 11:410, 07:690, 07:1964, 07:1971, 07:2268, 08:440, 08:471, 08:481
\msg_error:nnnnnn 08:1321, 08:1333, 08:1358, 08:1760
\msg_expandable_error:nn 08:341, 08:342
\msg_expandable_error:nnn 08:2542, 36:77, 36:113, 05:185, 07:416, 07:2975, 08:370, 08:1261
\msg_expandable_error:nnnn 52:464, 07:2160, 07:2187
\msg_info:nnnn 11:109, 11:193, 11:396, 11:641, 11:664, 11:680, 07:69, 07:80, 07:230, 07:231, 07:247, 07:251
\msg_line_context: 08:2624, 08:2629, 08:2633, 08:2680, 11:1006, 11:1012, 11:1019, 11:1026, 11:1033, 11:1040, 11:1048, 11:1135, 11:1137, 11:1141, 11:1144, 48:20, 48:92, 48:165, 48:230, 07:2767, 07:2772, 07:2777, 07:2782, 08:563
\msg_module_name:n 51:93, 51:223, 51:229, 51:233
\g_msg_module_name_prop 08:2566
\g_msg_module_type_prop . 08:2564, 08:2565, 10:169, 11:1155, 48:279, 07:54
\msg_new:nnn 08:2622, 08:2627, 08:2631, 08:2676, 08:2682, 08:2687, 08:2692, 08:2697, 08:2701, 08:2708, 08:2873, 11:1134, 11:1136, 11:1138, 11:1143, 11:1145, 11:1150, 05:187, 51:221, 52:470, 07:1116, 07:2721, 07:2764, 07:2769, 07:2774, 07:2779, 07:2784, 07:2796
\msg_new:nnnn 08:2567, 08:2577, 08:2584, 08:2591, 08:2600, 08:2614, 08:2636, 08:2652, 08:2665, 09:575, 09:585, 10:146, 10:151, 10:155, 10:159, 10:165, 11:962, 11:970, 11:976, 11:985, 11:991, 11:998, 11:1005, 11:1011, 11:1018, 11:1025, 11:1032, 11:1039, 11:1047, 11:1054, 11:1063, 11:1069, 11:1075, 11:1082, 11:1088, 11:1094, 11:1100, 11:1106, 11:1122, 11:1128, 16:144, 16:155, 36:245, 36:251, 36:257, 48:280, 48:288, 48:296, 51:89, 51:226, 51:249, 07:2557, 07:2564, 07:2571, 07:2578, 07:2585, 07:2592, 07:2599, 07:2606, 07:2613, 07:2620, 07:2628, 07:2635, 07:2642, 07:2649, 07:2657, 07:2665, 07:2674, 07:2681, 07:2688, 07:2695, 07:2702, 07:2708, 07:2714, 07:2723, 07:2729, 07:2736, 07:2743, 07:2754
\msg_redirect_module:nnn 07:53
\msg_show:nnnnn 11:953
\msg_show:nnnnnn 11:942
\msg_show_item_unbraced:nn 11:946, 11:958
\msg_warning:nnn . 07:77, 08:295, 08:313
\msg_warning:nnnn 08:2872, 51:217, 07:727, 08:978, 08:991, 08:1013, 08:1027
\msg_warning:nnnnn 08:768
\msg_warning:nnnnnn 08:1768
msg internal commands:
\c__msg_coding_error_text_tl 48:283, 48:291, 48:299
\c__msg_return_text_tl 48:286, 48:302
\mskip 38:36, 38:38, 38:219, 38:238, 38:241, 38:243, 38:244
\mu 30:286
\mubyte 57:352

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\multicolumn 41:233, *1167*
\multiput 42:78, 42:810, 42:827
\multispan 41:233, 41:379, *1228*
\muskip 04:34, 30:552, 30:553, 02:29, 02:55, 02:93
muskip commands:
  \muskip_eval:n 11:366
  \muskip_new:N 11:35
  \l_tmpa_muskip *346*
\muskipdef 04:228, 02:55, 02:93
\muskipzero 04:228

N

\n 04:331, 04:333, 04:340, 04:342, 04:469, 04:688, 04:713, 04:737, 04:778, 04:800, 04:819, 04:827, 04:828, 04:853, 04:870, 04:909, 04:916, 04:917, 04:924, 04:936, 57:121, 57:126
\nabla 30:326
\NAME *81*
\narrower 02:526
\natural 30:336
\ncallback 04:783
\ndefault 04:788, 04:792
\ne 30:420, *675*
\nearrow 30:412
\NeedsFormat *1198*
\NeedsTeXFormat 26:12, 33:817, 50:747, 50:1805, *1202*
\neg 30:332, 30:333
\negmedspace 38:214, *1254*
\negthickspace 38:214, *1254*
\negthinspace 18:544, 38:214, *1251*
\neq 30:419, *675*
new commands:
  new_attribute 04:410
  new_bytecode 04:444
  new_chunkname 04:457
  new_luafunction 04:473
  new_whatsit 04:432
\new_attribute *43*
\new_bytecode *43*
\new_chunkname *43*
\new_luafunction *43*
\new_whatsit *43*
\newattribute *42*
\newattribute 04:74, 04:238
\newbox . . . 12:13, 37:583, 38:66, 39:27, 40:154, 41:16, 41:17, 41:18, 41:343, 42:6, 42:669, 42:674, 54:86, 54:122, 54:123, 54:124, 02:47, 02:314, 02:523
\newcatcodetable *42*
\newcatcodetable 04:84, 04:93, 04:94, 04:120, 04:121, 04:242
\newcommand *81*
\newcommand 21:4, 25:530, 25:535, 25:540, 29:36, 29:72, 30:59, 30:60, 30:61, 30:62, 30:64, 30:65, 30:67, 30:68, 30:100, 30:101, 30:102, 30:103, 30:104, 30:105, 30:127, 30:128, 30:129, 36:240, 37:396, 37:397, 37:398, 37:399, 06:77, 42:681, 52:573, 52:574, 52:575, 52:576, 52:578, 54:2329, 54:2332, 54:2335, 54:2336, 54:2339, 54:2340, 57:561, *100*
\NewCommandCopy 06:473, 06:475, 07:1092, 07:1144, 07:1162, 07:1175, 07:1190, 07:1308, *98*
\newcount 12:7, 12:8, 18:125, 20:7, 22:41, 22:61, 26:25, 28:27, 28:142, 28:429, 38:55, 38:357, 38:358, 39:23, 39:24, 39:25, 39:26, 39:56, 39:250, 39:265, 40:473, 41:11, 41:12, 41:13, 41:14, 41:15, 41:335, 41:336, 41:337, 42:663, 42:664, 42:665, 42:666, 42:675, 44:36, 44:140, 44:141, 45:3, 45:267, 45:268, 45:269, 45:270, 50:1594, 54:105, 54:107, 54:109, 54:111, 54:113, 54:121, 54:2024, 54:2327, 54:2330, 54:2333, 54:2337, 57:3, 57:4, 57:5, 57:91, 02:47, 02:356, 02:427, *1241*
\newcounter *512*
\newcounter 22:10
\newdimen 12:10, 12:11, 12:12, 18:124, 26:398, 26:399, 38:53, 39:9, 39:10, 39:11, 39:12, 39:13, 39:14, 39:15, 39:16, 39:17, 39:18, 39:19, 39:20, 39:21, 39:22, 40:210, 40:211, 41:3, 41:5, 41:6, 41:7, 41:8, 41:166, 41:338, 41:339, 41:340, 41:341, 42:3, 42:4, 42:5, 42:7, 42:431, 42:432, 42:433, 42:434, 42:435, 42:436, 42:667, 42:668, 42:670, 42:671, 42:672, 42:673, 45:451, 54:71, 54:72, 54:73, 54:75, 54:76, 54:77, 54:78, 54:79, 54:80, 54:81, 54:82, 54:83, 54:84, 54:85, 54:91, 54:93, 54:94, 54:95, 54:96, 54:108, 54:110, 54:112, 54:114, 54:115, 54:116, 54:117, 54:118, 54:119, 54:120, 54:2025, 54:2026, 02:47, 02:309, 02:311, 02:312, 02:426
\NewDocumentCommand . 08:2713, 08:2715, 08:2717, 08:2721, 08:2723, 08:2725, 08:2736, 08:2738, 08:2748, 08:2750, 08:2759, 08:2761, 08:2768, 08:2770, 08:2772, 08:2774, 08:2776, 08:2800,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

08:2802, 08:2805, 17:24, 17:46, 17:55, 17:56, 17:57, 20:750, 22:127, 22:135, 35:66, 35:68, 35:155, 36:80, 51:184, 51:186, 51:197, 51:255, 57:627, 57:634, 57:641, 07:2804, *193*
\NewDocumentEnvironment 07:2588, 07:2595, 07:2840, *217*
\newenvironment *82*
\newenvironment 06:146, 50:1296, 50:1429, 50:1518, *217*
\NewEnvironmentCopy . 06:514, 06:516, *1264*
\NewExpandableDocumentCommand 36:118, 48:366, 48:368, 48:370, 48:372, 48:375, 48:378, 57:597, 07:2923
\newfam 04:38, 24:16, 02:47, *1225*
\newfont 29:618
\newgroup 28:47
\newhelp 02:306
\NewHook 08:2713, 08:2888, 20:73, 20:74, 20:75, 20:348, 20:351, 20:400, 26:150, 29:420, 29:421, 29:422, 29:423, 29:424, 29:425, 29:426, 29:427, 29:428, 37:34, 37:35, 37:36, 37:37, 37:38, 37:99, 37:100, 37:101, 37:102, 37:103, 50:1040, 50:1041, 52:177, *227*
\NewHookPair *305*
\NewHookWithArguments 08:2719, 35:101, *195*
\newif 03:73, 12:9, 20:5, 20:6, 24:204, 25:401, 25:413, 28:15, 29:528, 32:82, 33:868, 33:1221, 33:1222, 35:3, 38:75, 38:76, 38:203, 38:359, 39:28, 39:29, 39:30, 39:31, 39:32, 39:33, 39:140, 40:423, 40:529, 41:19, 41:251, 42:157, 42:427, 42:428, 42:429, 42:430, 42:459, 42:460, 06:168, 44:38, 44:124, 50:2, 54:97, 54:98, 54:99, 54:100, 54:101, 54:102, 54:103, 54:104, 06:872, *1193*
\newinsert 40:474, 45:390, 54:27, 54:1856, 02:193, 02:242, *1244*
\newlabel 35:84, 35:106, 35:166, *1227*
\newlanguage 57:284, 02:47
\newlength *521*
\newlength 23:3
\newline 18:93, 18:100, 18:106
\newlinechar 06:20, 01:56, 02:381
\newluabytecode *43*
\newluabytecode 04:197, 04:252
\newluachunkname *43*
\newluachunkname 04:205, 04:254
\newluacmd *42*
\newluacmd 04:181, *43*
\newluafunction *42*
\newluafunction 04:4, 04:173, 04:236, 04:248, *42*
\NewMarkClass 48:363, 48:482, 57:25, 57:26, 57:27, *943*
\newmarks 48:22, 57:6, *948*
\newmathalphabet 27:13, 27:109
\NewMirroredHookPair 08:2713, 08:2890, *195*
\NewMirroredHookPairWithArguments 08:2719, *195*
\NewModuleRelease 09:4, 10:4, 11:7, 03:152, 16:4, 17:3, 33:2, 36:4, 48:4, 55:242, 07:9, 08:4, *1257*
\newmuskip 02:47
\newpage 54:135, 54:141, 54:152, *328*
\NewProperty 36:38, 36:267, *749*
\newprotectedluacmd *43*
\newprotectedluacmd 04:181, 21:1010, 53:28
\newread 02:47, 02:307, *1196*
\NewReversedHook 08:2713, 08:2889, 20:349, 20:350, 20:401, 20:402, 50:1042, 50:1043, 52:178, *195*
\NewReversedHookWithArguments 08:2719, *195*
\newrobustcmd *96*
\newsavebox *818*
\newsavebox 40:154
\newskip 12:14, 12:15, 12:17, 18:450, 18:451, 18:452, 18:517, 18:530, 23:3, 37:493, 38:360, 38:460, 39:2, 39:3, 39:4, 39:5, 39:6, 39:7, 39:8, 54:2341, 54:2342, 54:2343, 54:2347, 54:2348, 54:2351, 54:2352, 54:2353, 54:2357, 54:2358, 54:2359, 02:47, 02:310, 02:313, 02:424, 02:425
\NewSocket 10:170, 10:182, 35:117, 35:118, 55:39, 55:49, 55:63, 55:64, 55:65, 55:66, 55:67, 55:68, 55:69, 55:70, 55:71, 55:72, 55:73, 55:74, 55:75, 55:76, 55:77, 55:78, 55:79, 55:80, 55:81, 55:82, 55:83, 55:84, 55:85, 55:86, 55:87, 55:88, 55:89, 55:90, 55:91, 55:92, 55:93, 55:94, 55:95, 55:96, 55:97, *334*
\NewSocketPlug 10:161, 10:170, 10:186, 35:119, 55:40, 55:51, *330*
\NewTemplateType 11:1066, 11:1125, 11:1156, *343*
\newtheorem 43:1, *1228*
\newtie . 21:851, 33:188, 33:189, 33:208, 33:705, 33:1012, 33:1013, 33:1320, *701*
\newtoks 03:36, 12:16, 16:77, 24:351, 24:352, 25:499, 26:247, 02:47, 02:306, *1244*
\newwhatsit *43*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\newwhatsit 04:189, 04:250
\newwrite 20:3, 20:4, 44:154, 46:4, 46:21, 02:47, 02:308, *1218*
\newXeTeXintercharclass 57:35
\next *1221*
\NextLinkTarget 17:20
nfss internal commands:
__nfss_init_mv_freeze:N 28:318, 28:343, 28:344
\NG 21:531, 21:1145, 57:650, *1213*
\ng 21:552, 21:1146, 57:650, *1213*
\ni 30:437, 30:438
\noalign 30:346, 30:538, 30:541, 30:544, 30:545, 30:549, 30:550, 38:171, 38:172, 38:188, 38:191, 38:205, 38:404, 38:414, 41:224, 41:230, 41:359, 41:378, 42:148, 42:154, *1193*
\nobreak . . 18:60, 18:72, 18:116, 18:142, 18:148, 18:161, 18:174, 18:200, 18:398, 18:406, 18:432, 18:440, 18:461, 18:468, 18:528, 20:203, 20:215, 21:409, 21:435, 21:437, 21:568, 21:1165, 37:418, 37:425, 40:599, 40:609, 44:90, 44:212, 44:213, 44:217, 44:235, 44:236, 44:240, 45:531, 48:241, 49:33, 49:44, 49:58, 49:66, 54:338, 54:1153, 54:1319, 06:886, 06:907, 57:208, 57:210, 57:214, 57:215, 57:216, 57:220, 02:503, 02:506, 02:508, *1224*
\nobreakdashes 18:453
\nobreakspace 18:467, *438*
\nobreakspace␣ *1233*
\NoCaseChange 57:571, *1262*
\nocide *940*
\nocite *937*
\nocite 47:59, *1220*
\nocorr . . . 32:43, 32:58, 32:62, 32:65, *1208*
\nocorrlist 32:89, 32:121
\noexpand *1195*
\nofiles *442*
\nofiles 20:199, *1222*
\noindent 24:694, 24:720, 44:139, *406*
\nointerlineskip 30:346, 30:538, 30:541, 30:545, 30:549, 38:297, 38:323, 42:562, 42:565, 42:575, 42:577, 54:1846, 54:1854, 02:498
\nolimits 30:355, 30:362, 38:3, 38:4, 38:5, 38:9, 38:10, 38:11, 38:12, 38:13, 38:14, 38:15, 38:16, 38:17, 38:18, 38:19, 38:20, 38:21, 38:26, 38:27, 38:28, 38:29, 38:31, 38:34
\nolinebreak *421*
\nolinebreak 18:9, 18:28
\nonfrenchspacing 02:709, 20:48, 20:118, 20:176, 06:887, 06:908, 02:431
\nonscript 38:36, 38:38
\nonstopmode *955*
\nonumber 38:387, 38:426, 38:427
\nopagebreak *421*
\nopagebreak 18:7, 18:26
\noprotrusion 44:222, 44:245
\normalbaselines 38:167, 38:169, 02:396, 02:409, 02:410, 02:435
\normalbaselineskip 26:188, 40:396, 40:415, 02:424, 02:436
\normalcolor 38:352, 38:456, 40:110, 40:465, 44:218, 44:241, 45:97, 45:166, 54:218, 54:496, 54:633, 54:643, 54:692, 54:702, 54:2264, 54:2297, *1217*
\normalfont 24:671, 24:732, 29:653, 29:673, 29:675, 29:684, 29:689, 29:691, 29:699, 29:706, 29:708, 29:714, 32:18, 37:571, 38:352, 38:456, 44:218, 44:241, 45:401, *1226*
normalfont 29:420
\normalize *1206*
\normallineskip 40:394, 40:414, 02:424, 02:435
\normallineskiplimit 38:206, 40:379, 40:395, 40:401, 02:424, 02:436
\normalmarginpar 45:387
\normalsfcodes 20:44, 20:46, 20:48, 20:114, 20:116, 20:118, 20:172, 20:174, 20:176, 20:198, 54:622, 54:681, *1235*
\normalshape 25:682, 25:723, *570*
\normalsize 20:42, 20:112, 20:170, 32:142, 45:23, 45:176, 45:372, 50:5, 54:621, 54:680, *198*
\not 30:345, 30:419, 30:420, 30:442
\notexpanded *318*
\notin 30:468
\noxpand *1198*
\nu 30:287
\null 21:327, 21:363, 21:486, 21:489, 21:834, 21:837, 21:1253, 35:21, 35:38, 35:55, 37:536, 37:557, 37:720, 37:730, 38:112, 38:121, 38:169, 38:198, 44:212, 44:235, 53:385, 02:449, *1223*
\nulldelimiterspace 30:649, 02:398
\nullfont 37:242
\number 03:52, 03:61, 04:105, 22:190, 24:621, 24:624, 26:439, 28:64, 28:93, 28:113, 28:128, 28:164, 28:195, 28:225, 28:257,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

29:645, 06:2, 06:114, 50:1205, 50:1298, 50:1431, 50:1520, 53:351, 01:70
\numberline 44:72, 44:82, 44:248, 45:17, *1197*
\numexpr 04:82, 04:105, 04:157, 21:1034, 28:150, 54:36, 06:747, 02:189, 02:205, 02:215, 02:246
\nunknown 04:805
\nwarrow 30:414
\nxt *1221*

O

\O 21:244, 21:400, 21:533, 21:770, 21:1130, 57:649
\o 21:253, 21:405, 21:554, 21:781, 21:1136, 57:649
\oalign 02:535
\obeycr 18:556
\obeyedline 07:1979, 07:1988, 07:1995, 02:459, 02:463, 02:486, *28*
\obeyedspace . . . 02:464, 02:467, 02:487, *29*
\obeylines 37:542, 37:563, 37:711, 37:712, 54:587, 06:888, 06:909, 02:452, *28*
\obeyspaces 54:587, 06:889, 06:910, 02:452, *29*
\oddsidemargin 54:72, 54:74, 54:615, 54:674
\odot 30:391
\OE 21:243, 21:399, 21:532, 21:769, 21:1147, 57:607, 57:612, 57:617, 57:649, *1189*
\oe 21:252, 21:404, 21:553, 21:782, 21:1148, 57:607, 57:612, 57:617, 57:649, *1264*
\of 38:67, 38:356
\offinterlineskip 02:498
\oint 30:362
\ointop 30:361, 30:362
\oldstylenums 33:4, 33:376, 33:377, 33:378, 33:379, 33:380, 33:381, 33:382, 33:383, 33:384, 33:385, 33:610, 33:1133, *476*
\Omega 30:314
\omega 30:297
\ominus 30:394
\omit 38:191, 38:192, 41:369, 41:372, 41:379, 41:383
\OmitIndent 16:91, 16:170, *412*
\onecolumn 54:143
\OnlyDescription 26:5, 31:3
\ooalign 21:327, 21:357, 21:394, 21:480, 21:486, 21:488, 21:499, 21:515, 21:731, 21:764, 21:834, 21:837, 21:894, 21:1253, 29:650, 30:469, 30:472, 02:535, *1236*
\openin *1250*
\openout *1196*
\openup 38:199, 38:204
\oplus 30:395
\OptionNotUsed . . . 50:509, 50:536, 50:1108
or commands:
\or: 07:1459, 07:1460, 08:1251, 08:1252, 08:1253, 08:1254, 08:1255, 08:1256, 08:1257, 08:1258, 08:1259, 08:1400, 08:1416
\oslash 30:392
\OT 21:372
\otimes 30:393
\outer 04:21, 04:38, *1223*
\output 54:258, 02:333
\outputpenalty . 54:260, 54:274, 54:297, 54:300, 54:301, 54:336, 54:1163, 54:1164, 54:1329, 54:1332, 02:333
\oval 42:450, 42:453, 42:811, 42:828
\over 30:476, 38:162, 38:354
\overbrace 30:543
\overfullrule 49:98, 02:392
\overleftarrow 30:540
\overrightarrow 30:537
\owns 30:438, 30:439

P

\P 21:310, *1218*
package/.../after *1037*
package/.../before *1037*
package/after *1037*
package/before *1037*
\PackageError . . 03:77, 03:135, 03:147, 14:84, 21:1547, 33:823, 33:874, 33:918
\PackageInfo . . . 14:84, 33:827, 33:843, 33:848, 33:864, 33:865, 33:925, 33:1210
\PackageNote 14:136, *1258*
\PackageNoteNoLine 14:136
\PackageWarning 14:84, 33:825, 33:875, 33:1139, 53:476
\PackageWarningNoLine 14:84, 21:986, 54:1997
page 36:234, *750*
\pagebreak *421*
\pagebreak 18:6, 18:7, 18:23, 18:25
\pagegoal 54:1884, 54:1891
pagenum 36:235, *750*
\pagenumbering *737*
\pagenumbering 34:5
\pageref 35:10, *746*
\pageshrink 54:542, 54:546, 54:562
\pagestyle 49:2
pagetarget *751*
\pagetotal 54:130
\paperheight 54:93

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\paperwidth 54:93
\par . . . 01:104, 15:3, 15:4, 15:5, 16:140, 16:176, 18:239, 18:290, 04:156, 24:697, 37:184, 37:240, 37:418, 37:425, 37:534, 37:555, 39:63, 06:9, 39:110, 39:127, 39:130, 39:137, 06:21, 39:185, 39:188, 40:385, 40:406, 40:461, 40:491, 40:508, 41:195, 41:385, 44:41, 44:90, 44:220, 44:242, 45:15, 45:24, 45:249, 45:344, 45:478, 45:496, 49:91, 49:92, 53:401, 02:11, 54:168, 54:195, 54:259, 54:1890, 02:445, 02:461, 02:463, 02:480, 02:502, 02:511, 02:512, 02:513, 02:515, 02:517, 02:519, *402*
para commands:
\para_end: 16:93, 16:93, 16:140, 16:141, 16:142, *413*
\g_para_indent_box . . 16:16, 16:49, 16:84, 16:86, 16:89, 16:91, 16:117, *409*
\para_omit_indent: . . 16:88, 16:92, *406*
\para_raw_end: 16:113, 16:135, 16:138, *406*
\para_raw_indent: 16:113, 16:113, 16:136, *406*
\para_raw_noindent: 16:113, 16:125, 16:137, *406*
para internal commands:
__para_handle_indent: 16:34, 16:66, 16:85, 16:85, 16:119
\g__para_standard_everypar_tl . . . 16:12, 16:76, 16:78, 16:118, 16:129, *411*
__para_tmp:w 16:37, 16:41, 16:68, 16:72
para/after 16:6, *404*
para/before 16:6, *404*
para/begin 16:6, *404*
para/end 16:6, *404*
\paracntvalue *407*
\paragraphmark 44:143
\parallel 30:408
\parbox *818*
\parbox 40:298, *402*
\parfillskip . . . 24:670, 24:686, 24:731, 37:454, 37:464, 37:473, 37:481, 37:530, 37:552, 39:76, 40:393, 40:414, 44:207, 44:230, 02:423, *415*
\parindent 37:454, 37:459, 37:464, 37:473, 37:477, 37:481, 37:530, 37:552, 39:50, 40:388, 40:409, 44:208, 44:231, 02:404, 02:527, 02:528, *401*
\parsep 39:1, 39:49, 39:90, *803*
\parseunicodedataI 04:123, 04:162
\parseunicodedataII 04:124, 04:126
\parseunicodedataIII 04:128, 04:134
\parseunicodedataIV 04:130, 04:142
\parseunicodedataV 04:146, 04:149
\parshape 39:54, *409*
\parskip 37:429, 37:528, 37:530, 37:550, 37:552, 38:528, 39:49, 39:73, 39:88, 39:90, 39:117, 39:177, 39:196, 39:247, 40:388, 40:409, 41:79, 54:1163, 54:1331, 02:411, *410*
\partial 30:322
\partopsep 38:526, 39:1, 39:61, *803*
\PassOptionsToClass *976*
\PassOptionsToClass 50:441
\PassOptionsToPackage *976*
\PassOptionsToPackage 50:441
\patterns 21:205, *1217*
\pausing 02:338
\pdffilesize 05:71, 05:117
\pdfgentounicode 57:315, 57:316, 57:320, 57:335, 57:340, 57:341, 57:342
\pdfhorigin 53:314
\pdfsavepos *751*
\pdftexrevision 57:321
\pdftexversion 57:319, 57:320, 57:321
\pdfvariable 53:313, 53:318
\pdfvorigin 53:319
peek commands:
\peek_meaning:NTF 07:2518, *182*
\peek_meaning_remove:NTF 07:1861, 07:1962, 07:2506, *182*
\peek_N_type:TF 07:1867, 07:1891, 07:1923
\peek_remove_spaces:n 07:1859
\penalty 18:35, 18:38, 18:47, 18:296, 18:306, 18:329, 18:339, 18:363, 18:367, 32:118, 37:536, 37:539, 37:557, 37:560, 38:37, 38:207, 38:404, 38:414, 39:214, 41:56, 45:195, 45:199, 45:201, 45:217, 45:221, 45:223, 47:37, 54:138, 54:178, 54:197, 54:200, 54:1161, 54:1327, 02:507, 02:508, 02:509, 02:510, 02:511, 02:512, 02:516, 02:518, 02:520, *1204*
\perp 30:454
\phantom 38:75
\Phi 30:312
\phi 30:294
\Pi 30:309
\pi 30:289
picture (env.) 42:21
\picture 42:21
Plugs:
default 55:40
kernel (refstepcounter/target) 35:119

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

kernel (tagsupport/recordtarget) 55:50
\pm 30:397
\pmatrix 38:173, 38:181
\pmod 38:39
\PopDefaultHookLabel 08:2772, *201*
\poptabs 14:256, 41:142, 41:161
\poptracing 26:148, 26:340
\postdisplaypenalty 18:13, 38:468, 38:480, 38:506, 02:330
\pounds 21:318, *1214*
\Pr 38:32
pre commands:
pre_shipout_filter *1063*
\prec 30:428
\preceq 30:431
\predisplaypenalty 18:12, 38:467, 38:479, 38:505, 02:329, *1264*
\predisplaysize 02:401
\pretolerance 24:672, 24:688, 24:733, 02:316
\prevdepth 18:302, 18:303, 18:335, 18:336, 18:396, 18:401, 18:430, 18:435, 38:205, 45:196, 45:198, 45:218, 45:220, 54:169, 54:171, 54:174, 02:498, 02:502, 02:503, *835*
\PreviousTotalPages 53:407, *1064*
prg commands:
\prg_break:n 07:2965, 07:2967, 07:2969, 07:2971, 07:2972
\prg_break_point: 07:2973
\prg_do_nothing: 08:2298, 09:193, 09:268, 09:387, 09:606, 48:236, 52:132, 52:134, 53:166, 53:167, 53:172, 53:181, 53:185, 53:321, 55:22, 55:32, 07:407, 07:959, 07:1803, *141*
\prg_generate_conditional_variant:Nnn 11:83, 36:165, 36:180, 36:195
\prg_new_conditional:Npnn 08:2413, 08:2428, 08:2445, 08:2462, 08:2468, 08:2474, 08:2480, 08:2489, 08:2497, 08:2511, 08:2529, 09:311, 11:77, 11:84, 11:90, 36:154, 36:169, 36:184, 48:265, 48:272, 55:10, 08:273, 08:881, 08:1194, 08:1404
\prg_new_protected_conditional:Npnn 09:107, 09:450, 08:732, 08:790, 08:820, 08:852
\prg_replicate:nn 08:2281, 08:2289, 08:2359, 08:2375, 52:552, 07:877, 07:920, 07:970, 07:1025, 07:2275, 07:2282, 08:1168
\prg_return_false: 08:2423, 08:2439, 08:2457, 08:2466, 08:2472, 08:2478, 08:2486, 08:2495, 08:2504, 08:2507, 08:2521, 08:2536, 09:122, 09:324, 09:465, 11:81, 11:88, 11:94, 36:162, 36:177, 36:192, 48:270, 48:277, 55:11, 08:279, 08:759, 08:813, 08:825, 08:845, 08:857, 08:872, 08:889, 08:893, 08:896, 08:1199, 08:1409, *246*
\prg_return_true: 08:2421, 08:2437, 08:2456, 08:2459, 08:2465, 08:2471, 08:2477, 08:2484, 08:2494, 08:2505, 08:2519, 08:2534, 09:121, 09:324, 09:464, 11:80, 11:87, 11:93, 36:159, 36:174, 36:189, 48:269, 48:276, 08:278, 08:750, 08:804, 08:843, 08:870, 08:892, 08:1197, 08:1407
\prime 30:260, 30:324, 38:256
\ProcessedArgument 07:388, 07:392, 07:399, 07:2208, 07:2209, 07:2226, 07:2228, 07:2250, 07:2259, 07:2265, 07:2273, 07:2290, 07:2302, 07:2329, 07:2395, 07:2397, 07:3000, *178*
\ProcessKeyOptions . . . 51:63, 51:197, *1262*
\ProcessKeyPackageOptions *1261*
\ProcessList 07:3005
\ProcessOption* *1198*
\ProcessOptions 21:1572, 26:71, 33:841, 33:878, 50:537, 50:628, 50:1193, *1204*
\ProcessOptions* 50:537
\prod 30:356
prop commands:
\prop_clear:N 11:144, 11:154, 11:170, 11:180, 11:209, 11:210, 11:405, 07:1711, 08:1141
\prop_clear_new:N 11:123
\prop_const_from_keyval:Nn 08:935, 08:943, 08:945, 08:947
\prop_gclear:N 09:58, 11:110, 08:968, 08:1003, 08:1425, 08:1452
\prop_gclear_new:N 08:2400, 08:2410, 11:99, 11:131
\prop_get:NnN 11:44, 11:550, 11:553, 11:580, 11:691, 11:712, 11:793, 08:565, 08:1595, 08:1681
\prop_get:NnNTF . . 11:418, 11:531, 11:622, 11:717, 11:719, 11:804, 11:877, 51:202, 07:1717, 08:574, 08:619
\prop_gpop:NnNTF 08:976, 08:1011
\prop_gput:Nnn 08:2564, 08:2565, 08:2566, 09:51, 10:169, 11:195, 11:1155,

48:279, 55:55, 07:54, 08:575, 08:576, 08:622, 08:625, 08:1502, 08:1540
\prop_gset_eq:NN 11:100, 11:113, 11:132, 08:1143
\prop_if_empty:NTF 08:2433, 08:1482, 08:1517, 08:1972
\prop_if_empty_p:N . . 08:2450, 08:1868
\prop_if_exist:NTF 08:2076, 08:2432, 08:2470, 11:107, 11:138, 11:148, 11:164, 11:174
\prop_if_exist_p:N 08:1867
\prop_if_in:NnTF . . . 11:67, 11:184, 11:544, 11:548, 08:827, 08:835, 08:839, 08:859, 08:862, 08:866, 08:891
\prop_map_break: 08:2084, 08:1572, 08:1654
\prop_map_function:NN 09:57, 11:946, 11:957, 08:1142, *256*
\prop_map_inline:Nn 08:2079, 08:2081, 11:409, 51:237, 08:1437, 08:1458, 08:1560, 08:1567, 08:1569, 08:1642, 08:1649, 08:1651, 08:1810, 08:1871, 08:1975
\prop_new:N . . . 09:17, 11:18, 11:28, 11:30, 11:31, 11:112, 07:49, 08:30, 08:31, 08:152, 08:164, 08:1292, 08:1293
\prop_put:Nnn 11:261, 11:353, 11:355, 11:360, 11:467, 11:481, 11:496, 11:525, 11:591, 11:660, 07:1708, 08:37, 08:42, 08:1177, 08:1780, 08:1791
\prop_remove:Nn 11:424
\prop_set_eq:NN . . 11:124, 11:141, 11:151, 11:167, 11:177, 08:1498, 08:1535
\prop_show:N *281*
property commands:
\property_gset:nnnn 36:6, 36:17, 36:46, *747*
\property_if_exist:n . 36:154, 36:165
\property_if_exist:nTF 36:154, 36:166, 36:167, 36:168, *749*
\property_if_exist_p:n . . . 36:154, *749*
\property_if_recorded:n 36:169, 36:180
\property_if_recorded:nn 36:184, 36:195
\property_if_recorded:nnTF 36:184, 36:196, 36:197, 36:198, 36:218, *749*
\property_if_recorded:nTF 36:169, 36:181, 36:182, 36:183, 36:208, 36:216, *749*
\property_if_recorded_p:n 36:169, *749*
\property_if_recorded_p:nn 36:184, *749*
\property_new:nnnn 36:6, 36:6, 36:41, 36:232, 36:234, 36:235, 36:236, 36:237, 36:239, 36:241, 36:242, 36:243, 36:244, *747*
\property_record:nN . 36:48, 36:48, *748*
\property_record:nn 36:48, 36:49, 36:50, 36:52, 36:85, *748*
\property_ref:nn 36:89, 36:89, 36:96, 36:122, *748*
\property_ref:nnn 36:97, 36:97, 36:117, 36:125, *748*
\property_ref_undefined_warn: 36:199, 36:199, *748*
\property_ref_undefined_warn:n 36:206, 36:206, *748*
\property_ref_undefined_warn:nn . . . 36:214, 36:214, 36:230, 36:231, *747*
property internal commands:
__property_data:nnn 36:128, 36:140, 36:142, 36:147
__property_gset:nnnn 36:6, 36:10, 36:19, 36:22, 36:37
__property_record:nn 36:48, 36:51, 36:53, 36:62
__property_record_value:n 36:48, 36:59, 36:63
__property_record_value_aux:n 36:48, 36:64, 36:65, 36:79
__property_ref:nnn 36:91, 36:97, 36:99, 36:104, 36:116
l__property_ref_flag 36:88
\propto 30:405
\protect 14:246, 14:248, 14:250, 14:256, 14:262, 14:269, 14:279, 14:282, 14:288, 20:211, 21:26, 21:32, 21:51, 21:55, 21:209, 21:217, 28:690, 28:1238, 29:638, 32:143, 35:16, 35:33, 35:50, 37:280, 37:283, 37:317, 37:327, 41:264, 06:102, 44:12, 44:72, 44:82, 44:164, 44:171, 44:177, 45:17, 06:232, 06:246, 06:255, 06:260, 06:263, 06:264, 47:5, 06:266, 06:267, 06:272, 06:273, 06:278, 06:281, 06:282, 06:307, 06:345, 06:373, 52:281, 52:304, 06:580, 06:600, 53:83, 53:95, 53:125, 53:132, 53:152, 54:600, 54:659, 57:358, *1215*
\protected 17:6, 18:57, 18:238, 18:289, 18:474, 18:538, 21:308, 21:309, 04:186, 22:242, 25:682, 25:686, 25:689, 25:692, 25:695, 25:698, 25:701, 25:704, 28:1080, 29:573, 37:201, 37:247, 37:417, 38:199, 38:388, 38:426, 38:445, 38:446, 06:7, 41:56,

41:201, 41:208, 42:142, 06:455, 53:496, 53:497, 53:498, 02:463, *1261*
\Provide *1217*
\providecommand 08:2897, 08:2900, 21:6, 21:981, 35:98, 35:99, 35:100, 06:178, 54:2008
\ProvideCommandCopy *98*
\ProvideDocumentCommand 07:2804
\ProvideDocumentEnvironment . . . 07:2840
\ProvideExpandableDocumentCommand 07:2923
\ProvideHook 08:2843, *1257*
\ProvideMirroredHookPair 08:2843
\ProvideReversedHook 08:2843
provides commands:
provides_module 04:298
\provides_module *45*
\ProvidesClass *976*
\ProvidesClass 50:421
\ProvidesExplPackage 52:541
\ProvidesFile . . 30:672, 30:674, 30:675, 30:676, 33:1214, 50:430, 01:73, *1220*
\ProvidesPackage *976*
\ProvidesPackage 26:13, 33:815, 33:846, 50:344, 50:423, 50:425, 50:1806, 52:570, 53:491, *1202*
\ProvideTextCommand 21:3, 21:60, *1217*
\ProvideTextCommandDefault . . 21:57, *1226*
\Psi 30:313
\psi 30:296
\PushDefaultHookLabel 08:2772, *202*
\pushtabs 14:256, 41:139, 41:138, 41:158, 41:160
\pushtracing 26:117, 26:321
\put 42:56, 42:58, 42:70, 42:72, 42:325, 42:326, 42:327, 42:328, 42:336, 42:338, 42:351, 42:352, 42:353, 42:354, 42:361, 42:364, 42:384, 42:385, 42:386, 42:387, 42:393, 42:395, 42:407, 42:408, 42:409, 42:410, 42:415, 42:420, 42:728, 42:784, 42:812, 42:829, *1065*

Q

\qbezier *862*
\qbezier 42:681, 42:813, 42:830
\qbeziermax . 42:680, 42:706, 42:707, 42:767
\qquad 18:549
\quad 18:549, 38:168, 38:170, 38:190, 44:111, *1224*
quark commands:
\q_mark 07:1300, 07:1303, 07:2109, 07:2113, 07:2125, 07:2455, 07:2463, 07:2466, 07:2471
\q_nil 11:79, 11:92, 07:601, 07:1292, 07:1303, 07:1470, 07:1617, 07:1638, 07:1642, 07:1648, 07:1657, 07:1659, 07:2029, 07:2053, 07:2062, 07:2094, 07:2109, 07:2113, 07:2125, 07:2131, 07:2157, *171*
\quark_if_nil:NTF 07:612, 07:2115, *171*
\quark_if_nil:nTF 07:1467
\quark_if_recursion_tail_stop:N 07:803, 07:1451
\quark_if_recursion_tail_stop:n 36:144, 51:7, 07:479, 07:665, 07:1218, 07:2468, 07:2469, 08:422
\quark_if_recursion_tail_stop_-do:Nn 07:620
\quark_if_recursion_tail_stop_-do:nn 09:406, 09:425, 07:505, 07:510, 07:519, 07:528, 07:574, 07:587, 07:597, 07:641
\quark_new:N 09:15, 09:16, 11:40, 07:46, 07:47, 07:1626
\q_recursion_stop 09:395, 36:140, 51:20, 07:332, 07:343, 07:456, 07:793, 07:1214, 07:1447, 07:1709, 07:1724, 07:1726, 07:1733, 07:2458, 07:2464, 08:428
\q_recursion_tail 09:395, 36:140, 51:19, 07:456, 07:793, 07:1214, 07:1447, 07:1724, 07:1728, 07:2454, 07:2457, 07:2463, 07:2464, 08:427, 08:428, *129*
\q_stop 11:79, 11:92, 07:420, 07:601, 07:610, 07:615, 07:1300, 07:1302, 07:1642, 07:1659, 07:2266
quark internal commands:
\q__cmd 07:404, 07:408, 07:994, 07:1330, 07:1331, 07:1413, 07:1421, 07:1624, 07:2013, 07:2014, 07:2017, 07:2022, 07:2028, 07:2029, 07:2031, 07:2035, 07:2040, 07:2043, 07:2046, 07:2057, 07:2061, 07:2062, 07:2065, 07:2066, 07:2069, 07:2080, 07:2083, 07:2087, 07:2098, 07:2100, 07:2101, 07:2102, 07:2103, 07:2104, 07:2105, 07:2106, 07:2110, 07:2129, 07:2130, 07:2131, 07:2132, 07:2133, 07:2134, 07:2135, 07:2136, 07:2137, 07:2138, 07:2139, 07:2140, 07:2141, 07:2142, 07:2145, 07:2150, 07:2156, 07:2157, 07:2163, 07:2168, 07:2169, 07:2172, 07:2183, 07:2190, 07:2195, 07:2196, 07:2197, 07:2200, 07:2201, 07:2203, *168*
\q__cmd_recursion_stop 07:47, 07:2335,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

07:2337, 07:2346, 07:2369, 07:2378, *177*
\q__cmd_recursion_tail 07:46, 07:2335, *116*
\q__hook_recursion_stop 09:15, 09:326, 09:327, 09:336, 09:361
\q__hook_recursion_tail 09:15, 09:326, 09:340
\q__template_nil . . . 11:40, 11:881, *352*
\quotedblbase 21:555, 21:783, 21:1176
\quotesinglbase 21:556, 21:1173

R

\r 21:236, 21:388, 21:431, 21:470, 21:610, 21:637, 21:647, 21:673, 21:756, 21:795, 21:1245, 21:1263, 21:1289, 21:1411, 21:1412, 33:183, 33:205, 02:443, 02:444, *1213*
\radical . . 28:1104, 28:1107, 28:1137, *1245*
\raggedbottom 49:82
\raggedleft . 37:460, 37:478, 37:489, 37:496
\raggedright 37:455, 37:474, 37:488, 37:494
\raise . . 21:327, 21:359, 21:430, 21:433, 21:732, 21:797, 21:895, 21:1253, 29:651, 30:472, 30:520, 30:522, 38:73, 40:579, 40:588, 42:61, 42:73, 42:105, 42:117, 42:195, 42:305, 42:452, 42:494, 42:525, 42:550, 42:618, 42:635, 42:636, 42:739, 42:795
\raisebox *819*
\raisebox 21:872, 21:1222, 40:556
\rangle 30:599
\RawIndent 16:136, 16:172, *406*
\RawNoIndent 16:136
\RawNoindent 16:137, 16:173, *406*
\RawParEnd . . . 16:136, 16:174, 02:482, *406*
\RawShipout 53:151, 53:438, *1065*
\rbrace 21:309, 30:603
\rbrack 02:441
\rceil 30:607
\Re 30:320
\read *462*
\ReadonlyShipoutCounter 53:346, 53:440, *1064*
\RecordProperties 36:80, 36:270, *749*
\Ref *739*
\Ref 35:112, 35:149, 35:151, 35:170, 35:178, 35:182, 35:189, 35:193, 35:200
\ref 35:10, 35:178, 35:189, 45:553, *746*
\RefProperty 36:118, 36:271, *748*
\refstepcounter *512*
refstepcounter (socket) 35:117
\refstepcounter 35:111, 35:172, 35:182, 35:184, 35:193, 35:195, 38:350, 38:507, 39:226, 43:37, 44:59, 45:9, *1164*
refstepcounter/target (socket) . . . 35:118
\RefUndefinedWarn . . . 36:214, 36:272, *748*
regex commands:
\regex_match:nnTF 11:383
\registernumber *44*
registernumber 04:389
\relax *1217*
\relax␣ *1211*
\Relbar 30:483, 30:491, 30:493, 30:499, *1239*
\relbar 30:480, 30:495, 30:497
\relpenalty 02:324
remove commands:
remove_from_callback 04:907
\remove_from_callback *46*
\RemoveFromHook 08:2770, 08:2899, 37:39, 37:40, 37:41, 37:104, 37:105, 37:106, *198*
\removelastskip 02:514, 02:516, 02:518, 02:520
\renewcommand *81*
\renewcommand 30:74, 30:76, 30:78, 30:79, 30:81, 30:83, 30:85, 30:86, 30:92, 30:94, 30:96, 30:97, 30:110, 30:111, 30:112, 30:120, 30:121, 38:455, 38:475, 38:496, 06:124, *98*
\RenewCommandCopy 06:473, 06:475, *98*
\RenewDocumentCommand 07:2804
\RenewDocumentEnvironment 07:2602, 07:2840
\renewenvironment *82*
\renewenvironment . 38:504, 38:516, 06:152
\RenewEnvironmentCopy 06:514, 06:516, *1264*
\RenewExpandableDocumentCommand 48:484, 48:485, 48:486, 48:487, 07:2923, *190*
\repeat 04:154, 04:164, 24:714, 41:382, 50:1246, 50:1307, 50:1377, 50:1440, 50:1491, 50:1529, 57:369, 57:380, 57:390, 57:401, 57:431, 57:457, 57:467, 01:65, 01:67, 02:490
\requestedLaTeXdate 08:2919, 50:1592, 50:1632, 50:1652, 50:1739
\requestedpatchdate 50:1662, 50:1740
\RequirePackage *976*
\RequirePackage 04:24, 50:676, 50:683, 50:726, 50:735, 50:1189, 54:2005, *1065*
\RequirePackageWithOptions *976*
\RequirePackageWithOptions 50:718
reserved@a commands:
\reserved@a: 20:233, 20:305, 20:367, 20:414, 50:1219, 50:1354
reserved@b commands:
\reserved@b: 50:232, 50:249

reserved@c commands:
\reserved@c: 20:755
\RestoreAtCatcode *1209*
\restorecr 18:556
\ResumeTagging *1161*
\ResumeTagging 23:22, 55:3, *1161*
\ReverseBoolean 07:3001
\reversemarginpar 45:387
\rfloor 30:611
\rgroup 30:615
\rhd 29:733
\rho 30:290
\rhook 30:486, 30:487
\right 30:632, 30:634, 30:636, 30:638, 30:643, 30:644, 30:645, 30:646, 38:168, 38:173, 38:197
\Rightarrow 30:418, 30:493, 30:505
\rightarrow 30:445, 30:447, 30:451, 30:485, 30:495, 30:503, 30:556
\rightarrowfill 30:538, 30:554
\rightharpoondown 30:462
\rightharpoonup 30:461, 30:473
\righthyphenmin 56:11, 02:371
\rightleftharpoons 30:471
\rightline 40:614
\rightmargin 39:9, 39:40, 39:51, *803*
\rightmark 49:77, *947*
\rightskip 24:685, 37:452, 37:456, 37:462, 37:472, 37:475, 37:480, 37:529, 37:551, 39:75, 40:392, 40:413, 44:207, 44:230, 02:417, 02:528
\rlap 21:430, 21:433, 21:797, 38:428, 38:456, 40:618, 41:81, *1225*
\rm *1201*
\rmdefault 29:6, 29:208, 29:398, 29:407, 29:439, 29:450, 29:482, 30:58, 30:127, 33:7, 33:613, *648*
\rmfamily . . 29:4, 29:5, 29:405, 29:448, 29:449, 29:480, 29:481, 32:15, *1226*
rmfamily 29:420
\rmmath *1194*
\rmoustache 30:572
\rmsubstdefault 30:18, 30:30, 33:28, 33:39, 33:85
\Roman *512*
\Roman 22:186, *1064*
\roman *512*
\roman 22:185
\romannumeral 22:191, 22:192, 37:316, 37:348, 37:365, 39:43, 39:258, 39:269, *771*
\root 38:66, 38:356
\rootbox 38:66
\rq 02:439
\rule *819*
\rule 40:490, 40:507, 40:524, 40:530, 45:477, 45:495, 45:513

S

\S 21:311, *1218*
\safesubencodingfoundfalse 33:1255
\safesubencodingfoundtrue 33:1245
\samepage *421*
\samepage 18:11, 18:29, *1195*
\SaveAtCatcode *1209*
\savebox *818*
\savebox 40:155
\savecatcodetable . 04:117, 04:168, 04:170
\savepos *751*
\sb 38:212
\sbox *818*
\sbox 19:4, 21:498, 21:514, 39:229, 40:161, 40:168, 40:172, 40:177, 40:182, 02:522, *1216*
\sc *566*
scan commands:
\scan_new:N 11:38, 11:39, 08:44
\scan_stop: 08:2530, 08:2531, 09:452, 09:453, 09:511, 09:512, 09:547, 09:548, 16:94, 48:104, 48:106, 48:108, 48:109, 48:251, 48:252, 48:253, 51:85, 51:97, 53:44, 53:325, 53:327, 53:328, 07:1856, 07:1857, 08:661, 08:668, 08:679, 08:688, 08:723, 08:734, 08:770, 08:782, 08:792, 08:822, 08:854, 08:1196, *412*
scan internal commands:
\s__file_stop 52:103, 52:105, 52:108, 52:110, 52:112, 52:125
\s__hook_mark 08:2247, 08:2261, 08:2310, 08:2318, 08:2421, 08:2423, 08:2484, 08:2486, 08:2490, 08:2491, 08:2498, 08:2499, 08:2512, 08:2513, 09:24, 09:26, 09:35, 09:37, 09:113, 09:118, 09:312, 09:323, 09:457, 09:460, 09:472, 09:476, 09:481, 09:516, 09:536, 09:552, 09:564, 08:44, 08:45, 08:46, 08:348, 08:351, 08:354, 08:358, 08:361, 08:362, 08:712, 08:763, 08:765, 08:773, 08:775, 08:778, 08:780, 08:882, 08:907, 08:908, 08:912, 08:1217, 08:1234, 08:1238, 08:1280, 08:1281, 08:1336, *224*
\s__keys_stop . . . 51:47, 51:175, 51:176
\s__template_mark . . 11:38, 11:881, *352*
\s__template_stop 11:39, 11:286, 11:297, 11:308, 11:329,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

11:343, 11:351, 11:430, 11:433, 11:740, 11:743, 11:881, 11:885, *352*
\scdefault 25:694, 29:27, 30:102
\scriptfont 26:338
\scriptscriptfont 26:339
\scriptscriptstyle 38:65, 38:68
\scriptspace 02:399
\scriptstyle 30:344, 38:64
\scshape 21:300, 25:692, 25:693, 29:25, 29:26, 32:23, *1235*
\searrow 30:413
\sec 38:20
\secdef 44:142
\secondoftwo *457*
\section *304*
\sectionmark 44:143, *946*
\selectfont 19:7, 21:302, 21:329, 21:360, 21:449, 21:809, 21:871, 21:1221, 21:1255, 21:1573, 24:291, 24:301, 24:311, 25:528, 25:533, 25:538, 25:684, 25:688, 25:691, 25:694, 25:697, 25:700, 25:703, 25:706, 26:114, 26:115, 26:159, 26:162, 26:164, 29:6, 29:9, 29:12, 29:15, 29:18, 29:21, 29:24, 29:27, 29:30, 29:264, 29:287, 29:325, 29:345, 29:360, 29:371, 29:382, 29:385, 29:409, 29:414, 29:419, 29:452, 29:457, 29:462, 29:476, 29:479, 29:482, 29:485, 29:488, 29:564, 29:641, 29:665, 29:682, 29:697, 33:36, 33:94, 33:104, 33:636, 33:673, 33:913, 33:930, 45:403, 45:424, *222*
selectfont 26:150
seq commands:
\seq_clear:N 11:160, 11:211, 08:1557, 08:1639
\seq_clear_new:N 08:1565, 08:1647
\seq_const_from_clist:Nn 11:16
\seq_count:N 55:130
\seq_gclear_new:N 11:115
\seq_gpop:NN 08:2561
\seq_gpop:NNTF . . 52:72, 08:451, 08:459
\seq_gpop_right:NN 08:430
\seq_gpush:Nn 08:2549, 08:2555, 52:64, 08:444
\seq_gput_right:Nn 48:23, 08:108, 08:137, 08:419, 08:423
\seq_gset_eq:NN 11:116
\seq_if_empty:NTF 08:418, 08:469
\seq_if_exist:NTF 11:155, 52:60
\seq_if_in:NnTF . 11:230, 48:9, 48:218
\seq_map_break: 11:249, 11:340
\seq_map_function:NN 11:226, 11:346, 11:707
\seq_map_inline:Nn 08:2942, 48:124, 48:157, 48:195, 48:206, 48:354, 48:430, 07:2227, 08:1426, 08:1453, 08:1581, 08:1599, 08:1664, 08:1687
\seq_mapthread_function:NNN . . . *180*
\seq_new:N 08:2539, 10:36, 11:29, 48:6, 52:61, 55:133, 07:2211, 08:28, 08:33, 08:1548
\seq_put_right:Nn . . 10:86, 11:263, 08:1563, 08:1645, 08:1748, 08:1755
\seq_set_eq:NN 11:157
\seq_set_split:Nnn . . . 55:129, 07:2225
\seq_use:Nnnn . 10:60, 08:1808, 08:1813
seq internal commands:
\g__mark_classes_seq 48:6, 48:9, 48:23, 48:124, 48:157, 48:195, 48:206, 48:218, 48:354, 48:430, *950*
\series *1194*
\seriesdefault 21:1574, 28:390, 29:220, 29:222, 29:660, 29:678, 29:694, 29:711, 29:773, 30:127, *642*
\setattribute *43*
\setattribute 04:82, 04:239
\setbox *812*
\setbox... . . . . *1233*
\setbox0 . . . . . *1229*
\setcounter *512*
\setcounter 20:445, 22:2, 22:42, 22:62, 28:143, 39:249, 54:2328, 54:2331, 54:2334, 54:2338, *607*
\SetDefaultHookLabel 08:2772, *200*
\SetKeys 20:751, 51:255, *1028*
\setlength *521*
\setlength 18:84, 18:244, 18:259, 18:518, 23:4, 38:524, 38:529, 38:530, 38:531, 40:47, 40:62, 40:247, 40:266, 40:333, 40:336, 40:360, 40:363, 40:439, 40:546, 40:547, 40:548, 40:577, 40:578, 40:585, 40:586, 40:587, 41:176, 41:384, 54:2344, 54:2345, 54:2346, 54:2349, 54:2350, 54:2354, 54:2355, 54:2356, 54:2360, 54:2361, 54:2362, *426*
\SetMathAlphabet 24:11, 27:140, 27:141, 28:695, 30:162, 30:163
\setminus 30:400
\SetProperty 36:38, 36:268, *749*
\setrangecatcode 04:96, 04:104, 04:113, 04:114
\SetSymbolFont 28:530, 30:152, 30:153, 30:154, *1198*
\SetTemplateKeys 11:1194, *349*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\settodepth *521*
\settodepth 23:17
\settoheight *521*
\settoheight 23:17
\settowidth *521*
\settowidth 23:17
\sf *1201*
\sfcode 18:463, 20:45, 20:115, 20:173, 57:250, 57:545, 02:431, 02:432, 02:433, 02:434, 02:551
\sfdefault 29:9, 29:212, 29:399, 29:412, 29:440, 29:455, 29:485, 30:58
\sffamily . . 29:7, 29:8, 29:410, 29:453, 29:454, 29:483, 29:484, 32:16, *635*
sffamily 29:420
\sfsubstdefault 30:19, 30:31, 33:30, 33:87
\shape *1196*
\shapedefault 21:1574, 25:684, 28:391, 29:661, 29:679, 29:695, 29:712, 30:127, *642*
\sharp 30:337
\shipout 53:4, 53:52, 53:431, 53:434, 54:606, 54:664, *1215*
shipout 53:153
shipout commands:
\l_shipout_box 53:23, 53:35, 53:38, 53:50, 53:61, 53:126, 53:149, 53:179, 53:184, 53:207, 53:208, 53:215, 53:226, 53:229, 53:230, 53:234, 53:241, 53:251, 53:259, 53:266, 53:276, 53:282, 53:283, 53:286, 53:293, 53:295, 53:296, 53:302, 53:304, *1060*
\l_shipout_box_dp_dim 53:194, 53:197, 53:199, 53:230, 53:283, 53:512, *1060*
\l_shipout_box_ht_dim 53:193, 53:197, 53:199, 53:229, 53:249, 53:282, 53:511, *1060*
\l_shipout_box_ht_plus_dp_dim 53:196, 53:199, 53:215, 53:266, 53:277, 53:279, *1060*
\l_shipout_box_wd_dim . . . 53:195, 53:199, 53:241, 53:293, 53:513, *1060*
\shipout_debug_off: 53:7, 53:13, 53:415, *1064*
\shipout_debug_on: 53:7, 53:8, 53:414, *1064*
\shipout_discard: 53:343, 53:343, 53:411, *1063*
\g_shipout_readonly_int 36:233, 53:102, 53:104, 53:111, 53:116, 53:346, 53:355, 53:359, 53:363, 53:367, 53:371, *1064*
\g_shipout_totalpage_int *1064*
\g_shipout_totalpages_int 53:87, 53:348, *1064*
shipout internal commands:
__shipout_add_background_box:n . 53:180, 53:206, 53:206, 53:338, 53:420
__shipout_add_background_picture:n 53:69, 53:337, 53:337, 53:424
__shipout_add_firstpage_material:Nn 53:173, 53:189, 53:189, 53:413, 53:418
__shipout_add_firstpage_specials: 53:110, 53:166, 53:178, 53:178, 53:181, *1071*
__shipout_add_foreground_box:n . 53:117, 53:257, 53:257, 53:341, 53:422
__shipout_add_foreground_picture:n 53:64, 53:340, 53:340, 53:426
__shipout_debug:n 53:7, 53:7, 53:20, 53:103, 53:115, 53:148, *1067*
\g__shipout_debug_bool 53:6, 53:10, 53:15, 53:21
__shipout_debug_gset: 53:7, 53:11, 53:16, 53:18
\g__shipout_discard_bool 53:81, 53:88, 53:90, 53:203, 53:344
__shipout_drop_firstpage_specials: 53:127, 53:167, 53:178, 53:183, 53:185, *1074*
__shipout_excuse_extra_page: 53:384, 53:392, 53:392
__shipout_execute: 53:46, 53:46, 53:52, *1068*
__shipout_execute_cont: 53:57, 53:59, 53:59
__shipout_execute_main_cont:Nnnn 53:60, 53:77, 53:77, 53:147, *1072*
__shipout_execute_nohooks_cont: 53:144, 53:146
__shipout_execute_raw: 53:135, 53:135, 53:151, *1072*
__shipout_execute_test_level: 53:49, 53:54, 53:54
__shipout_execute_test_level_raw: 53:135, 53:138, 53:141
__shipout_finalize_box: 53:26, 53:28, 53:44, 53:124
\l__shipout_firstpage_box 53:170, 53:180, 53:187, *1074*
__shipout_get_box_size:N . 53:85, 53:107, 53:192, 53:192, 53:207, *1075*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\l__shipout_group_level_tl 53:47, 53:53, 53:56, 53:136, 53:143
\c__shipout_horigin_tl 53:310, 53:325
__shipout_init_page_origins: 53:310, 53:310, 53:321, 53:324
\g__shipout_lastpage_handled_bool 53:120, 53:188, 53:365
__shipout_picture_overlay:n 53:323, 53:323, 53:338, 53:341
\l__shipout_raw_box 53:25, 53:139, 53:147, 53:149, 53:179, 53:184, *1071*
__shipout_run_firstpage_hook: .. . 53:108, 53:163, 53:163, 53:172, *1071*
\l__shipout_saved_badness_tl 53:204, 53:210, 53:218, 53:231, 53:236, 53:244, 53:253, 53:261, 53:269, 53:281, 53:288, 53:298, 53:306
__shipout_saved_protect: 53:83, 53:132, 53:152, 53:152
\l__shipout_tmp_box 53:204, 53:217, 53:219, 53:220, 53:221, 53:225, 53:243, 53:245, 53:246, 53:247, 53:250, 53:268, 53:270, 53:271, 53:272, 53:278, 53:297, 53:299, 53:300, 53:301, 53:303, 53:329, 53:331, 53:332, 53:333
\c__shipout_vorigin_tl 53:310, 53:327
shipout/after 53:153, *1060*
shipout/background 53:153, *1060*
shipout/before 53:153, *1060*
shipout/firstpage 53:153, *1060*
shipout/foreground 53:153, *1060*
shipout/lastpage 53:153, *1060*
\ShipoutBox ... 53:23, 53:439, 53:494, *1060*
\ShipoutBoxDepth 53:459, 53:511
\ShipoutBoxHeight .. 53:458, 53:511, *1086*
\ShipoutBoxWidth 53:460, 53:511
\shortstack 42:132, 42:147, 42:152, 42:814, 42:831
\show 06:552, 06:625, *100*
show commands:
\show_hook:n *275*
\showbox 54:1919
\showboxbreadth 02:631, 02:655, 02:672, 02:697, 54:1919, 02:387, 02:554
\showboxdepth 02:630, 02:654, 02:671, 02:698, 24:672, 24:717, 24:734, 54:1919, 02:388, 02:554
\ShowCommand . 06:545, 07:1311, 07:1340, 07:1352, 07:1360, 07:1363, 07:1524, *100*
\ShowEnvironment 06:680, *104*
\ShowFloat 54:1900
\ShowHook 08:2796, 08:2904, *205*
\showhyphens 24:661, *1268*
\ShowInstanceValues 11:1176, *349*
\ShowMarksAt 48:358
\showoutput 02:553, *1226*
\showoverfull 02:577, 02:612, 02:620, 02:552, 02:555, *1226*
\ShowSocket ... 10:149, 10:170, 10:183, *335*
\ShowTemplateCode 11:1176, *350*
\ShowTemplateDefaults 11:1176, *350*
\ShowTemplateInterface 11:1176, *350*
\ShowTemplateVariables 11:1176, *350*
\showtokens 06:660, *105*
\Sigma 30:310
\sigma 30:291
\sim 30:452, 30:464
\simeq 30:453
\sin 38:9
\sinh 38:11
\size *1194*
\skew 30:551
\skip 04:33, 40:464, 45:391, 54:318, 54:494, 02:28, 02:53, 02:92, 02:209, 02:250, 02:295
skip commands:
\skip_eval:n 11:370, 05:165
\skip_new:N 11:36
\skip_set:Nn 16:25
\skip_zero:N 16:58, 53:222, 53:223, 53:224, 53:273, 53:274, 53:275
\l_tmpa_skip *346*
\c_zero_skip 16:26
\skipdef 04:229, 02:45, 02:53, 02:92
\skipeval *77*
\skipeval 05:158, 05:175, *77*
\skipzero 04:229
\slash 06:890, 06:911, 02:507
\sldefault 25:691, 29:24, 30:102
\SLiTeX *1216*
\sloppy 40:397, 40:416, 49:86, 49:91
sloppypar (env.) 49:91
\sloppypar 49:91
\slshape 21:440, 21:800, 25:689, 25:690, 29:22, 29:23, 32:22, 33:627
\small *198*
\smallbreak 06:891, 06:912, 02:515
\smallint 30:364
\smallskip 18:447, 02:516
\smallskipamount .. 18:447, 18:450, 02:515
\smash 17:29, 17:51, 30:480, 30:554, 30:555, 30:558, 30:559, 38:126, *788*
\smile 30:457
socket commands:
\socket_assign_plug:nn 10:42, 10:44, 10:116, 10:116, 10:174, *334*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\socket_debug_off: 10:7, 10:14, 10:177, *335*
\socket_debug_on: 10:7, 10:9, 10:176, *335*
\socket_log:n 10:53, 10:53, 10:72, 10:172, *335*
\socket_new:nn 10:27, 10:27, 10:170, *334*
\socket_new_plug:nnn 10:38, 10:41, 10:73, 10:73, 10:173, *334*
\socket_set_plug:nnn 10:73, 10:95, 10:163, *334*
\socket_show:n 10:53, 10:72, 10:171, *335*
\socket_use:n 10:132, 10:138, *335*
\socket_use:nn . . . 10:132, 10:139, *335*
\socket_use:nnn . . 10:132, 10:140, *335*
\socket_use:nnnn . . 10:132, 10:141, *335*
\socket_use:nw 10:132, 10:132, 10:138, 10:139, 10:140, 10:141, 10:175, *339*
\socket_use_expandable:n 10:142, 10:145, *335*
\socket_use_expandable:nw 10:142, 10:142, 10:145, *335*

socket internal commands:

__socket_debug:n 10:7, 10:7, 10:21, *336*
\g__socket_debug_bool 10:6, 10:11, 10:16, 10:22, 10:24
__socket_debug_gset: 10:7, 10:12, 10:17, 10:19
__socket_debug_term:n 10:7, 10:8, 10:23, 10:45, 10:87, 10:105, 10:121, 10:133, *336*

Sockets:

refstepcounter 35:117
refstepcounter/target 35:118
tagsupport/caption/begin 55:94
tagsupport/caption/end 55:94
tagsupport/caption/label/begin 55:96
tagsupport/caption/label/end . 55:96
tagsupport/float/begin 55:92
tagsupport/float/end 55:92
tagsupport/float/hmode/begin . 55:90
tagsupport/float/hmode/end . . . 55:90
tagsupport/para/begin 55:49
tagsupport/para/end 55:49
tagsupport/para/restore 55:39
tagsupport/refstepcounter 55:49
tagsupport/tbl/cell/begin 55:69
tagsupport/tbl/cell/end 55:69
tagsupport/tbl/colspan 55:79
tagsupport/tbl/finalize 55:77
tagsupport/tbl/hmode/begin . . . 55:80
tagsupport/tbl/hmode/end 55:80
tagsupport/tbl/init 55:75
tagsupport/tbl/init/celldata . 55:76
tagsupport/tbl/leaders/begin . 55:88
tagsupport/tbl/leaders/end . . . 55:88
tagsupport/tbl/longtable/finalize 55:84
tagsupport/tbl/longtable/foot . 55:86
tagsupport/tbl/longtable/head . 55:86
tagsupport/tbl/longtable/init . 55:84
tagsupport/tbl/pcell/end 55:69, 55:69
tagsupport/tbl/restore/celldata 55:78
tagsupport/tbl/row/begin 55:69
tagsupport/tbl/row/end 55:69
tagsupport/tbl/vmode/begin . . . 55:80
tagsupport/tbl/vmode/end 55:80
tagsupport/toc/contentsline/after 55:63
tagsupport/toc/contentsline/before 55:63
tagsupport/toc/leaders/after . 55:67
tagsupport/toc/leaders/before . 55:67
tagsupport/toc/starttoc/after . 55:65
tagsupport/toc/starttoc/before 55:65

\sourceLaTeXdate 08:2918, 03:164, 50:65, 50:103
\sp 38:212
\space 02:447, *1212*
\spacefactor 17:28, 17:30, 17:50, 17:52, 18:130, 18:139, 18:158, 18:172, 18:184, 18:198, 18:212, 18:463, 18:504, 18:509, 21:70, 21:73, 45:531, 45:533, 02:505, 02:506, *1237*
\spacefactor␣in␣math␣mode␣gh/643 . *1260*
\spaceskip 33:6, 33:612, 02:421
\spadesuit 30:341
\span 41:383
\special 17:26, 17:48, *1061*
\SplitArgument 07:3001
\splitfirstmark 54:2249
\SplitList 07:3001, *192*
\splitmaxdepth 45:469, 45:488, 45:506, 54:2243, 02:394
\splittopskip 45:468, 45:487, 45:505, 02:419
\sqcap 30:383
\sqcup 30:384
\sqrt 38:355, *1225*
\sqrtsign 30:536, 38:71, 38:355, *1225*
\sqsubset 29:729
\sqsubseteq 30:406
\sqsupset 29:730
\sqsupseteq 30:407
\SS . . 21:304, 21:534, 21:1157, 57:650, *1230*
\ss 21:254, 21:406, 21:557, 21:784, 21:1132, 57:650
\sscdefault 25:536, 25:609, 25:706

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\sscshape 25:536, 25:608, 25:704, 25:705, 32:31
\stackrel 38:353
\stamp *1204*
\star 30:404
\stepcounter *512*
\stepcounter 22:17, 22:27, 24:626, 28:48, 35:124, 35:134, 35:141, 35:172, 35:184, 35:195, 38:363, 38:423, 38:517, 45:452, 45:520, 54:650, 54:709, *1197*
\stockheight 54:95
\stockwidth 54:95
\stop 33:1636, 37:240
\storedpar 04:156, 04:161
str commands:
\c_backslash_str 09:147, 09:398, 07:2554
\c_hash_str 09:481, 09:485, 09:509, 07:1384, 07:1510
\c_right_brace_str 08:2056
\str_case:nn 09:595, 11:510, 57:562, 07:1119
\str_case:nnTF 09:40, 11:448, 36:29, 51:30, 07:2789
\str_case_e:nnTF 07:2475
\str_count:n 09:161, 09:236, 08:37, 08:43, 08:1249, 08:1853
\str_gset:Nn 08:2781
\str_head:n 07:2420
\str_if_empty:NTF 57:584
\str_if_eq:nn *263*
\str_if_eq:NNTF 52:555
\str_if_eq:nnTF 08:2116, 08:2312, 08:2516, 08:2570, 08:2640, 09:215, 09:287, 09:369, 09:520, 09:527, 09:556, 09:559, 11:79, 11:92, 11:243, 11:264, 11:462, 11:477, 11:493, 11:565, 11:722, 11:724, 11:898, 05:137, 52:131, 52:549, 52:559, 07:100, 07:102, 07:425, 07:1198, 07:1203, 07:1615, 07:2006, 07:2019, 07:2021, 07:2051, 07:2071, 07:2073, 07:2092, 07:2119, 07:2147, 07:2149, 07:2174, 07:2176, 07:2199, 07:2316, 07:2328, 07:2356, 07:2532, 08:346, 08:438, 08:479, 08:535, 08:537, 08:608, 08:610, 08:740, 08:798, 08:831, 08:884, 08:966, 08:973, 08:1001, 08:1008, 08:1801, 08:1909, 08:2009, *168*
\str_if_eq_p:nn . . 07:485, 07:486, 07:487, 07:488, 07:935, 07:2364, 07:2386, 08:888, 08:1592, 08:1678
\str_if_exist:NTF 10:28, 10:55, 10:74, 10:96, 10:117
\str_lowercase:n 57:596
\str_map_function:NN 05:144
\str_new:N 10:35, 07:19
\str_replace_all:Nnn 09:508
\str_set:Nn 10:124, 52:399, 52:400, 57:582, 57:586, 07:228, 07:242, 07:243, 07:287
\str_tail:n 07:2006, 07:2417, 07:2443, *168*
\str_use:N 10:62, 10:65, 10:122, 10:135, 10:136, 10:143
str internal commands:
__str_if_eq:nn 08:23, *223*
\strcmp 50:346, 50:362
\stretch 18:532
\string *1232*
\stripmeaning *1201*
\strut 38:191, 38:192, 41:29, 06:892, 06:913, 02:523, *222*
\strutbox 26:189, 40:490, 40:507, 40:524, 41:186, 41:187, 45:469, 45:477, 45:488, 45:495, 45:506, 45:513, 02:523
\subencodingresult . . 33:1236, 33:1241, 33:1410, 33:1414, 33:1416, 33:1417
\subparagraphmark 44:143
\subsectionmark 44:143, *946*
\subset 30:433
\subseteq 30:435
\substdefault *1250*
\subsubsectionmark 44:143
\succ 30:427
\succeq 30:430
\sum 30:357
\sup 38:24
\superject *1228*
\suppressfloats 54:2010
\supset 30:432
\supseteq 30:434
\surd 30:343
\SuspendTagging *1161*
\SuspendTagging 23:22, 55:3, *1161*
\swarrow 30:415
\swdefault 25:531, 25:607, 25:703
\swshape 25:531, 25:606, 25:701, 25:702, 32:30
\symbol 21:162, 29:619
\symletters . . . 33:8, 33:14, 33:614, 33:1136
\symoperators 30:650
sys commands:
\sys_if_engine_luatex:TF 53:26

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

T

\T 14:23, 21:335, 21:337, 21:339, 21:341, 21:343, 21:345, 21:347, 21:349, 21:351, 21:374, 50:1572, 50:1576, 50:1577
\t 21:282, 21:741, 21:849, 33:157, 33:158, 33:185, 33:189, 33:191, 33:203, 33:206, 33:208, 33:688, 33:856, 33:1123, 33:1125, 33:1318, *1218*
tabbing (env.) 41:71
\tabbing 41:71
\tabbingsep 41:130, 41:132, 41:166
\tabcolsep 41:259, 41:338
\tableofcontents *766*
\tabskip 38:208, 38:209, 38:369, 38:372, 38:375, 38:377, 38:522, 38:535, 38:538, 38:540, 41:167, 41:192, 02:420, 02:534, *1198*
tabular (env.) 41:174
\tabular 41:174
\tabular* 41:175
\tabularnewline 41:194, 41:207, *1222*
tag commands:
  \tag_get:n 55:58
  \tag_if_active: 55:10
  \tag_if_active:TF 55:8, 55:135
  \tag_resume:n 55:3, 55:4, 55:6
  \tag_socket_use:n . . . 55:7, 55:15, *1162*
  \tag_socket_use:nn . . 55:7, 55:16, *1162*
  \tag_socket_use:nnn . 55:7, 55:17, *1162*
  \tag_socket_use_expandable:n 55:7, 55:14, *1162*
  \tag_suspend:n 55:3, 55:3, 55:5
tag internal commands:
  \l__tag_block_flattened_level_int 55:38, 55:45
  \l__tag_para_bool 55:46
  \l__tag_para_flattened_bool . . 55:44
  \l__tag_para_main_tag_tl 55:42
  \l__tag_para_tag_default_tl . . 55:43
  \l__tag_para_tag_tl 55:43
  \g__tag_struct_dest_num_prop . 55:56
\tag_resume:n *1161*
\tag_socket_use:n *1161*
\tag_socket_use:nn *1161*
\tag_socket_use:nnn *1161*
\tag_socket_use_expandable:n *1161*
\tag_suspend:n *1161*
tagsupport/caption/begin (socket) . 55:94
tagsupport/caption/end (socket) . . . 55:94
tagsupport/caption/label/begin (socket) 55:96
tagsupport/caption/label/end (socket) 55:96
tagsupport/float/begin (socket) . . . 55:92
tagsupport/float/end (socket) 55:92
tagsupport/float/hmode/begin (socket) 55:90
tagsupport/float/hmode/end (socket) 55:90
tagsupport/para/begin (socket) 55:49
tagsupport/para/end (socket) 55:49
tagsupport/para/restore (socket) . . 55:39
tagsupport/refstepcounter (socket) 55:49
tagsupport/tbl/cell/begin (socket) 55:69
tagsupport/tbl/cell/end (socket) . . 55:69
tagsupport/tbl/colspan (socket) . . . 55:79
tagsupport/tbl/finalize (socket) . . 55:77
tagsupport/tbl/hmode/begin (socket) 55:80
tagsupport/tbl/hmode/end (socket) . 55:80
tagsupport/tbl/init (socket) 55:75
tagsupport/tbl/init/celldata (socket) 55:76
tagsupport/tbl/leaders/begin (socket) 55:88
tagsupport/tbl/leaders/end (socket) 55:88
tagsupport/tbl/longtable/finalize (socket) 55:84
tagsupport/tbl/longtable/foot (socket) 55:86
tagsupport/tbl/longtable/head (socket) 55:86
tagsupport/tbl/longtable/init (socket) 55:84
tagsupport/tbl/pcell/end (socket) 55:69, 55:69
tagsupport/tbl/restore/celldata (socket) 55:78
tagsupport/tbl/row/begin (socket) . 55:69
tagsupport/tbl/row/end (socket) . . . 55:69
tagsupport/tbl/vmode/begin (socket) 55:80
tagsupport/tbl/vmode/end (socket) . 55:80
tagsupport/toc/contentsline/after (socket) 55:63
tagsupport/toc/contentsline/before (socket) 55:63
tagsupport/toc/leaders/after (socket) 55:67
tagsupport/toc/leaders/before (socket) 55:67
tagsupport/toc/starttoc/after (socket) 55:65
tagsupport/toc/starttoc/before (socket) 55:65
\tan 38:15
\tanh 38:17
target 36:239, 36:240, *751*
\tau 30:292

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

tbl commands:
\tbl_count_missing_cells:n 55:134, 55:134, 55:235, *1171*
\tbl_count_table_cols: 55:128, 55:128, *1168*
\tbl_crcr:n 55:232, 55:232
\tbl_gdecr_row_count: 55:192, 55:198
\tbl_gincr_row_count: 55:192, 55:195
\tbl_gzero_row_count: 55:192, 55:192
\tbl_if_row_was_started:TF 55:186, 55:186, 55:189
\tbl_inbetween_rows: . 55:201, 55:201
\tbl_init_cell_data_for_row: ... 55:182, 55:182
\tbl_init_cell_data_for_table . *1165*
\tbl_init_cell_data_for_table: .. 55:158, 55:158
\tbl_restore_outer_cell_data: ... 55:204, 55:204, *1165*
\tbl_save_outer_table_cols: 55:155, 55:155
\tbl_update_cell_data: 55:124, 55:124
\tbl_update_cell_data_for_next_-row: 55:178, 55:178
\tbl_update_multicolumn_cell_-data:n 55:221, 55:221
tbl internal commands:
\g__tbl_col_int 55:100, 55:125, 55:136, 55:141, 55:159, 55:175, 55:180, 55:183, 55:187, 55:190, 55:202, 55:205, 55:212, 55:222, 55:225, 55:228, 55:233, *1170*
\g__tbl_missing_cells_int 55:116, 55:138, 55:145, 55:149
\g__tbl_missingcells_int 55:116
\g__tbl_row_int 55:100, 55:160, 55:174, 55:179, 55:193, 55:196, 55:199, 55:206, 55:211, *1166*
\l__tbl_saved_col_tl 55:107, 55:159, 55:165, 55:205
\l__tbl_saved_row_tl 55:107, 55:160, 55:164, 55:206
\l__tbl_saved_span_tl 55:107, 55:161, 55:166, 55:207, 55:213
\l__tbl_saved_table_cols_tl 55:107, 55:156, 55:168, 55:170, 55:208
\g__tbl_span_tl .. 55:100, 55:125, 55:126, 55:142, 55:161, 55:176, 55:184, 55:207, 55:228, 55:230, *1171*
\g__tbl_table_cols_tl 55:100, 55:130, 55:131, 55:140, 55:156, 55:208, 55:215, 55:217, *1168*
\l__tbl_tmpa_seq 55:129, 55:130, 55:133
__tbl_trace:n ... 55:118, 55:121, 55:123, 55:131, 55:146, 55:163, 55:210
template internal commands:
__template_assign_boolean: 11:759, 11:759
__template_assign_boolean_aux:n 11:759, 11:762, 11:763, 11:765
__template_assign_choice: 11:785, 11:785
__template_assign_choice_-aux:nTF 11:785, 11:787, 11:790, 11:802, 11:807
__template_assign_function: ... 11:808, 11:808
__template_assign_function_-aux:N . 11:808, 11:811, 11:812, 11:814
__template_assign_instance: ... 11:825, 11:825
__template_assign_instance_-aux:N . 11:825, 11:828, 11:829, 11:831
__template_assign_variable: ... 11:729, 11:843, 11:843
__template_assign_variable:n ... 11:843, 11:845, 11:853
__template_assignments_pop: ... 11:908, 11:908, 11:1193
__template_assignments_push:n .. 11:667, 11:909, 11:909
\l__template_assignments_tl 11:19, 11:668, 11:706, 11:770, 11:778, 11:805, 11:816, 11:833, 11:858, 11:865, 11:908, 11:910, *351*
\c__template_code_root_tl 11:9, 11:54, 11:395, 11:398, 11:640, 11:642, 11:643, 11:669, 11:756, 11:912, *351*
__template_convert_to_assignments: 11:662, 11:704, 11:704, 11:755
__template_convert_to_assignments_-aux:n 11:704, 11:708, 11:710
__template_convert_to_assignments_-aux:nn 11:704, 11:713, 11:715, 11:734
__template_declare_instance:nnnn 11:645, 11:645, 11:1169
__template_declare_instance_-aux:nnnn 11:645, 11:651, 11:654, 11:701
__template_declare_template_-code:nnnn 11:374, 11:384, 11:386, 11:393
__template_declare_template_-code:nnnnn . 11:374, 11:374, 11:1161
__template_declare_template_-keys:nnnn .. 11:203, 11:203, 11:1159

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

__template_declare_type:nn 11:182, 11:186, 11:188
\l__template_default_tl .. 11:20, *351*
\c__template_defaults_root_tl 11:10, 11:99, 11:100, 11:139, 11:142, *351*
__template_define_type:nn 11:182, 11:182, 11:1157
__template_edit_defaults:nnn 11:598, 11:598, 11:1165
__template_edit_instance:nnn 11:686, 11:686, 11:1173
__template_edit_instance_-aux:nnnn ... 11:693, 11:698, 11:703
__template_edit_instance_-aux:nnnnn 11:686
\l__template_error_bool ... 11:21, 11:222, 11:228, 11:248, 11:277, 11:290, 11:615, 11:656, 11:658, *351*
__template_execute_if_arg_-agree:nnTF 11:42, 11:42, 11:207, 11:378
__template_execute_if_code_-exist:nnTF 11:52, 11:52, 11:647, 11:750, 11:933
__template_execute_if_keys_-exist:nnTF 11:71
__template_execute_if_keytype_-exist:nTF 11:58, 11:58, 11:64, 11:224
__template_execute_if_type_-exist:nTF 11:65, 11:65, 11:205, 11:376
__template_find_global: 11:725, 11:735, 11:735
__template_find_global_aux:w 11:735, 11:740, 11:743
\l__template_global_bool 11:22, 11:737, 11:746, 11:761, 11:810, 11:827, 11:849, *351*
__template_if_instance_exist:nn 11:84
__template_if_instance_exist:nnTF ... 11:84, 11:675, 11:688, 11:904, 11:950, 11:1187, 11:1189, 11:1191
__template_if_key_value:n 11:77, 11:83
__template_if_key_value:nTF 11:77, 11:767, 11:855
__template_if_keys_exist:nnTF .. . 11:71, 11:380, 11:600, 11:915, 11:924
__template_if_use_template:n . 11:90
__template_if_use_template:nTF 11:90, 11:892
__template_implement_choice_-elt:n 11:528, 11:561, 11:593
__template_implement_choice_-elt:nnn 11:529, 11:561, 11:561
__template_implement_choice_-elt_aux:n 11:561, 11:567, 11:575, 11:578
__template_implement_choice_-elt_aux:nnn 11:561, 11:566, 11:573, 11:585
__template_implement_choices:nn 11:450, 11:522, 11:522
__template_implement_choices_-default: ... 11:522, 11:533, 11:540
__template_instance_set_eq:nnn 11:673, 11:673, 11:1171
\c__template_instances_root_tl 11:11, 11:86, 11:663, 11:665, 11:679, 11:681, 11:682, 11:905, *351*
\l__template_key_name_tl .. 11:23, 11:231, 11:234, 11:261, 11:263, 11:284, 11:299, 11:306, 11:311, 11:353, 11:355, 11:360, 11:416, 11:419, 11:424, 11:468, 11:482, 11:497, 11:506, 11:525, 11:526, 11:531, 11:543, 11:547, 11:550, 11:553, 11:556, 11:557, 11:580, 11:583, 11:588, 11:590, 11:596, 11:620, 11:622, 11:625, 11:727, 11:788, 11:791, 11:793, 11:797, *352*
\c__template_key_order_root_tl 11:13, 11:115, 11:116, 11:155, 11:158, *351*
\l__template_key_order_seq 11:29, 11:117, 11:157, 11:160, 11:211, 11:230, 11:263, 11:707, *352*
__template_key_to_value: 11:769, 11:857, 11:872, 11:872
__template_key_to_value_auxi:w 11:872, 11:873, 11:874
__template_key_to_value_auxii:w 11:872, 11:880, 11:885
\l__template_keytype_arg_tl 11:25, 11:245, 11:258, 11:259, 11:266, 11:323, 11:337, 11:463, 11:524, 11:821, 11:838, *352*
\l__template_keytype_tl 11:24, 11:224, 11:243, 11:257, 11:264, 11:273, 11:322, 11:333, 11:420, 11:422, 11:448, 11:510, 11:630, 11:722, 11:724, 11:728, *352*
\c__template_keytypes_arg_seq 11:16, 11:226, 11:346, *351*
\l__template_keytypes_prop 11:28, 11:114, 11:151, 11:154, 11:210, 11:261, 11:409,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

11:418, 11:424, 11:550, 11:580, 11:622, 11:712, 11:793, 11:927, *352*
\c__template_keytypes_root_tl 11:12, 11:73, 11:107, 11:110, 11:112, 11:113, 11:149, 11:152, *351*
__template_map_var_type: 11:489, 11:492, 11:508, 11:508, 11:847
__template_parse_keys_elt:n 11:213, 11:219, 11:219, 11:272, *358*
__template_parse_keys_elt:nn 11:213, 11:270, 11:270
__template_parse_keys_elt_aux: 11:219, 11:236, 11:253
__template_parse_keys_elt_aux:n 11:219, 11:227, 11:241
__template_parse_values:nnn . . . 11:603, 11:607, 11:607, 11:657, 11:754
__template_parse_values_elt:n 11:611, 11:613, 11:613
__template_parse_values_elt:nn 11:611, 11:618, 11:618
__template_parse_values_elt_aux:n 11:618, 11:624, 11:627
__template_parse_vars_elt:n 11:407, 11:412, 11:412
__template_parse_vars_elt:nnn 11:407, 11:414, 11:414
__template_parse_vars_elt_aux:nn 11:414, 11:423, 11:428
__template_parse_vars_elt_aux:nnn 11:414, 11:436, 11:440, 11:446, 11:502
__template_parse_vars_elt_aux:nw 11:414, 11:430, 11:432
__template_parse_vars_elt_key:nn 11:414, 11:455, 11:472, 11:490, 11:503
__template_quark_if_nil:N . . . 11:41
__template_quark_if_nil:NTF 11:887
__template_quark_if_nil:nTF . 11:41
__template_quark_if_nil_p:n . 11:41
__template_recover_defaults:nn 11:136, 11:136, 11:403, 11:602, 11:634, 11:649, 11:752, 11:917
__template_recover_keytypes:nn 11:136, 11:146, 11:404, 11:609, 11:636, 11:926
__template_recover_values:nn . . . 11:136, 11:162, 11:677, 11:690, 11:952
__template_recover_vars:nn 11:136, 11:172, 11:638, 11:650, 11:700, 11:753, 11:935
\c__template_restrict_root_tl . . *351*
__template_show:Nnnn 11:913, 11:918, 11:927, 11:936, 11:940
__template_show_code:nn 11:911, 11:911, 11:1177
__template_show_defaults:nn 11:913, 11:913, 11:1179
__template_show_keytypes:nn 11:913, 11:922, 11:1181
__template_show_values:nn 11:948, 11:948, 11:1185
__template_show_vars:nn 11:913, 11:931, 11:1183
__template_split_keytype:n 11:221, 11:275, 11:275
__template_split_keytype_arg:n 11:316, 11:320, 11:320, 11:349, 11:422, 11:552, 11:582, 11:629, 11:721, 11:795
__template_split_keytype_arg_aux:n . 11:320, 11:324, 11:347, 11:350
__template_split_keytype_arg_aux:w . 11:320, 11:328, 11:343, 11:351
__template_split_keytype_aux:w 11:275, 11:285, 11:296, 11:308
__template_store_defaults:nn 11:96, 11:96, 11:214, 11:604, 11:635
__template_store_key_implementation:nnn 11:382, 11:401, 11:401
__template_store_keytypes:nn 11:96, 11:104, 11:215, 11:637
__template_store_value:n 11:354, 11:354, 11:356, 11:357, 11:358
__template_store_value_aux:Nn 11:359, 11:359, 11:362, 11:364, 11:366, 11:368, 11:370, 11:372
__template_store_value_boolean:n 11:352, 11:352
__template_store_value_choice:n 11:354, 11:356
__template_store_value_commalist:n 11:359, 11:373
__template_store_value_function:n 11:354, 11:357
__template_store_value_instance:n 11:354, 11:358
__template_store_value_integer:n 11:359, 11:361
__template_store_value_length:n 11:359, 11:363
__template_store_value_muskip:n 11:359, 11:365
__template_store_value_real:n 11:359, 11:367

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

__template_store_value_skip:n 11:359, 11:369
__template_store_value_tokenlist:n 11:359, 11:371, 11:373
__template_store_values:nn 11:96, 11:120, 11:661, 11:678
__template_store_vars:nn 11:96, 11:128, 11:408, 11:639
__template_template_set_eq:nnn 11:632, 11:632, 11:1163
\l__template_tmp_clist 11:32, 11:524, 11:534, 11:536, 11:563, 11:570, 11:572, 11:1195, *352*
\l__template_tmp_dim 11:33, *352*
\l__template_tmp_int 11:34, 11:190, 11:191, 11:194, 11:196, 11:200, *352*
\l__template_tmp_muskip . . 11:35, *352*
\l__template_tmp_skip . . . 11:36, *352*
\l__template_tmp_tl 11:37, 11:44, 11:45, 11:49, 11:255, 11:262, 11:278, 11:279, 11:280, 11:286, 11:532, 11:542, 11:543, 11:544, 11:546, 11:547, 11:548, 11:551, 11:552, 11:554, 11:581, 11:582, 11:589, 11:591, 11:623, 11:629, 11:692, 11:694, 11:712, 11:713, 11:794, 11:795, 11:804, 11:805, 11:876, 11:877, 11:883, *352*
\g__template_type_prop 11:18, 11:44, 11:67, 11:184, 11:195, *351*
__template_use_instance:nn 11:837, 11:890, 11:890, 11:1175
__template_use_instance_aux:nn 11:890, 11:894, 11:902
__template_use_instance_-aux:nNnnn . . 11:890, 11:893, 11:896
__template_use_template:nnn 11:748, 11:748, 11:899, 11:1167
\l__template_value_tl 11:26, 11:717, 11:767, 11:774, 11:780, 11:788, 11:798, 11:822, 11:839, 11:855, 11:861, 11:868, 11:873, 11:878, 11:880, 11:888, *352*
\l__template_values_prop 11:30, 11:101, 11:125, 11:141, 11:144, 11:167, 11:170, 11:209, 11:353, 11:355, 11:360, 11:531, 11:553, 11:660, 11:691, 11:717, 11:918, 11:957, *361*
\c__template_values_root_tl 11:14, 11:123, 11:124, 11:165, 11:168, *351*
\l__template_var_tl 11:27, 11:719, 11:738, 11:740, 11:745, 11:773, 11:781, 11:819, 11:835, 11:860, 11:867, *352*
\l__template_vars_prop 11:31, 11:133, 11:177, 11:180, 11:405, 11:467, 11:481, 11:496, 11:525, 11:544, 11:548, 11:591, 11:719, 11:804, 11:877, 11:936, *352*
\c__template_vars_root_tl . 11:15, 11:131, 11:132, 11:175, 11:178, *351*
\tencirc 31:10, 42:125, 42:677
\tencircw 31:10, 42:128
\tenln . 31:9, 42:124, 42:126, 42:676, 42:678
\tenlnw 31:9, 42:127, 42:129
\test *1204*
\testallgroups 33:1250, 33:1423, 33:1424, 33:1425, 33:1427, 33:1428, 33:1429, 33:1430, 33:1431, 33:1432, 33:1433, 33:1434, 33:1435, 33:1436, 33:1437, 33:1438, 33:1439, 33:1440, 33:1441, 33:1442, 33:1443, 33:1444, 33:1445, 33:1449, 33:1450, 33:1451, 33:1452, 33:1453, 33:1454, 33:1458, 33:1459, 33:1460, 33:1464, 33:1465, 33:1466, 33:1467, 33:1468, 33:1469, 33:1470, 33:1471, 33:1472, 33:1473, 33:1474, 33:1475, 33:1476, 33:1477, 33:1478, 33:1479, 33:1480, 33:1481, 33:1482, 33:1483, 33:1484, 33:1486, 33:1490, 33:1491, 33:1492, 33:1493, 33:1494, 33:1495, 33:1496, 33:1497, 33:1498, 33:1500, 33:1501, 33:1502, 33:1503, 33:1504, 33:1505, 33:1506, 33:1507, 33:1508, 33:1509, 33:1510, 33:1511, 33:1512, 33:1513, 33:1514, 33:1515, 33:1516, 33:1517, 33:1518, 33:1519, 33:1520, 33:1521, 33:1522, 33:1523, 33:1524, 33:1525, 33:1526, 33:1527, 33:1528, 33:1529, 33:1530, 33:1531, 33:1532, 33:1533, 33:1534, 33:1535, 33:1536, 33:1537, 33:1538, 33:1539, 33:1540, 33:1541, 33:1542, 33:1543, 33:1544, 33:1545, 33:1546, 33:1547, 33:1548, 33:1549, 33:1550, 33:1551, 33:1552, 33:1553, 33:1554, 33:1555, 33:1556, 33:1557, 33:1558, 33:1559, 33:1560, 33:1561, 33:1562, 33:1563, 33:1564, 33:1565, 33:1566, 33:1567, 33:1568, 33:1569, 33:1570, 33:1571, 33:1572, 33:1573, 33:1574, 33:1575, 33:1576, 33:1577, 33:1578, 33:1579, 33:1580, 33:1581, 33:1582, 33:1583, 33:1584, 33:1585, 33:1586, 33:1587, 33:1588, 33:1589, 33:1590, 33:1591, 33:1592,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

33:1593, 33:1594, 33:1595, 33:1596, 33:1597, 33:1598, 33:1599, 33:1600, 33:1601, 33:1602, 33:1603, 33:1604, 33:1605, 33:1606, 33:1607, 33:1608, 33:1609, 33:1610, 33:1611, 33:1612, 33:1613, 33:1614, 33:1615, 33:1634
\testallkerneldefinedfamilies 33:1422, 33:1631
\testFont 33:1218, 33:1219, 33:1251, 33:1252
\testgroup . 33:1223, 33:1261, 33:1270, 33:1276, 33:1282, 33:1286, 33:1290, 33:1300, 33:1304, 33:1372, 33:1376
\TeX 19:1, 19:12, *1249*
TEX and LATEX 2ε commands:
\...-h@@k *1249*
\..@without@substitution *577*
\@ 09:375, 14:19, 04:20, 18:500, 19:2, 04:1026, 50:43, 50:58, 50:71, 50:80, 50:118, 06:833, 06:834, 57:512, 01:49, *1199*
\@...hook *229*
\@?@? *1199*
\@@ . 13:15, 13:19, 13:20, 13:21, 13:22, 13:24, 13:27, 13:28, 13:30, 13:31, 20:753, 20:773, 26:510, 26:512, 26:513, 41:238, 41:239, 41:240, 41:250, 01:315, 01:316, 54:10, 54:11
\@@DeclareMathDelimiter *1210*
\@@defaultsubs 24:610
\@@enc@update 21:183, 24:259, 24:263, *1222*
\@@end 20:722, 20:723, 01:206, 37:33, 37:98, 37:174, 37:240, 06:23, 56:18, 57:667, 57:688, 01:53, *462*
\@@endpbox 41:193, 41:236, 41:386
\@@eqncr 38:381, 38:403, 38:413, 38:418, 38:541
\@@fileswith@pti@ns 50:605, 50:624, 50:1104
\@@hyph 06:24, 06:850
\@@hyphenation 21:205
\@@if@newlist 54:603, 54:648, 54:661, 54:707
\@@ifdefinable 21:17, 06:132
\@@input 20:620, 20:633, 20:673, 37:27, 37:92, 37:154, 06:22, 50:1727, 52:172, 52:188, 52:196, 57:331, 01:52, *1204*
\@@italiccorr 32:113, 32:117, 06:25, *1206*
\@@line 40:614
\@@math@bgroup 32:131, 32:138
\@@math@egroup 32:128
\@@par . 15:4, 16:140, 16:177, 37:240, 37:531, 37:536, 37:539, 37:553, 37:557, 37:560, 39:82, 39:85, 06:21, 40:334, 40:361, 40:385, 40:406, 41:199, 44:67, 44:118, 54:259, *1232*
\@@patterns 21:205
\@@protect 06:266, 06:272, 06:281
\@@sqrt *1231*
\@@startpbox . . . 41:193, 41:236, 41:386
\@@sverb 37:646, *781*
\@@text@case@aux 57:577, 57:608, 57:613, 57:618
\@@underline . . . 40:551, 40:554, 40:555
\@@unprocessedoptions 50:1016, 50:1081, 50:1179
\@@warning 14:215, *1202*
\@Alph 22:188, 22:204, *109*
\@DeclareEncodingSubset . . . 33:45, 33:47, 33:48, 33:49, 33:50, 33:53, 33:60
\@DeclareMathDelimiter 28:967, 28:986, *1210*
\@DeclareMathSizes 24:206, 24:207, 24:209
\@Esphack 18:191, 45:201, 45:223, 45:241, *1196*
\@Ialph *1212*
\@IncludeInRele@se 03:71
\@IncludeInRelease 03:71
\@LRmoderr 14:271, 18:239, 18:290, *1269*
\@M 18:11, 18:12, 18:13, 18:14, 18:15, 18:16, 18:17, 18:18, 18:19, 18:120, 24:679, 24:687, 26:439, 26:452, 38:391, 39:218, 06:37, 06:39, 41:56, 44:67, 44:100, 44:118, 44:130, 44:209, 44:232, 54:178, 54:197, 54:200, 54:260, 02:21, 02:508, 02:509
\@MM 45:469, 45:488, 45:506, 54:301, 02:21, *382*
\@Mi 12:3, 54:138
\@Mii 12:3, 45:53, 45:122, 45:194, 45:216, 45:241, 45:311, 54:297, 54:1163, 54:1330
\@Miii 12:3, 45:55, 45:124, 45:313, 54:300
\@Miv 12:3, 45:195, 45:201, 45:217, 45:223, 54:274
\@Roman 22:186, 22:192, *1235*
\@TeXversion 14:28, 01:310, *2*
\@abspage@last . . . 53:105, 53:111, 53:351, 53:353, 53:355, 53:363, 53:367, 53:370, 53:409, 53:410, *1080*
\@acci . . . 29:775, 40:387, 40:408, *1207*
\@accii . . 29:775, 40:387, 40:408, *1212*
\@acciii . 29:775, 40:387, 40:408, *1212*
\@acol 41:168, 41:178, 41:260, 41:261, 41:273, 41:274, 41:277, 41:294, 41:309, 41:317, 41:327

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@acolampacol 41:258, 41:275, 41:277, 41:284, 41:292, 41:326, 41:329
\@activechar@info 54:579, *1209*
\@activechar@warning *1209*
\@addamp 41:251, 41:260, 41:261, 41:276, 41:290, 41:327, 41:328
\@addfield 41:43, 41:53, 41:86, 41:93, 41:125, 41:140, 41:142
\@addmarginpar 54:333, 54:1815
\@addtobot 54:979, 54:1066, 54:1133, 54:1185, 54:1294, 54:1353
\@addtocurcol 54:330, 54:1070, 54:2003
\@addtodblcol 54:858, 54:1566
\@addtofilelist 20:64, 20:133, 20:188, 20:620, 20:742, 29:749, 29:752, 29:759, 29:762, 29:769, 29:772, 50:1726, 52:169, 52:188, 52:196, 57:279, 57:282, 57:708, 01:85, 01:87, *1206*
\@addtonextcol 54:857, 54:1390, 54:2004
\@addtopreamble .. 41:311, 41:324, 41:330, 41:331, 41:332, 41:334, 41:346
\@addtoreset 22:16, 22:44, 22:64, 22:77, 22:137, 22:163, 22:166
\@addtotoporbot 54:1016, 54:1179, 54:1347, 54:1439, 54:1528
\@afterheading ... 44:92, 44:125, *1233*
\@afterindentfalse 44:45
\@afterindenttrue 44:43, 44:124, 44:208, 44:231
\@alph 22:187, 22:200, 45:399, *109*
\@ampacol 41:258, 41:275, 41:286, 41:329
\@arabic 22:52, 22:70, 22:159, 22:171, 22:184, 22:190, 45:397
\@argarraycr 41:203, 41:204
\@argdef 06:80
\@argrsbox 40:575
\@argtabularcr 41:210, 41:211
\@array 41:181, 41:182
\@arrayacol 41:168, 41:258
\@arrayclassiv 41:169, 41:331
\@arrayclassz 41:168, 41:275
\@arraycr 41:170, 41:201, 41:203
\@arrayparboxrestore 40:377, 40:419, 41:384
\@arrayrule 41:309, 41:311, 41:315, 41:317, 41:319, 41:346
\@arstrut 41:192, 41:237, 41:343
\@arstrutbox 41:185, 41:218, 41:343, 41:385, *1210*
\@author 44:8, 44:32
\@auxout ... 20:217, 20:223, 20:267, 20:285, 20:309, 20:342, 20:371, 20:394, 20:418, 20:433, 35:105, 35:165, 36:55, 44:181, 47:7, 47:8, 47:39, 47:49, 47:57, 47:67, 47:84, 53:362
\@backslashchar .. 14:234, 14:236, 30:273, 06:231, 50:1308, 50:1441, 06:481, 50:1530, 06:630, 06:642, 06:646, 06:647, 06:652, 06:696, *1212*
\@badcrerr 14:279, *1210*
\@badend 14:247, 37:390, *1197*
\@badlinearg 14:267, 42:165, 42:177, 42:188, 42:189, 42:193, 42:242, 42:247, 42:257, 42:262, 42:275
\@badmath 14:251, 38:275, 38:277, 38:282, 38:285, 38:294, 38:306, 38:311, 38:320, 38:333, 38:338, 38:464, 38:476, 38:492, 38:501
\@badpoptabs 14:255, 41:85, 41:151
\@badrequireerror 50:497, 50:1187
\@badtab 14:258, 41:22, 41:87, 41:108, 41:114, 41:121, 41:148
\@begin@tempboxa 40:27, 40:46, 40:61, 40:246, 40:265, 40:334, 40:361, 40:576, 40:584, *1201*
\@begindocumenthook 20:61, 20:127, 20:130, 20:182, 20:185, 47:53, 50:1115, 50:1136, *216*
\@begindvi 54:627, 54:686, 54:714, *1066*
\@begindvibox 53:449, 53:450, 54:86, 54:715, *1061*
\@beginparpenalty .. 18:15, 38:467, 38:479, 38:505, 39:23, 39:194, *803*
\@begintheorem 43:43, 43:48
\@bezier 42:683, 42:682
\@bibitem 47:3, 47:8
\@biblabel 47:4, 47:97
\@bitor 54:15, 54:885, 54:905, 54:941, 54:964, 54:1031, 54:1115, 54:1125, 54:1273, 54:1284, 54:1426, 54:1513, 54:1631, 54:1756
\@botlist 54:65, 54:386, 54:388, 54:433, 54:435, 54:721, 54:742, 54:751, 54:752, 54:993, 54:996, 54:1031, 54:1125, 54:1284, 54:1959, 54:1987
\@botnum 45:274, 54:111, 54:990, 54:991, 54:996, 54:1000, 54:1462, 54:1467, 54:1555, 54:1562, 54:1951, 54:1979, 54:2021
\@botroom 45:275, 54:112, 54:993, 54:996, 54:1952, 54:1980
\@boxfpsbit . 54:2069, 54:2071, 54:2076
\@break@loop *1210*
\@break@tfor 13:31, 20:593, 20:610, 32:98, *1210*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@bsphack 18:37, 18:126, 18:387, 18:403, 18:421, 18:437, 35:102, 35:164, 36:82, 45:52, 45:121, 45:310, 46:6, 46:18, 46:23, 46:35, 47:63, 47:80, 54:1886, *749*
\@caption 45:12, 45:14
\@captype 45:5, 45:9, 45:12, 45:40, 45:88, 45:109, 45:157, 54:2033
\@car 19:14, 21:89, 21:110, 06:53
\@carcube 06:55, 06:135, 06:633
\@cclv 54:302, 54:306, 54:384, 54:385, 54:414, 54:431, 54:432, 54:461, 54:487, 54:491, 54:492, 02:16, 57:67, *1066*
\@cclvi 04:30, 04:58, 50:1306, 50:1439, 50:1528, 02:21, 02:57, 02:82, 02:93, 02:95, 02:99, 02:159, 02:173
\@cdr 20:251, 06:53, 06:776, 06:777
\@centercr . . 37:413, 37:451, 37:456, 37:461, 37:471, 37:475, 37:479, *1250*
\@centering 38:360, 38:361, 38:369, 38:372, 38:375, 38:534, 38:538
\@cflb 54:718
\@cflt 54:718
\@changed@cmd 21:3, 21:63, 21:223, 24:131, 24:267, 57:476, *1223*
\@changed@x . 21:3, 21:211, 21:219, *1222*
\@changed@x@err *1230*
\@changed@x@mouth 21:211, 21:219, *1222*
\@charlb 20:439, 20:447
\@charrb 20:441, 20:447
\@chclass 41:271, 41:272, 41:335, 41:348, 41:353
\@check@IncludeInRelease 03:71
\@check@c 06:189, 06:191
\@check@eq 06:195, 06:196, 06:200
\@checkcommand *1214*
\@checkend 37:16, 37:82, 37:146, 37:337, 37:352, 37:369, 37:379, 37:389, *1224*
\@chnum 41:279, 41:298, 41:335, 41:350, 41:351, 41:352
\@circ 42:621, 42:639, 42:648, 42:653, 42:656
\@circle 42:604, 42:605
\@circlefnt 42:125, 42:128, 42:447, 42:490, 42:519, 42:544, 42:614, 42:630, 42:662, 42:677
\@cite 47:36, 47:95
\@cite@ofmt 47:44, 47:96
\@citea 47:35, 47:37
\@citeb 47:38, 47:39, 47:40, 47:43, 47:44, 47:66, 47:67, 47:68, 47:69, 47:83, 47:84, 47:85, 47:86
\@citex . 47:20, 47:21, 47:29, 47:34, *938*
\@citex@checkblank 47:17, 47:18, 47:30
\@classi 41:271, 41:307
\@classii 41:271, 41:321
\@classiii 41:271, 41:326
\@classiv 41:169, 41:180, 41:272
\@classoptionslist 50:9, 50:546, 50:561, 50:562, 50:579, 50:810, 50:811, 50:839, 50:840, 50:866, 50:867, 50:1808, *1257*
\@classv 41:272, 41:332
\@classz 41:168, 41:179, 41:271
\@cline 41:367
\@clnht 42:195, 42:196, 42:204, 42:206, 42:208, 42:218, 42:225, 42:273, 42:671
\@clnwd 42:197, 42:203, 42:207, 42:209, 42:210, 42:671
\@cls@pkg 08:2673, 50:353, 50:354, 50:367, 50:368, 50:955, 50:965, 50:1013, 50:1060, 50:1090, 50:1142, 50:1172, 50:1174, 50:1191, 50:1736, 50:1758, 50:1788
\@clsextension 50:31, 50:156, 50:164, 50:222, 50:383, 50:399, 50:411, 50:493, 50:517, 50:528, 50:546, 50:560, 50:578, 50:677, 50:692, 50:716, 50:809, 50:838, 50:865, 50:959, 50:1006, 50:1033, 50:1094, 50:1107, 50:1147, 50:1162, 50:1613, 51:99, 51:165, *1046*
\@clubpenalty . . 20:7, 20:25, 20:95, 20:152, 39:128, 39:220, 44:106, 44:135
\@colht 20:22, 20:92, 20:149, 45:273, 45:275, 45:278, 45:284, 45:285, 45:298, 45:299, 54:116, 54:233, 54:244, 54:253, 54:254, 54:389, 54:401, 54:436, 54:449, 54:478, 54:509, 54:539, 54:545, 54:549, 54:559, 54:564, 54:649, 54:708, 54:781, 54:819, 54:863, 54:888, 54:907, 54:947, 54:969, 54:1646, 54:1772, 54:2134, 57:155
\@colnum 45:276, 54:113, 54:999, 54:1044, 54:1113, 54:1114, 54:1142, 54:1150, 54:1271, 54:1272, 54:1304, 54:1316, 54:1424, 54:1425, 54:1462, 54:1467, 54:1511, 54:1512, 54:1554, 54:1561, 54:1947, 54:1975, 54:2014, 54:2189
\@colroom 20:23, 20:93, 20:150, 54:117, 54:254, 54:275, 54:276, 54:287, 54:290, 54:389, 54:436, 54:781, 54:998, 54:1043, 54:1109, 54:1112, 54:1141, 54:1266,

54:1270, 54:1303, 54:1420, 54:1423, 54:1506, 54:1510, 54:1948, 54:1976, 54:2144, 54:2149, 54:2194, 57:154, *1231*
\@combinedblfloats 54:754, 54:2268, 54:2307
\@combinefloats 54:505, 54:718
\@comdblflelt 54:754
\@comflelt 54:724, 54:740, 54:754
\@compatibility *1204*
\@cons 22:81, 22:90, 06:52, 45:193, 45:215, 45:239, 45:379, 54:239, 54:892, 54:911, 54:927, 54:951, 54:953, 54:973, 54:975, 54:1145, 54:1213, 54:1309, 54:1382, 54:1455, 54:1545, 54:1648, 54:1671, 54:1774, 54:1799, 54:1816, 54:1817, 54:2195, 02:196, 02:214
\@contentsline@destination 44:188, 44:190, 44:197, *908*
\@contfield 41:50, 41:141, 41:153
\@copy@... . . . . *100*
\@copy@DeclareRobustCommand 06:497, 06:567, 06:588, 06:666, 06:669
\@copy@newcommand 06:318, 06:498, 06:567, 06:609, 06:639, 06:666, 06:672, *93*
\@copytexsys *1204*
\@ctrerr 14:243, 22:203, 22:207, 22:221, 22:229
\@curfield 41:16, 41:41, 41:47, 41:51, 41:52, 41:54, 41:130, 41:131
\@curline 41:16, 41:27, 41:39, 41:44, 41:53, 41:54, 41:55, 41:90, 41:91, 41:103, 41:128, 41:129
\@curr@enc 21:154, 21:156
\@curr@file 20:227, 20:228, 20:237, 20:239, 20:263, 20:271, 20:452, 20:471, 20:640, 20:655, 50:940, 50:1253, 50:1258, 50:1264, 50:1270, 50:1274, 50:1285, 50:1294, 50:1321, 50:1384, 50:1389, 50:1395, 50:1418, 50:1427, 50:1453, 52:266, 52:374, 52:376, *1049*
\@curr@file@reqd 52:266, 52:376, 52:380, *1049*
\@currbox 45:60, 45:91, 45:95, 45:129, 45:160, 45:164, 45:193, 45:214, 45:215, 45:239, 45:257, 45:259, 45:261, 45:319, 45:322, 45:327, 45:331, 54:215, 54:216, 54:227, 54:228, 54:230, 54:231, 54:239, 54:313, 54:314, 54:857, 54:858, 54:1106, 54:1108, 54:1116, 54:1139, 54:1143, 54:1145, 54:1160, 54:1201, 54:1213, 54:1261, 54:1264, 54:1301, 54:1306, 54:1309, 54:1326, 54:1371, 54:1382, 54:1414, 54:1430, 54:1444, 54:1455, 54:1497, 54:1534, 54:1545, 54:1585, 54:1589, 54:1600, 54:1606, 54:1608, 54:1612, 54:1617, 54:1626, 54:1635, 54:1641, 54:1648, 54:1671, 54:1706, 54:1710, 54:1722, 54:1729, 54:1731, 54:1735, 54:1741, 54:1751, 54:1766, 54:1774, 54:1799, 54:1817, 54:1826, 54:2039, 54:2040, 54:2069, 54:2099, 54:2104, 54:2150, 54:2153, 54:2165, 54:2173, 54:2190, 54:2195, 02:275, 02:276, 02:277
\@currdir 01:92, 01:114, 01:116, 01:122, 01:124, 01:130, 01:132, 01:137, 01:139, 01:149, 01:162, 01:227, 01:240, 01:253, 50:1232, 50:1258, 50:1367, 50:1389, 50:1418, 50:1501, *9*
\@current@cmd 21:25, 24:271, *1222*
\@currentHpage . . . 36:240, 36:241, *751*
\@currentHref 17:34, 17:38, 35:99, 35:107, 35:116, 35:120, 35:140, 36:239, 55:53, 55:57, *751*
\@currentcounter . . 35:113, 35:125, 35:135, 35:142, 35:171, 35:173, 35:183, 36:242, 38:365, 38:519, 40:485, 43:53, 43:55, 45:471, *1260*
\@currentlabel . . . 35:106, 35:126, 35:136, 35:143, 35:166, 35:174, 35:185, 35:196, 35:204, 36:236, 38:364, 38:518, 40:486, 40:503, 40:520, 45:472, 45:490, 45:508, *795*
\@currentlabelname 35:98, 35:107, 36:238, *751*
\@currenttarget 35:162
\@currenttitle 35:161
\@currenvir 14:249, 37:3, 37:251, 37:272, 37:290, 37:300, 37:390, 39:112, 40:188, 50:1296, 50:1308, 50:1316, 50:1320, 50:1327, 50:1429, 50:1441, 50:1449, 50:1453, 50:1459, 50:1518, 50:1530, 50:1538, 50:1542, 50:1548, *407*
\@currenvline 14:249, 37:252, 37:273, 37:291, 37:301, 37:391, 40:189
\@currext 50:30, 50:42, 50:57, 50:70, 50:79, 50:117, 50:379, 50:382, 50:383, 50:398, 50:399, 50:410, 50:411, 50:517, 50:528, 50:539, 50:546, 50:560, 50:578, 50:607, 50:687, 50:700, 50:702, 50:710, 50:724, 50:908, 50:909, 50:911,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

50:915, 50:918, 50:920, 50:925, 50:930, 50:932, 50:937, 50:943, 50:951, 50:957, 50:959, 50:963, 50:968, 50:974, 50:983, 50:986, 50:991, 50:994, 50:997, 50:999, 50:1000, 50:1002, 50:1006, 50:1015, 50:1017, 50:1018, 50:1023, 50:1026, 50:1029, 50:1033, 50:1039, 50:1052, 50:1057, 50:1058, 50:1063, 50:1069, 50:1073, 50:1075, 50:1076, 50:1078, 50:1080, 50:1082, 50:1083, 50:1086, 50:1092, 50:1094, 50:1107, 50:1120, 50:1147, 50:1150, 50:1162, 50:1180, 50:1181, 51:62, 51:99, 51:106, 51:108, 51:165, 51:167, 51:170, *1002*

`\@currextension` *1198*

`\@currlist` 45:193, 45:215, 45:379, 54:67, 54:313, 54:390, 54:393, 54:437, 54:440, 54:1816

`\@currname` 03:71, 03:116, 03:124, 20:762, 20:784, 20:790, 50:29, 50:41, 50:56, 50:69, 50:78, 50:116, 50:351, 50:353, 50:365, 50:367, 50:379, 50:382, 50:398, 50:410, 50:539, 50:607, 50:700, 50:702, 50:710, 50:724, 50:906, 50:909, 50:911, 50:915, 50:918, 50:920, 50:925, 50:927, 50:930, 50:932, 50:937, 50:942, 50:951, 50:955, 50:957, 50:963, 50:965, 50:966, 50:968, 50:974, 50:983, 50:985, 50:991, 50:994, 50:997, 50:999, 50:1000, 50:1004, 50:1008, 50:1015, 50:1017, 50:1018, 50:1023, 50:1026, 50:1030, 50:1034, 50:1039, 50:1051, 50:1075, 50:1076, 50:1078, 50:1080, 50:1082, 50:1083, 50:1120, 50:1172, 50:1174, 50:1181, 50:1191, 50:1736, 50:1758, 50:1788, 51:62, 51:75, 51:106, 51:108, 51:167, 51:170, 51:184, 51:186, 51:197, 51:255, 08:368, 08:374, 08:429, *1002*

`\@currnamestack` 50:33, 50:136, 52:546, 08:426, *982*

`\@curroption` *1198*

`\@curroptions` 50:539, 50:547, 50:597, 50:616, 50:1181, 50:1182

`\@currpath` 50:15, 50:46, 50:139, 50:351, 50:353, 50:907, 50:925, 50:932, 50:941, 50:983, 50:984, 50:1011, *980*

`\@currpkg@reqd` 50:381, 50:935, 50:937, 50:946, 50:979, 50:993, 50:996, 50:1011, 50:1013, *1003*

`\@currsize` 29:639, *1197*

`\@currtype` 54:121, 54:882, 54:883, 54:884, 54:885, 54:902, 54:903, 54:904, 54:905, 54:1031, 54:1115, 54:1125, 54:1273, 54:1284, 54:1426, 54:1513, 54:1631, 54:1756, 54:2039, 54:2041, 54:2042, 54:2045

`\@curtab` 41:11, 41:26, 41:86, 41:87, 41:88, 41:94, 41:95, 41:98, 41:102, 41:103, 41:107, 41:146, 41:147, *1229*

`\@curtabmar` 41:11, 41:25, 41:26, 41:38, 41:44, 41:89, 41:102, 41:106, 41:107

`\@d@r` 01:145, 01:146

`\@dashbox` 42:324, 42:325, 42:326, 42:327, 42:328, 42:331, 42:337, 42:339, 42:349, 42:351, 42:352, 42:353, 42:354, 42:358, 42:362, 42:365, 42:382, 42:384, 42:385, 42:386, 42:387, 42:390, 42:394, 42:396, 42:405, 42:407, 42:408, 42:409, 42:410, 42:413, 42:417, 42:422, 42:673

`\@dashcnt` 42:317, 42:319, 42:320, 42:321, 42:322, 42:323, 42:336, 42:338, 42:341, 42:343, 42:344, 42:345, 42:347, 42:348, 42:361, 42:364, 42:376, 42:377, 42:378, 42:379, 42:380, 42:381, 42:393, 42:395, 42:398, 42:399, 42:400, 42:401, 42:403, 42:404, 42:416, 42:421, 42:673

`\@dashdim` 42:316, 42:317, 42:318, 42:319, 42:320, 42:322, 42:325, 42:327, 42:328, 42:329, 42:336, 42:338, 42:340, 42:341, 42:342, 42:343, 42:344, 42:347, 42:351, 42:353, 42:354, 42:355, 42:363, 42:366, 42:375, 42:376, 42:377, 42:378, 42:380, 42:384, 42:386, 42:387, 42:388, 42:393, 42:395, 42:397, 42:398, 42:399, 42:400, 42:403, 42:407, 42:409, 42:410, 42:411, 42:419, 42:424, 42:673

`\@date` 44:9, 44:33

`\@dbflt` 45:32, 45:264, *1218*

`\@dblarg` . . 44:54, 44:142, 45:12, 06:798

`\@dbldeferlist` 45:239, 54:70, 54:447, 54:452, 54:454, 54:820, 54:827, 54:828, 54:1756, 54:1799, 54:1963, 54:1992, *1243*

`\@dblfloat` 45:31

`\@dblfloatplacement` 20:31, 20:101, 20:159, 45:280, 54:403, 54:451, 54:1944, 54:1972, 54:2273, 54:2313

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

\@dblflset 45:26
\@dblfpbot 45:290, 45:304, 54:2357
\@dblfpsep 45:289, 45:303, 54:2357
\@dblfptop 45:288, 45:302, 54:2357
\@dbltoplist 54:69, 54:234, 54:237, 54:239, 54:399, 54:400, 54:447, 54:448, 54:759, 54:763, 54:765, 54:766, 54:1643, 54:1648, 54:1768, 54:1774, 54:1962, 54:1990
\@dbltopnum 45:283, 45:297, 54:109, 54:129, 54:240, 54:242, 54:770, 54:1582, 54:1583, 54:1647, 54:1650, 54:1678, 54:1683, 54:1703, 54:1704, 54:1773, 54:1777, 54:1806, 54:1811, 54:1955, 54:1983
\@dbltoproom 45:284, 45:286, 45:298, 45:300, 54:110, 54:1585, 54:1588, 54:1589, 54:1598, 54:1599, 54:1602, 54:1605, 54:1608, 54:1612, 54:1616, 54:1620, 54:1625, 54:1645, 54:1706, 54:1709, 54:1710, 54:1719, 54:1720, 54:1721, 54:1724, 54:1728, 54:1731, 54:1735, 54:1740, 54:1744, 54:1749, 54:1750, 54:1771, 54:1956, 54:1984
\@dec@text@cmd 21:3
\@declarecommandcopylisthook 06:494, 06:496, 06:508, 07:1127, *99*
\@declaredoptions 50:8, 50:500, 50:543, 50:581, 50:602, 50:621, 50:1113, *1198*
\@declareoption 50:498, 50:499, 50:507
\@defaultfamilyhook 29:429, 29:664, 29:681, 29:696, 29:701, 29:716, *1251*
\@defaultsubs 24:564, 24:598, 24:610, 37:57, 37:122, 37:161, *1227*
\@defaultunits 24:214, 24:218, 24:219, 24:220, 24:235, 24:328, 26:179, 26:181, 42:13
\@defaultunitsset . . . 40:74, 40:85, 42:8, 42:29, 42:30, 42:32, 42:34, 42:60, 42:63, 42:84, 42:85, 42:107, 42:108, 42:164, 42:241, 42:316, 42:318, 42:332, 42:340, 42:342, 42:357, 42:479, 42:480, 42:611, 42:647, 42:689, 42:690, 42:692, 42:693, 42:696, 42:697, 42:699, 42:700, 42:711, 42:712, 42:714, 42:715, 42:717, 42:718, 42:720, 42:721
\@defdefault@ds 50:498, 50:503, 50:508
\@deferlist 54:68, 54:386, 54:395, 54:396, 54:399, 54:404, 54:406, 54:412, 54:433, 54:442, 54:444, 54:782, 54:790, 54:791, 54:802, 54:807, 54:808, 54:1115, 54:1213, 54:1273, 54:1382, 54:1426, 54:1455, 54:1513, 54:1545, 54:1631, 54:1671, 54:1961, 54:1989
\@definecounter 22:12, 22:36, 38:349, 39:251, 39:252, 39:253, 39:254, 43:8, 43:16, 45:396, 45:398
\@depth 26:191, 30:565, 30:566, 30:568, 30:569, 06:26, 40:550, 40:603, 40:610, 41:187, 41:219, 42:227, 42:300, 42:303, 42:324, 42:333, 42:383, 42:391, 42:726, 42:782, 54:1855
\@dir 01:144, 01:147, 01:149, 01:151, 01:152
\@disable@packageload@do 50:930, 52:477
\@dischyph 40:386, 40:407, 06:835, 06:869, *1208*
\@doclearpage 54:298, 54:373
\@documentclass *1200*
\@documentclasshook 50:3, 50:815, 50:843, 50:870, *1205*
\@doendpe 37:338, 37:353, 37:370, 37:380, 39:123, 39:152, *811*
\@dofilelist 20:759, 20:801, 37:42, 37:107, 37:156, *1222*
\@dofilelist@hash 20:790, 20:796
\@dofilelist@size 20:784, 20:796
\@donoparitem 39:168, 39:182
\@dot 42:604, 42:642
\@dotsep 44:215, 44:238
\@dottedtocline 44:200, 44:226, 44:227
\@downline 42:297, 42:301, 42:306
\@downvector 42:268, 42:306
\@eha 03:195, 14:219, 14:237, 14:239, 14:241, 14:250, 14:252, 14:284, 20:224, 20:268, 20:286, 21:52, 21:84, 24:28, 24:58, 24:102, 24:144, 24:187, 24:253, 24:339, 26:106, 28:25, 28:70, 28:99, 28:172, 28:203, 28:233, 28:265, 28:478, 28:499, 28:551, 28:602, 28:647, 28:652, 28:707, 28:825, 28:829, 28:833, 28:868, 28:872, 28:876, 28:933, 28:943, 28:1028, 28:1033, 28:1036, 28:1068, 28:1071, 28:1144, 28:1147, 28:1150, 28:1217, 28:1223, 32:146, 33:24, 33:81, 33:912, 33:921, 37:250, 37:271, 37:289, 37:299, 47:71, 47:88, 06:292, 06:331, 06:359, 54:1880, 54:1896
\@ehb 14:219, 14:244, 14:270, 14:272, 14:274, 14:276, 54:236, 54:392, 54:439
\@ehc 03:184, 14:219, 14:279, 14:282, 14:288, 14:290, 37:655, 37:672, 37:686, 37:701,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

37:714, 38:422, 39:244, 06:128, 06:155, 44:31, 06:481, 50:1788, 06:522
\@ehd 03:102, 03:159, 14:219, 14:246, 14:254, 14:257, 14:259, 14:265, 28:118, 05:88, 41:100, 41:109, 45:6, 50:757, 50:1013
\@elt 20:440, 21:1576, 21:1578, 22:20, 22:35, 22:101, 22:104, 29:141, 29:152, 29:154, 29:155, 29:180, 29:505, 29:507, 29:508, 29:518, 29:742, 30:17, 30:25, 06:52, 54:8, 54:11, 54:15, 54:27, 54:30, 54:31, 54:32, 54:33, 54:38, 54:39, 54:40, 54:41, 54:42, 54:43, 54:44, 54:45, 54:47, 54:51, 54:57, 54:58, 54:59, 54:60, 54:502, 54:724, 54:735, 54:740, 54:750, 54:762, 54:764, 54:792, 54:809, 54:829, 54:848, 54:861, 54:868, 54:919, 54:922, 54:931, 54:1922, 54:1934, *651*
\@empty 13:14, *1197*
\@emptycol 54:200, 54:247, 54:250, 54:279, 54:283, *1209*
\@end@check@IncludeInRelease 03:143, 03:145
\@end@tempboxa 40:36, 40:55, 40:64, 40:258, 40:270, 40:352, 40:374, 40:582, 40:592
\@enddocument@kernel@warnings 37:43, 37:52, 37:108, 37:117, 37:176
\@enddocumenthook 37:145, 50:1115, 50:1137
\@endfloatbox 45:190, 45:211, 45:236, 45:248
\@endparenv 39:120, 39:123, *811*
\@endparpenalty . . . 18:16, 38:468, 38:480, 38:506, 39:23, 39:124, *803*
\@endpbox 41:193, 41:236, 41:266, 41:333, 41:384, 41:387
\@endpefalse 37:256, 37:276, 37:294, 37:304, 39:129, 39:133, 39:137, 39:138, 39:140, 40:138, 40:191, *1201*
\@endpetrue 39:124, 39:126, 39:136, 39:140, *813*
\@endpreamblehook *446*
\@endtheorem 43:13, 43:19, 43:25, 43:48
\@enlargepage 54:1865, 54:1870, 54:1872
\@ensuredmath 38:433, 38:435
\@enumctr 39:258, 39:261, 39:262
\@enumdepth 39:250, 39:256, 39:257, 39:258, *816*
\@enumspacing *816*
\@eqcnt 38:357, 38:419, 38:424, 38:521, 38:536, 38:537, 38:539
\@eqncr 38:370, 38:388, 38:425, 38:426, 38:523
\@eqnnum 38:351, 38:352, 38:423, 38:454, 38:513, *1196*
\@eqnsel 38:357, 38:535, *1198*
\@eqnswfalse 38:387
\@eqnswtrue 38:359, 38:366, 38:424, 38:520
\@eqpen 38:357, 38:391, 38:393, 38:404, 38:414
\@er@ext *1199*
\@err@ 14:37, 14:41, 14:44, 14:52, 14:64, 14:68, 14:71, 14:79
\@esphack . . . 18:39, 18:132, 18:392, 18:409, 18:426, 18:443, 35:109, 35:167, 36:86, 45:385, 46:17, 46:19, 46:34, 47:74, 47:90, 54:1888, *1196*
\@evenfoot . 49:12, 49:15, 54:617, 54:676
\@evenhead . 49:12, 49:15, 54:616, 54:675
\@execute@begin@hook 37:253, 37:257, 37:260, *769*
\@executeoption *1199*
\@expandtwoargs 03:188, 06:217, 50:224, 50:350, 50:545, 50:581, 50:635, 50:644
\@expast 41:239, 41:267
\@expl@@@filehook@clear@replacement@flag@@ 50:449, 50:472, 52:295, 52:318, 52:339, 52:519
\@expl@@@filehook@drop@extension@@N 52:292, 52:293, 52:315, 52:316, 52:336, 52:337, 52:521, *1050*
\@expl@@@filehook@file@pop@@ 50:948, 52:157, 52:525, 52:535
\@expl@@@filehook@file@pop@assign@@nnnn 52:162, 52:527, *1046*
\@expl@@@filehook@file@push@@ 50:931, 52:151, 52:523
\@expl@@@filehook@if@file@replaced@@TF . 52:289, 52:312, 52:333, 52:517, *1049*
\@expl@@@filehook@if@no@extension@@nTF 52:285, 52:308, 52:329, 52:507, 52:509, 52:534
\@expl@@@filehook@normalize@file@name@@w 52:291, 52:314, 52:335, 52:515
\@expl@@@filehook@resolve@file@subst@@w 50:447, 50:470, 52:288, 52:311, 52:332, 52:513
\@expl@@@filehook@set@curr@file@@nNN 50:446, 50:469, 50:900, 52:374, 52:380, 52:511, *1049*
\@expl@@@hook@curr@name@pop@@ 08:2880, 50:89
\@expl@@@hook@curr@name@push@@n *1255*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@expl@@@initialize@all@@ 08:2880, 37:261, 08:1469
\@expl@@@mark@update@dblcol@structures@@ 48:477, 48:490, 54:470
\@expl@@@mark@update@singlecol@structures@@ 48:475, 48:489, 54:473
\@expl@@@shipout@add@background@box@@n 53:417, 53:502
\@expl@@@shipout@add@background@picture@@n 53:417, 53:506
\@expl@@@shipout@add@firstpage@material@@N 53:417, 53:499
\@expl@@@shipout@add@foreground@box@@n 53:417, 53:504
\@expl@@@shipout@add@foreground@picture@@n 53:417, 53:508
\@expl@char@generate@@nn 05:145, 06:814, *1254*
\@expl@cs@⟨*thing*⟩@spec@@N *1253*
\@expl@cs@argument@spec@@N ... *1268*
\@expl@cs@parameter@spec@@N 05:140, 05:141, 05:142, 05:153, 06:657, 06:710, 06:721, 06:726, *1268*
\@expl@cs@prefix@spec@@N 05:138, 05:152, 06:655, 06:725
\@expl@cs@replacement@spec@@N 05:143, 05:154, 06:615, 06:650, 06:657, 06:711, 06:722, 06:726
\@expl@cs@to@str@@N 05:133, 05:136, 05:148, 05:150, 06:489, 06:581, 06:601, 06:603, 06:604, 06:617, 06:630, 06:642, 06:646, 06:647, 06:652, 06:696, 06:705, 06:709, 06:717, 06:719, *1253*
\@expl@finalise@setup@@ 05:30, 57:268, 57:269
\@expl@pop@filename@@ 05:26, 50:88, 50:92, 50:98, 50:109, *1253*
\@expl@push@filename@@ 05:24, 50:37, 50:39, 50:52, 50:54, 50:64, 50:76, 50:96, 50:102, *1253*
\@expl@push@filename@aux@@ 08:2778, 05:25, 50:37, 50:48, 50:60, 50:64, 50:82, 50:102, *298*
\@expl@str@if@eq@@nnTF ... 05:137, 05:151, 50:350, 06:453, 06:455, *1253*
\@expl@str@map@function@@NN 05:144, 05:155, 06:811, *109*
\@expl@str@map@function@@NN and \@expl@char@generate@@nn ... *1254*
\@expl@sys@load@backend@@ 20:17, 05:21, 05:23
\@expl@text@uppercase@@n *1262*
\@extra@page@added 37:68, 37:133, 53:387
\@failedlist 54:846, 54:869, 54:885, 54:892, 54:905, 54:911, 54:927, 54:941, 54:964
\@fcolmadefalse 54:837
\@fcolmadetrue 54:925
\@file-subst@⟨*file*⟩ *1047*
\@filef@und . 20:491, 20:503, 20:504, 20:507, 20:513, 20:545, 20:567, 20:590, 20:607, 20:620, 20:673, 52:155, 52:172, 52:188, 52:196, *1045*
\@filehook@file@push *1002*
\@filehook@set@CurrentFile 20:315, 20:376, 50:933, 52:152, 52:363
\@filelist .. 20:63, 20:132, 20:187, 20:741, 20:742, 20:761, 29:749, 29:759, 29:769, 57:279, 57:692, 57:708
\@fileswfalse 20:200, 50:1228, 50:1229, 50:1363, 50:1364
\@fileswith@pti@ns 50:497, 50:605, 50:624, 50:801, 50:802, 50:806, 50:808, 50:836, 50:837, 50:863, 50:864, 50:891, 50:1104, *1198*
\@fileswith@ptions 50:786, 50:787, 50:794, 50:795, 50:799, 50:803
\@fileswithoptions 50:677, 50:684, 50:692, 50:775
\@fileswtrue 20:5, 50:1211, 50:1214, 50:1271, 50:1275, 50:1346, 50:1349, 50:1404, 50:1408
\@finalstrut 40:490, 40:507, 40:524, 40:593, 41:385, 45:477, 45:495, 45:513, *1211*
\@firstampfalse 41:254, 41:277, 41:294
\@firstamptrue 41:262
\@firstcolfirstmark 54:2250, 54:2251, 54:2255
\@firstcoltopmark ... 54:2248, 54:2256
\@firstcolumnfalse .. 54:2240, 54:2285
\@firstcolumntrue ... 20:28, 20:98, 20:156, 54:100, 54:209, 54:2259, 54:2291
\@firstoffive 35:23, 35:25, 35:40, 35:57
\@firstofone 04:12, 20:126, 20:181, 04:100, 04:108, 20:459, 21:68, 21:153, 04:166, 26:346, 28:53, 28:81, 28:146, 28:184, 28:214, 28:245, 28:978, 05:34, 05:61, 05:76, 37:144, 38:431, 41:372, 45:10, 06:212, 47:38, 47:66, 47:83, 50:949, 50:978, 06:477, 06:482, 06:483, 06:486, 06:487, 06:518, 06:523, 06:524, 06:527, 06:528, 52:352, 52:369, 57:359, *1224*
\@firstoftwo 03:85, 17:17, 20:506, 20:512, 20:546, 20:591, 20:608, 21:133,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

22:239, 22:244, 28:982, 29:510, 33:69, 33:834, 33:884, 33:900, 35:42, 05:35, 35:78, 05:43, 06:212, 49:16, 50:161, 50:195, 50:207, 50:230, 50:248, 50:307, 50:337, 06:454, 06:456, 06:478, 50:1797, 06:519, 06:584, 06:635, 06:750, 06:760, 06:770, 06:797, 06:821, 01:71, *460*
\@firsttab 41:2, 41:74, 41:75, 41:76, 41:106, 41:118, *1210*
\@flcheckspace 54:993, 54:1029, 54:2140
\@flfail 54:869, 54:920, 54:941, 54:951, 54:964, 54:973
\@float 45:26, 45:32
\@floatboxreset 45:101, 45:170, 45:174
\@floatpenalty 45:3, 45:53, 45:55, 45:58, 45:122, 45:124, 45:127, 45:191, 45:194, 45:199, 45:201, 45:212, 45:216, 45:221, 45:223, 45:237, 45:241, 45:311, 45:313, 45:317, 45:321, 45:379
\@floatplacement . . . 20:31, 20:101, 20:159, 45:271, 54:151, 54:211, 54:255, 54:481, 54:1945, 54:1973, *1200*
\@flsetnum 54:990, 54:1026, 54:1113, 54:1271, 54:1424, 54:1511, 54:1582, 54:1703, 54:2108
\@flsettextmin 54:1089, 54:1241, 54:1410, 54:1493, 54:2124
\@flstop 54:2010
\@flsucceed 54:862, 54:870, 54:919, 54:953, 54:975
\@fltovf . 14:275, 45:93, 45:162, 45:322
\@flupdates . . 54:996, 54:1041, 54:2186
\@flushglue . 12:17, 37:452, 37:456, 37:462, 37:472, 37:475, 37:480, 37:530, 37:552, 39:76, 40:393, 40:414
\@fnsymbol 22:189, 22:208
\@font@aliasinfo 26:539
\@font@info 24:133, 24:171, 24:177, 24:391, 24:408, 24:646, 25:478, 26:30, 26:38, 26:46, 26:74, 26:87, 26:154, 26:200, 26:214, 26:225, 26:239, 26:255, 26:261, 26:274, 26:281, 26:288, 26:293, 26:303, 26:315, 26:327, 26:491, 26:503, 26:508, 26:515, 26:545, 26:558, 26:566, 28:279, 28:290, 28:299, 28:312, 28:329, 28:338, 28:353, 28:368, 28:426, 28:482, 28:581, 28:587, 28:631, 28:644, 28:727, 28:816, 28:859, 28:924, 28:1018, 28:1185, 28:1214, 33:55, 33:56, 33:99, 57:478
\@font@series@contextfalse 29:502, 29:538
\@font@series@contexttrue 29:521, 29:525, 29:537
\@font@shape@subst@warning 25:445, 25:448, 25:453, 25:511, 25:658, 25:661, *569*
\@font@warning 24:3, 24:560, 24:565, 24:592, 24:599, 25:456, 26:19, 26:33, 26:41, 26:49, 26:61, 26:77, 26:476, 26:490, 26:502, 26:507, 26:514, 26:557, 26:565, 27:30, 37:54, 37:119, 37:158, 57:300
\@fontenc@load@list 21:1577, 29:742, 30:17, 30:25, *1251*
\@fontswitch 32:126, 32:128, *1210*
\@footnotemark 45:454, 45:460, 45:522, 45:528, 45:529, 45:555
\@footnotetext 40:455, 45:454, 45:460, 45:461, 45:538, 45:544
\@for 13:16, 20:233, 20:305, 20:367, 20:414, 20:761, 47:36, 47:65, 47:82, 50:232, 50:249, 50:543, 50:562, 50:579, 50:597, 50:602, 50:616, 50:621, 50:657, 50:667, 50:1182, 50:1219, 50:1354, *938*
\@forced@seriesfalse 25:394, 25:407, 26:140
\@forced@seriestrue . . 25:398, 25:409
\@forloop 13:19, 13:20
\@fornoop 13:15, 13:23, 13:29
\@fortmp 13:17, 13:18, 13:26, 50:655, 50:657, 50:1218, 50:1219, 50:1353, 50:1354
\@fpbot 45:290, 45:304, 54:867, 54:2351
\@fpmin 45:278, 45:287, 45:301, 54:115, 54:924, 54:1953, 54:1981, 54:2203, 54:2220
\@fps 45:41, 45:42, 45:44, 45:47, 45:64, 45:110, 45:111, 45:113, 45:116, 45:133, 54:2031, 54:2033, 54:2036
\@fpsadddefault 45:45, 45:48, 45:114, 45:117, 54:2028
\@fpsep 45:289, 45:303, 54:865, 54:874, 54:946, 54:968, 54:2351
\@fpstype 54:987, 54:1008, 54:1009, 54:1023, 54:1054, 54:1055, 54:1079, 54:1081, 54:1084, 54:1086, 54:1137, 54:1193, 54:1194, 54:1229, 54:1232, 54:1235, 54:1238, 54:1299, 54:1361, 54:1362, 54:1400, 54:1402, 54:1405, 54:1407, 54:1481, 54:1484, 54:1487, 54:1490, 54:1579, 54:1594, 54:1596, 54:1614, 54:1623,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

54:1659, 54:1660, 54:1700, 54:1715, 54:1717, 54:1737, 54:1747, 54:1786, 54:1787, 54:2024, 54:2040, 54:2042, 54:2044, 54:2047, 54:2048, 54:2049, 54:2051, 54:2052, 54:2056, 54:2057, 54:2059, 54:2060, 54:2094, 54:2096, 54:2098, 54:2110, 54:2112, 54:2126, 54:2128, 54:2158, 54:2161, 54:2172
\@fptop 45:288, 45:302, 54:864, 54:2351
\@frameb@x 40:218, 40:257, 40:269, 40:273, *1217*
\@framebox 40:225, 40:232, 40:236
\@framepicbox . . 40:225, 40:232, 40:294
\@freelist 45:60, 45:129, 45:319, 45:320, 54:29, 54:34, 54:48, 54:56, 54:215, 54:503, 54:736, 54:751, 54:765, 54:870, 54:1816, 54:1817, 02:196, 02:214, 02:275, *1244*
\@generic@error *1215*
\@generic@message *1215*
\@getcirc 42:437, 42:484, 42:513, 42:540, 42:612, 42:628
\@getfpsbit 54:984, 54:1020, 54:1576, 54:1697, 54:2067
\@getlarrow 42:266, 42:274, 42:276
\@getlinechar 42:190, 42:229
\@getpen . . . 18:35, 18:38, 18:47, 18:118
\@getrarrow 42:267, 42:274, 42:283
\@glossaryfile . . . 46:21, 46:22, 46:31
\@gnewline 18:96, 18:102, 18:109, 18:112
\@gobble 03:163, 13:6, 13:9, 03:187, 14:101, 14:127, 14:147, 14:155, 14:180, 14:189, 14:202, 17:15, 18:76, 04:11, 18:559, 20:64, 20:133, 20:188, 04:98, 20:466, 20:468, 20:741, 21:29, 24:561, 24:594, 26:345, 27:26, 28:28, 28:30, 28:430, 28:441, 28:525, 28:592, 28:593, 28:622, 28:628, 28:636, 28:641, 28:659, 28:673, 28:683, 28:692, 28:705, 28:722, 28:731, 28:805, 28:807, 28:811, 28:819, 28:853, 28:862, 28:914, 28:916, 28:927, 28:1011, 28:1021, 28:1102, 28:1107, 28:1176, 28:1207, 29:139, 29:180, 29:518, 29:752, 29:762, 29:772, 33:861, 05:33, 05:65, 05:88, 37:281, 06:111, 06:133, 44:143, 44:144, 44:145, 44:146, 44:147, 44:182, 06:208, 45:7, 06:231, 06:248, 06:252, 47:11, 47:45, 47:46, 06:289, 06:295, 06:298, 06:308, 06:328, 06:334, 06:337, 06:346, 06:356, 06:362, 06:365, 06:374, 06:392, 06:396, 06:398, 06:399, 06:401, 06:409, 06:413, 06:415, 50:745, 50:928, 50:1203, 50:1267, 50:1294, 50:1398, 50:1427, 50:1511, 50:1516, 50:1590, 50:1742, 50:1754, 52:261, 54:623, 54:624, 54:625, 54:682, 54:683, 54:684, 54:931, 54:1936, 54:2204, 54:2221, 57:282, 57:411, 57:708, *1211*
\@gobble@AddToHook@args 08:2896, 08:2897
\@gobble@IncludeInRelease 03:71
\@gobble@RemoveFromHook@arg 08:2899, 08:2900
\@gobblecr 18:557, 18:558
\@gobblefour 28:24, 28:427, 28:583, 28:585, 28:589, 28:591, 28:601, 28:605, 28:729, 28:781, 06:208, 50:1296, 50:1429, 50:1518
\@gobblethree 06:208, 50:128, 06:644, 06:656, *1222*
\@gobbletwo 13:12, 20:32, 20:102, 20:160, 24:566, 24:600, 28:132, 37:23, 37:55, 37:89, 37:120, 37:151, 37:159, 05:123, 05:128, 06:175, 06:176, 06:208, 49:11, 49:13, 50:1266, 50:1397, 50:1510, 52:260, 52:489, 57:306, *1227*
\@gtempa 20:685, 20:686, 20:692, 20:693, 20:694, 20:719, 20:720, 20:722, 20:723, 20:724, 41:3, 41:5, 41:6, 41:7, 41:8, 06:126, 06:127, 06:181, 06:183, 50:349, 50:351, 50:364, 50:365, 50:384, 50:386, 50:400, 50:402, 50:412, 50:414, 06:575, 06:583, *1206*
\@halfwidth 42:2, 42:126, 42:129, 42:131, 42:227, 42:299, 42:302, 42:324, 42:333, 42:349, 42:361, 42:364, 42:383, 42:391, 42:405, 42:416, 42:421, 42:679, 42:705, 42:724, 42:725, 42:726, 42:765, 42:780, 42:781, 42:782
\@halignto 41:170, 41:174, 41:177, 41:191
\@hangfrom 44:66, 44:117, 44:138
\@height 18:397, 18:405, 18:431, 18:439, 21:293, 21:295, 26:190, 30:347, 30:565, 30:566, 30:568, 30:569, 06:26, 40:202, 40:207, 40:280, 40:290, 40:550, 40:603, 40:610, 41:186, 41:219, 41:359, 41:376, 42:227, 42:300, 42:303, 42:324, 42:333, 42:351, 42:359, 42:383, 42:391, 42:407, 42:414, 42:589,

42:599, 42:725, 42:781, 54:1855, 02:502
\@highpenalty 18:119, 57:3
\@hightab 41:11, 41:21, 41:23, 41:74, 41:86, 41:95, 41:96, 41:111, 41:146, 41:147, *1229*
\@hline . 42:167, 42:179, 42:226, 42:265
\@holdpg 54:124, 54:302, 54:304, 54:305, 54:310, 54:311, 54:312
\@hspace . 18:512, 18:513, 18:529, *1197*
\@hspacer 18:512, 18:528, *1197*
\@hvector 42:244, 42:259, 42:265
\@ialph *1212*
\@icentercr 37:430, 37:431
\@iden 06:215
\@if 06:171, 06:172, 06:174
\@if.. . . . . *1212*
\@if@DeclareRobustCommand 09:126, 06:497, 06:554, 06:565, 06:566, 06:570, 06:664, 06:665, 06:668, 06:691, *101*
\@if@newcommand . . 09:127, 09:138, 06:316, 06:498, 06:555, 06:566, 06:608, 06:620, 06:626, 06:692, *103*
\@if@newlist *1239*
\@if@pti@ns 50:224, 50:227, 50:229, 50:246, 50:247
\@if@ptions 50:221, 50:222, 50:223, 50:963, 50:1058
\@if@short@command *1212*
\@ifatmargin 41:55, 41:106
\@ifbothcounters 22:109, 22:120, 22:128, 22:136, 22:152, 22:154, 22:163, 22:165
\@ifclasslater . 50:163, 50:171, 50:179
\@ifclassloaded 50:155, 50:268, 50:277
\@ifclasswith . . 50:221, 50:270, 50:279
\@ifdefinable 21:14, 21:17, 22:11, 23:3, 29:618, 40:154, 06:84, 06:86, 06:130, 06:132, 43:7, 43:15, 43:22, 06:250, *1212*
\@iffileonpath 20:487, 20:541, 20:563, 20:580, 20:601
\@ifl@aded 50:155, 50:156, 50:157, 50:909, 50:1039, 50:1057, *1002*
\@ifl@t@r 21:1538, 05:201, 05:207, 50:169, 50:172, 50:177, 50:180, 50:184, 50:188, 50:190, 50:201, 50:202, 50:766
\@ifl@ter 21:1583, 21:1584, 50:163, 50:164, 50:183, 50:951, 50:1086
\@ifl@ter@@ 21:1583, 21:1584
\@ifnch 05:49, 05:50, 06:782, 06:784, 06:796
\@ifnextchar 18:94, 18:558, 20:633, 22:13, 26:411, 05:45, 37:429, 38:355, 39:167, 40:9, 40:11, 40:18, 40:20, 40:26, 40:68, 40:160, 40:161, 40:167, 40:168, 40:175, 40:179, 40:224, 40:225, 40:231, 40:232, 40:237, 40:295, 40:303, 40:311, 40:318, 40:322, 40:425, 40:429, 40:433, 40:534, 40:539, 40:562, 40:569, 40:574, 41:57, 41:181, 41:203, 41:210, 42:23, 42:132, 42:143, 42:453, 43:3, 43:5, 43:32, 43:38, 45:27, 45:264, 45:324, 45:452, 45:519, 45:536, 47:3, 47:17, 47:29, 50:356, 50:370, 50:761, 50:785, 50:793, 50:800, 50:1212, 50:1215, 50:1347, 50:1350, 54:211, 06:778, 06:783, 06:797, 54:2012, 01:82, *1240*
\@iforloop 13:21, 13:22
\@ifpackagelater 50:163, 50:170, 50:178, *977*
\@ifpackageloaded 50:155, 50:267, 50:276, 54:1996, *977*
\@ifpackagewith 50:221, 50:269, 50:278, *977*
\@iframebox 40:238, 40:239, 40:240
\@iframepicbox 40:295, 40:296
\@ifstar 18:60, 18:72, 18:377, 18:512, 22:150, 22:161, 24:206, 27:121, 37:418, 37:425, 37:725, 37:734, 38:390, 41:56, 06:73, 41:202, 41:209, 42:142, 42:604, 44:52, 44:142, 50:498, 50:540, 06:797, 54:1860
\@ifundefin@d@i 06:740, 06:741, 06:758, 06:761
\@ifundefin@d@ii 06:740, 06:743, 06:746
\@ifundefined 22:3, 22:7, 22:16, 22:98, 22:110, 22:112, 24:100, 24:186, 26:424, 28:472, 33:54, 33:863, 35:85, 37:216, 37:233, 37:249, 37:270, 37:288, 37:298, 05:132, 06:127, 06:134, 06:154, 06:161, 43:21, 06:183, 06:194, 47:40, 47:68, 47:85, 06:289, 06:295, 06:328, 49:3, 49:7, 06:334, 06:356, 06:362, 50:128, 50:153, 06:392, 50:451, 50:457, 50:474, 50:486, 50:564, 06:409, 50:598, 50:617, 50:911, 06:490, 06:530, 06:531, 52:484, 06:735, 07:3014, *1211*
\@ignore... . . . . *1233*
\@ignorefalse 24:362, 37:4, 37:255, 37:275, 37:293, 37:303, 37:339, 37:356, 37:372, 37:381, 45:384, *1196*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@ignoretrue 18:201, 18:214, 24:358, 37:4, 37:7, 38:348, 38:351, 38:384, 38:544, *1232*
\@iiiminipage 40:427, 40:431, 40:434, 40:435, 40:436
\@iiiparbox 40:305, 40:313, 40:320, 40:323, 40:324, 40:325, 40:472
\@iiminipage 40:430, 40:432
\@iinput 20:633, 20:634, *460*
\@iiparbox 40:319, 40:321
\@iirsbox 40:574, 40:583
\@imakebox 40:26, 40:41, 40:177
\@imakepicbox 40:68, 40:69, 40:182, 40:297
\@iminipage 40:426, 40:428
\@in@minipage@envtrue 40:451
\@include 20:228, 20:271, 20:287, 20:291, *1252*
\@includeinreleasefalse 03:74, 03:79, 03:133, 03:141, 06:873
\@includeinreleasetrue 03:123
\@index 46:18, 46:19, 46:35
\@indexfile 46:4, 46:5, 46:14
\@inlabel *802*
\@inlabelfalse 39:28, 39:104, 39:208, 54:165, 54:192, *1207*
\@inlabeltrue 39:28, 39:202
\@inmatherr 14:285, 39:112, 39:166, 42:604, *1209*
\@inmathwarn 21:3
\@inpenc@test 57:356, 57:423
\@input 20:34, 20:104, 20:162, 20:299, 20:361, 20:408, 20:672, 44:152, 57:711, *1036*
\@input@ 20:319, 20:346, 20:379, 20:398, 20:423, 20:674, 24:418, 47:51, *1203*
\@input@file@exists@with@hooks 52:143, *1255*
\@inputcheck 20:482, 20:483, 20:490, 20:536, 20:537, 20:544, 20:558, 20:559, 20:566, 20:588, 20:589, 20:592, 20:605, 20:606, 20:609, 01:175, 01:176, 01:179, 01:187, 05:38, 05:39, 05:42, 06:38, 06:45, 50:1255, 50:1256, 50:1290, 50:1386, 50:1387, 50:1423, 50:1498, 50:1499, 50:1506, 01:54, 02:307, *1257*
\@insertfalse . . . 54:1077, 54:1227, 54:1398, 54:1479, 54:1574, 54:1695
\@inserttrue 54:1003, 54:1048, 54:1165, 54:1333, 54:1653, 54:1780
\@invalidchar 14:290
\@iparbox 40:304, 40:312, 40:317
\@irsbox 40:562, 40:569, 40:574, 40:575
\@isavebox 40:175, 40:176
\@isavepicbox 40:180, 40:181
\@ishortstack 42:133, 42:141
\@istackcr 42:143, 42:144
\@itabcr 41:57, 41:58
\@item 39:167, 39:180
\@itemdepth 39:265, 39:267, 39:268, 39:269, *816*
\@itemfudge 41:38, 41:44, 41:82
\@itemitem 39:269, 39:272
\@itemlabel . 39:44, 39:96, 39:167, *1201*
\@itempenalty 18:17, 39:23, 39:199, *803*
\@itemspacing *816*
\@iwhiledim 13:7
\@iwhilenum 13:3
\@iwhilesw 13:10
\@ixpt 24:743
\@ixstackcr 42:142
\@kernel@... . . . . *980*
\@kernel@Ref 35:151, 35:156
\@kernel@after@⟨*hook*⟩ *215*
\@kernel@after@begindocument . . . 09:61, 20:58, 20:76, 05:7, 51:235, *1257*
\@kernel@after@begindocument@before 20:16, 20:76, 25:731, *446*
\@kernel@after@enddocument 05:1, 37:15, 37:81, 53:352
\@kernel@after@enddocument@afterlastpage . . . 05:1, 37:19, 37:85, 53:358, 08:455
\@kernel@after@para@after 16:8, 16:106, *405*
\@kernel@after@para@end 16:8, 16:99, *405*
\@kernel@after@shipout@background 53:72, 53:160, *1256*
\@kernel@after@shipout@lastpage 53:114, 53:119, 53:160, 53:378, 53:383, *1081*
\@kernel@before@⟨*hook*⟩ *215*
\@kernel@before@begindocument 20:56, 20:76, 05:7, 53:407, *1257*
\@kernel@before@enddocument 37:13, 37:79, 37:180, *765*
\@kernel@before@insertmark 48:221, 48:249, *959*
\@kernel@before@para@before 16:8, 16:21, 16:54, *404*
\@kernel@before@para@begin 16:8, 16:30, 16:62, *405*
\@kernel@before@shipout@background 53:68, 53:70, 53:160, *1069*
\@kernel@currentdata 35:163

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@kernel@currpathstack 50:45, 50:47, 50:91, 50:123, *982*
\@kernel@eqno . . 38:443, 38:445, 38:451
\@kernel@leqno . 38:444, 38:446, 38:452
\@kernel@make@file@csname 52:287, 52:290, 52:310, 52:313, 52:331, 52:334, 52:363
\@kernel@new@label@record@testdef 36:149, 36:279, 37:25
\@kernel@pageref 35:65, 35:69
\@kernel@pageref@exp 35:70
\@kernel@ref 35:64, 35:67, 35:151
\@kernel@ref@exp 35:72
\@kernel@refstepcounter 35:132, 35:146, 43:31, *1267*
\@kernel@rename@newcommand 06:297, 06:314, 06:351, 06:379, 06:384, 06:397, *95*
\@kernel@reserved@label@data 35:100, 35:107
\@kernel@sRef 35:153, 35:156
\@kernel@spageref 35:26, 35:43, 35:60, 35:65, 35:69
\@kernel@sref . 35:13, 35:25, 35:30, 35:42, 35:47, 35:59, 35:64, 35:67, 35:153
\@killglue 42:59, 42:73, 42:103, 42:115, 42:123
\@kludgeins . 54:321, 54:322, 54:323, 54:325, 54:378, 54:379, 54:425, 54:426, 54:506, 54:522, 54:523, 54:529, 54:530, 54:531, 54:540, 54:556, 54:560, 54:570, 54:1856, 54:1887
\@labels 39:27, 39:170, 39:171, 39:213, 39:230, 39:231, *802*
\@largefloatcheck 45:192, 45:213, 45:238, 45:256, *1210*
\@lastchclass 41:262, 41:272, 41:273, 41:275, 41:283, 41:308, 41:322, 41:326, 41:335, 41:348, 41:349
\@latex@error 08:2923, 03:100, 03:159, 03:184, 03:195, 14:163, 14:217, 14:233, 14:239, 14:241, 14:244, 14:246, 14:248, 14:252, 14:254, 14:256, 14:259, 14:263, 14:268, 14:272, 14:274, 14:276, 14:278, 14:279, 14:281, 14:284, 14:288, 14:290, 17:10, 20:224, 20:268, 20:286, 21:50, 21:84, 24:5, 24:28, 24:58, 24:102, 24:144, 24:187, 24:253, 24:339, 26:105, 27:100, 27:111, 28:23, 28:68, 28:97, 28:117, 28:170, 28:201, 28:231, 28:263, 28:364, 28:380, 28:478, 28:499, 28:551, 28:601, 28:605, 28:647, 28:652, 28:707, 28:775, 28:781, 28:825, 28:829, 28:833, 28:868, 28:872, 28:876, 28:933, 28:943, 28:1028, 28:1033, 28:1036, 28:1068, 28:1071, 28:1144, 28:1147, 28:1150, 28:1217, 28:1223, 29:49, 29:60, 29:82, 29:93, 29:600, 29:720, 32:143, 05:80, 37:250, 37:271, 37:289, 37:299, 37:655, 37:672, 37:686, 37:701, 37:713, 38:422, 39:243, 41:100, 41:109, 06:128, 06:155, 44:31, 45:6, 45:83, 47:71, 47:88, 06:290, 06:329, 06:357, 50:688, 50:739, 50:752, 50:781, 50:964, 50:1012, 50:1059, 50:1171, 50:1188, 50:1196, 50:1201, 50:1223, 50:1280, 50:1358, 50:1413, 06:481, 50:1757, 50:1787, 06:522, 54:236, 54:392, 54:1878, 54:1895
\@latex@info 14:163, 14:208, 21:85, 33:76, 06:236, 06:310, 06:348, 06:376, 52:487, 06:835, *394*
\@latex@info@no@line 14:163, 14:209, 53:89, 54:580, *1214*
\@latex@note 14:190, 24:33
\@latex@note@no@line 14:190, 50:1237, 50:1257, 50:1263, *1258*
\@latex@warning 14:163, 14:215, 20:344, 20:396, 21:55, 22:54, 22:72, 35:18, 35:35, 35:52, 36:211, 36:221, 36:227, 40:51, 40:252, 40:347, 40:447, 42:449, 45:260, 47:42, 47:69, 47:86, 50:1319, 50:1326, 50:1452, 50:1458, 50:1541, 50:1547, 53:174, 54:2034
\@latex@warning@no@line 14:163, 14:216, 20:19, 20:89, 20:146, 20:739, 35:8, 35:88, 35:89, 36:133, 36:134, 05:98, 37:62, 37:69, 37:127, 37:134, 37:166, 44:32, 06:202, 50:352, 50:366, 50:767, 50:952, 50:1087, 50:1239, 50:1388, 50:1394, 50:1417, 50:1500, 50:1507, 50:1574, 50:1658, 53:79, 53:98, 53:369, 54:245, 54:277, 54:1831, 54:2100
\@latexbug 14:277, 54:335, 54:1817
\@latexerr 14:215, 54:439, *1211*
\@latexinfo *1209*
\@latexrelease@catcode@null 07:6, 07:3015
\@lbibitem 47:3, 47:4
\@ldots 30:511, 30:513
\@leftcolumn 54:123, 54:2241, 54:2262, 54:2286, 54:2295

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@leftmark 49:16, 49:79
\@let@token 18:457, 18:458, 18:465, 32:83, 32:96, 05:49, 05:51, 38:256, 38:258, 38:261, 06:782, 06:785, 06:788, 06:796, *1228*
\@lign 38:208, 38:210
\@linechar . . 42:190, 42:191, 42:192, 42:196, 42:197, 42:199, 42:204, 42:206, 42:207, 42:208, 42:209, 42:211, 42:215, 42:216, 42:219, 42:220, 42:225, 42:272, 42:669, *1240*
\@linefnt . . . 42:124, 42:127, 42:190, 42:265, 42:273, 42:304, 42:307, 42:676
\@linelen . . . 42:164, 42:165, 42:176, 42:177, 42:203, 42:210, 42:219, 42:221, 42:226, 42:227, 42:228, 42:241, 42:242, 42:256, 42:257, 42:300, 42:303, 42:305, 42:306, 42:670
\@list *803*
\@listctr . . . 39:226, 39:249, 47:9, *1196*
\@listdepth 39:23, 39:35, 39:38, 39:43, 39:99, 40:456, *802*
\@listfiles 20:62, 20:131, 20:186, 20:753, 20:772
\@listi *803*
\@listii *803*
\@listvi *803*
\@lnbk *1234*
\@loadwithoptions 50:694, 50:716, 50:726, 50:735
\@lowpenalty 18:118, 57:3
\@ltab 41:71, 41:106
\@ltxnomath *1209*
\@m 18:303, 18:336, 18:504, 18:509, 20:45, 20:115, 20:173, 39:80, 42:213, 42:217, 47:37, 02:21, 02:429, 02:431, 02:432, 02:498, 02:499, *382*
\@mainaux 20:3, 20:37, 20:38, 20:107, 20:108, 20:165, 20:166, 20:217, 20:299, 20:342, 20:361, 20:394, 20:408, 20:433, 37:22, 37:88, 37:150, *1220*
\@makebox 40:11, 40:20, 40:25
\@makecaption 45:24, *1166*
\@makecol 54:263, 54:415, 54:462, 54:484
\@makefcolumn 54:395, 54:396, 54:404, 54:406, 54:442, 54:444, 54:452, 54:454, 54:2199, 54:2201, 54:2217, 54:2218
\@makefnmark 45:400, 45:532
\@makefntext 40:489, 40:506, 40:523, 45:476, 45:494, 45:512
\@makeother 01:110, 24:431, 24:432, 24:433, 24:434, 24:435, 24:436, 24:437, 24:438, 24:439, 24:440, 24:441, 37:541, 37:562, 37:707, 37:722, 37:732, 50:438, 50:439, 50:1301, 50:1434, 50:1523, 06:800, 06:801, 01:60, 01:81
\@makepicbox 40:10, 40:19, 40:67, 42:366, 42:424
\@makespecialcolbox 54:507, 54:526, *1199*
\@marbox 45:320, 45:322, 45:326, 45:330, 45:331, 45:379, 54:1816, 54:1826, 54:1829, 54:1837, 54:1839, 54:1840, 54:1842, 54:1843, 54:1844, 54:1853
\@marginparreset 45:343, 45:361, 45:370
\@markright 49:37, 49:62, 49:77
\@maxdepth . . 20:60, 20:129, 20:184, 54:91, 54:490, 54:518, 57:152, *1227*
\@maxtab 41:2, 41:94, *1210*
\@medpenalty 18:119, 57:3
\@meta@family@list 29:116, 29:141, 29:153, 29:242, 29:506
\@midlist 54:66, 54:503, 54:504, 54:1031, 54:1033, 54:1145, 54:1309, 54:1960, 54:1988
\@minipage... . . . . *1232*
\@minipagefalse 39:205, 40:420, 40:422, 40:469, 45:187, 45:250, 45:345, 45:363
\@minipagerestore 40:457, 40:459
\@minipagetrue 40:421, 45:186
\@minus 06:26, 54:2344, 54:2345, 54:2346, 54:2349, 54:2350
\@missing@onefilewithoptions 50:926, 50:982, 50:1100
\@missingfile@area 20:646, 20:687, 20:700, 50:984
\@missingfile@base 20:646, 20:688, 20:701, 50:985
\@missingfile@ext 20:646, 20:689, 20:702, 50:986
\@missingfileerror 20:641, 20:656, 20:666, 20:675, 50:983, 50:1080, *461*
\@mkboth 49:11, 49:13
\@mklab 39:45, 39:164
\@mkpream . 41:189, 41:234, 41:262, *1219*
\@mparbottom 45:387, 45:388, 54:120, 54:480, 54:1827, 54:1835, 54:1836, 54:1837, 54:1838
\@mpargs 40:440, 40:472
\@mparswitchfalse 54:104
\@mpfn 40:454, 45:452, 45:457, 45:541, 45:545

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@mpfootins 40:446, 40:463, 40:464, 40:467, 40:473, 40:480, 40:481, 40:498, 40:499, 40:515, 40:516
\@mpfootnotetext 40:455, 40:475
\@mplistdepth 40:456, 40:473
\@multicnt 41:370, 41:372, 41:373, 41:374, 41:381, 41:382, 41:383, 42:103, 42:104, 42:106, 42:115, 42:116, 42:118, 42:666, 42:703, 42:705, 42:706, 42:707, 42:709, 42:710, 42:716, 42:722, 42:733, 42:737, 42:763, 42:765, 42:767, 42:769, 42:770, 42:774, 42:778, 42:789, 42:793, *1228*
\@multiplelabels . . . 20:33, 20:103, 20:161, 35:87, 35:93, 36:132, 37:60, 37:66, 37:125, 37:131, 37:164, 37:170
\@multiput 42:86, 42:95, 42:98
\@multispan 41:371, 41:375, 41:379
\@mypkg@name *1027*
\@mypkg@other@name *1027*
\@namedef 20:446, 24:135, 24:136, 24:160, 25:5, 25:525, 26:418, 28:401, 28:405, 28:414, 33:57, 33:93, 33:866, 35:90, 37:287, 37:347, 37:364, 37:378, 37:584, 37:593, 38:426, 38:427, 06:50, 41:175, 43:12, 43:13, 43:18, 43:19, 43:23, 43:24, 43:25, 50:1214, 50:1349, 52:482, 57:480, 57:481, *1198*
\@nameuse 20:336, 20:392, 20:431, 20:445, 06:51, 43:23, 49:5, 50:920, 52:486, 54:611, 54:669, *1214*
\@nbitem 39:192, 39:245
\@ne 02:16, *1214*
\@needsf@rmat . . 50:762, 50:765, 50:770
\@needsformat . . 50:750, 50:760, 50:764
\@negargfalse 42:186
\@negargtrue 42:185
\@newcommand 06:79, 06:80
\@newctr 22:13, 22:15, 43:8
\@newenv 06:150, 06:151, 06:160
\@newenva 06:149, 06:148
\@newenvb 06:151, 06:150
\@newfontswitch *1208*
\@newl@bel 35:84, 37:24, 37:90, 37:152, 47:10, *1226*
\@newline 18:95, 18:97
\@newlistfalse 39:29, 39:33, 39:108, 39:206, 54:604, 54:662
\@newlisttrue 39:29, 39:33, 39:87
\@next 45:60, 45:129, 45:319, 45:320, 54:9, 54:215, 54:313, 54:881, 54:901, 54:1816, 02:275
\@nextchar 41:269, 41:270, 41:330, 41:331, 41:332
\@nil 03:13, 03:19, 03:83, 03:104, 03:105, 03:117, 03:118, 03:165, 03:166, 03:167, 13:13, 13:19, 13:27, 19:14, 20:247, 20:248, 20:249, 21:89, 21:110, 01:145, 01:146, 21:996, 21:1000, 21:1063, 21:1075, 21:1077, 24:383, 24:394, 24:510, 24:525, 24:629, 24:632, 24:633, 24:641, 25:431, 25:433, 25:465, 25:467, 25:645, 25:647, 25:671, 25:673, 26:350, 26:351, 26:353, 26:366, 26:372, 26:376, 26:377, 26:413, 26:434, 26:439, 26:519, 26:533, 27:26, 27:44, 27:53, 27:57, 28:40, 28:571, 28:579, 28:612, 28:1228, 28:1230, 32:58, 32:62, 06:53, 06:54, 06:58, 06:63, 41:367, 41:368, 06:135, 50:90, 50:91, 50:97, 50:105, 50:108, 50:115, 50:140, 50:191, 50:192, 50:203, 50:204, 50:214, 50:215, 50:217, 50:447, 50:470, 50:514, 50:520, 50:636, 50:656, 50:660, 50:666, 50:670, 50:877, 50:886, 50:903, 50:904, 06:451, 06:452, 50:1631, 50:1636, 50:1639, 50:1641, 50:1642, 50:1657, 50:1677, 50:1694, 50:1749, 50:1771, 50:1795, 52:168, 52:176, 52:369, 52:411, 52:413, 06:633, 06:776, 06:777
\@nmbrlistfalse 39:33, 39:46, 39:91, *1196*
\@nmbrlisttrue 39:249
\@nnil 13:13, 13:20, 13:21, 13:22, 13:28, 20:248, 20:249, 24:214, 24:218, 24:219, 24:220, 24:235, 26:179, 26:181, 26:345, 26:347, 26:359, 26:361, 26:366, 26:380, 26:382, 26:389, 26:400, 26:401, 26:403, 26:434, 26:439, 42:13, 50:822, 50:823, 50:830, 50:850, 50:851, 50:858, 06:429, 06:435
\@no@font@optfalse 27:17, 27:129
\@no@lnbk 18:9, 18:10, 18:41, *1234*
\@no@pgbk 18:7, 18:8, 18:33, *1227*
\@nobreak *943*
\@nobreak... . . . . *1232*
\@nobreakfalse 18:121, 18:123, 39:217, 44:94, 44:129, 44:157, 45:182, 54:167, 54:194, 54:1154, 54:1320, *1233*
\@nobreaktrue . . 18:122, 44:126, 45:181
\@nocnterr 14:240, *1207*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@nocounterr 14:240, 22:4, 22:8, 22:16, 22:110, 22:112, 43:21, *1208*
\@nodocument 14:245, 16:143, 20:68, 20:137, 20:192, 20:296, 20:358, 37:241, 45:39, 45:108, 54:158, 54:185, 54:214, *1233*
\@noitemargfalse 39:32, 39:224
\@noitemargtrue 39:32, 39:167
\@noitemerr 14:280, 18:264, 18:281, 18:351, 18:373, 39:69, 39:81, 39:107, *1269*
\@noitemerror *1220*
\@noligs 37:542, 37:563, 37:723, 37:733, 37:744, *1231*
\@nolnbk *1234*
\@nolnerr 14:238, 18:43, 18:114, 37:417, 37:424, *1210*
\@nomath 24:1, 24:337, 29:551, 29:586, 29:592, 29:613, 29:615, 29:637, *1194*
\@noparitemfalse 39:30, 39:169
\@noparitemtrue 39:30, 39:66
\@noparlistfalse 39:31, 39:70
\@noparlisttrue 39:31, 39:67
\@nopgbk *1227*
\@normalcr . . 18:53, 18:93, 40:419, *1250*
\@normalsize 50:4, 50:5, *1207*
\@normalsize␣check *1207*
\@noskipsec *802*
\@noskipsecfalse 20:54, 20:124, 20:179, 44:98, 54:160, 54:187
\@noskipsectrue 44:38, 44:95
\@notdefinable 14:232, 06:136, 06:137, 06:141, 06:478, 06:519
\@notprerr 14:283, 20:66, 20:135, 20:190
\@nthm 43:3, 43:4
\@nxttabmar 41:11, 41:21, 41:23, 41:25, 41:75, 41:111, 41:112, 41:118, 41:119, *1229*
\@obsoletefile 20:738
\@oddfoot 49:11, 49:14, 49:15, 54:126, 54:614, 54:673
\@oddhead 49:11, 49:14, 54:125, 54:614, 54:673
\@onefilewithoptions 50:814, 50:818, 50:824, 50:842, 50:846, 50:852, 50:869, 50:873, 50:879, 50:895, 50:897, 50:980, 50:1047, 50:1600, *1260*
\@onefilewithoptions@clashchk 50:912, 50:962
\@onelevel@sanitize 45:42, 45:111, 06:802, *1218*
\@onfilewithoptions *1006*
\@onlypreamble 08:2804, 20:197, 20:206, 20:243, 20:740, 20:800, 21:23, 21:24, 21:61, 21:62, 21:66, 21:125, 21:145, 21:189, 21:190, 21:204, 24:17, 24:115, 24:117, 24:123, 24:139, 24:167, 24:182, 24:203, 24:208, 24:250, 24:537, 26:419, 27:28, 27:36, 27:42, 27:79, 27:83, 27:88, 27:93, 27:98, 27:108, 27:126, 27:127, 27:128, 27:134, 27:138, 27:142, 28:17, 28:19, 28:44, 28:46, 28:107, 28:116, 28:136, 28:394, 28:395, 28:408, 28:454, 28:502, 28:514, 28:516, 28:529, 28:554, 28:611, 28:613, 28:655, 28:694, 28:710, 28:787, 28:881, 28:890, 28:946, 28:949, 28:952, 28:972, 28:985, 28:1039, 28:1074, 28:1085, 28:1099, 28:1153, 28:1173, 28:1177, 28:1241, 32:140, 32:141, 33:58, 33:59, 33:60, 33:867, 35:92, 06:66, 06:188, 06:190, 06:199, 06:207, 46:12, 46:29, 47:64, 47:81, 48:16, 48:364, 50:10, 50:13, 50:85, 50:112, 50:120, 50:122, 50:154, 50:357, 50:420, 50:426, 50:491, 50:494, 50:495, 50:506, 50:507, 50:508, 50:535, 50:541, 50:554, 50:591, 50:609, 50:629, 50:649, 50:673, 50:678, 50:682, 50:685, 50:693, 50:714, 50:717, 50:727, 50:746, 50:759, 50:764, 50:770, 50:798, 50:803, 50:891, 50:980, 50:1105, 50:1114, 50:1122, 50:1123, 50:1141, 50:1169, 50:1178, 50:1185, 50:1186, 50:1194, 50:1199, 50:1204, 50:1580, 50:1581, 50:1582, 50:1583, 50:1585, 51:199, *1258*
\@opargbegintheorem 43:45, 43:48
\@opcol 54:264, 54:272, 54:396, 54:415, 54:444, 54:462, 54:467, *949*
\@options 50:628
\@othm 43:3, 43:20
\@outerparskip 39:1, 39:88, 39:117, 39:176, 39:246
\@outputbox . 48:382, 48:385, 48:389, 48:413, 48:416, 48:420, 54:122, 54:487, 54:489, 54:509, 54:512, 54:513, 54:533, 54:535, 54:536, 54:541, 54:544, 54:549, 54:551, 54:558, 54:564, 54:566, 54:640, 54:699, 54:727, 54:733, 54:743, 54:744, 54:767, 54:774, 54:860, 54:863, 54:866, 54:872, 54:873, 54:2241, 54:2245, 54:2246, 54:2260, 54:2266, 54:2286, 54:2292, 54:2301, *965*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@outputdblcol 54:471, 54:2236, 54:2238, 54:2282, 54:2283
\@outputpage 54:405, 54:454, 54:474, 54:594, 54:2270, 54:2275, 54:2308, 54:2316
\@oval 42:453, 42:470
\@ovbtrue 42:475, 42:503, 42:533
\@ovdx 42:431, 42:486, 42:488, 42:494, 42:496, 42:515, 42:517, 42:525, 42:527, 42:542, 42:550, 42:552, 42:588, 42:591, 42:598, 42:600, 42:692, 42:693, 42:694, 42:695, 42:711, 42:712, 42:713, 42:716, 42:732, 42:752, 42:753, 42:754, 42:755, 42:771, 42:772, 42:774, 42:788
\@ovdy 42:431, 42:487, 42:489, 42:495, 42:496, 42:516, 42:518, 42:526, 42:527, 42:543, 42:551, 42:552, 42:563, 42:568, 42:576, 42:580, 42:699, 42:700, 42:701, 42:702, 42:717, 42:718, 42:719, 42:722, 42:736, 42:759, 42:760, 42:761, 42:762, 42:775, 42:776, 42:778, 42:792
\@ovhlinefalse 42:476, 42:504
\@ovhlinetrue 42:456, 42:460, 42:464, 42:482, 42:488, 42:510, 42:517
\@ovhorz 42:493, 42:494, 42:524, 42:525, 42:549, 42:550, 42:583
\@ovltrue 42:475, 42:503, 42:533
\@ovri 40:33, 42:431, 42:485, 42:514, 42:541, 42:563, 42:576, 42:592, 42:601
\@ovro 42:431, 42:485, 42:494, 42:495, 42:514, 42:525, 42:526, 42:541, 42:550, 42:551, 42:562, 42:568, 42:575, 42:580, 42:587, 42:597, 42:613, 42:620, 42:629, 42:638
\@ovrtrue 42:475, 42:503, 42:533
\@ovttrue 42:475, 42:503, 42:533
\@ovvert 42:491, 42:492, 42:520, 42:522, 42:545, 42:547, 42:556
\@ovvlinefalse 42:476, 42:504
\@ovvlinetrue 42:459, 42:481, 42:489, 42:509, 42:518
\@ovxx 42:431, 42:479, 42:481, 42:482, 42:486, 42:492, 42:493, 42:507, 42:509, 42:510, 42:515, 42:522, 42:524, 42:536, 42:538, 42:542, 42:547, 42:549, 42:587, 42:597, 42:689, 42:690, 42:691, 42:695, 42:704, 42:705, 42:714, 42:715, 42:716, 42:731, 42:749, 42:750, 42:751, 42:755, 42:764, 42:765, 42:773, 42:774, 42:787
\@ovyy 42:431, 42:480, 42:481, 42:482, 42:487, 42:494, 42:508, 42:509, 42:510, 42:516, 42:525, 42:537, 42:538, 42:543, 42:550, 42:560, 42:573, 42:696, 42:697, 42:698, 42:702, 42:704, 42:720, 42:721, 42:722, 42:735, 42:756, 42:757, 42:758, 42:762, 42:764, 42:777, 42:778, 42:791
\@p@pfilename 50:90, 50:97, 50:105, 50:108, 50:115, 50:120
\@p@pfilepath . . . 50:91, 50:138, 50:147
\@p@pfilepath@aux 50:139, 50:140, 50:148
\@pagedp . . . 54:119, 54:310, 54:315, 54:1095, 54:1248, 54:1845, 54:1855
\@pageht . . . 54:118, 54:311, 54:315, 54:317, 54:318, 54:319, 54:323, 54:1094, 54:1247, 54:1828, 54:1835
\@par 15:3, 15:5
\@parboxrestore . . 40:334, 40:361, 40:419, 40:453, 40:484, 40:502, 40:519, 45:19, 45:100, 45:169, 45:342, 45:360, 45:470, 45:489, 45:507, 54:221, 54:605, 54:663, *1233*
\@parboxto 40:325, *1241*
\@parmoderr 14:273, 45:58, 45:127, 45:316
\@parse@version 03:83, 03:104, 03:105, 03:117, 03:118, 03:165, 03:166, 03:167, 50:197, 50:203, 50:204, 50:214, 50:1642, 50:1657, 50:1694, 50:1771, 50:1795
\@parse@version@ 50:191, 50:192, 50:197, 50:209
\@parse@version@dash 50:215, 50:217, *1246*
\@partaux 20:3, 20:223, 20:267, 20:285, 20:309, 20:311, 20:312, 20:332, 20:371, 20:373, 20:374, 20:388, 20:418, 20:420, 20:421, 20:427, 20:439, 20:441, 20:444
\@partlist 20:232, 20:236, 20:237, 20:239, 20:263, 20:282, 20:305, 20:367, 20:414
\@partswfalse 20:6
\@partswtrue . . . 20:231, 20:261, 20:281
\@pass@ptions 50:443, 50:445, 50:465, 50:466, 50:468, 50:481, 50:482, 50:484, 50:491, 50:492, 50:493, 50:991, 50:1073
\@pboxswfalse . . 40:332, 40:359, 40:438
\@pboxswtrue 40:342, 40:369
\@pdef *1203*
\@penup 38:199, 38:200

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@percentchar . . . 50:1293, 50:1295, 50:1297, 50:1299, 50:1426, 50:1428, 50:1430, 50:1432, 50:1515, 50:1517, 50:1519, 50:1521, 50:1569, 01:90
\@pgbk *1227*
\@picbox 42:6, 42:31, 42:43, 42:51, 42:52, *1195*
\@picht 42:6, 42:29, 42:42, 42:51
\@picture 42:23, 42:24
\@picture@warn 42:223, 42:441, 42:445, 42:449
\@pkgetension *1203*
\@pkgextension 50:31, 50:155, 50:163, 50:221, 50:492, 50:684, 50:687, 50:726, 50:735, 50:825, 50:853, 50:880, 50:1002, 50:1029, 50:1150, 50:1180, 50:1614, 52:482, 52:492, *1004*
\@plus 18:532, 06:26, 44:16, 44:206, 44:229, 49:83, 54:2344, 54:2345, 54:2346, 54:2349, 54:2350, 54:2354, 54:2355, 54:2356, 54:2360, 54:2361, 54:2362
\@pnumwidth 44:218, 44:241
\@popfilename 50:33, 50:960, 50:1095, *981*
\@pr@videpackage 50:356, 50:370, 50:373, 50:394, 50:396, 50:407, 50:409, 50:420
\@preamble 41:190, 41:192, 41:200, 41:237, 41:256, 41:258, 41:259, 41:263, 41:278, 41:296, 41:297, 41:334, 55:129
\@preamblecmds 20:67, 20:136, 20:191, 06:66, 50:1809, 50:1810, *940*
\@preamerr 14:260, 41:199, 41:274, 41:355
\@process@pti@ns 50:553, 50:571, 50:588, 50:595, 50:596, 50:609, 50:613, 50:615
\@process@ptions 50:540, 50:542, 50:554
\@protect@... . . . . *1214*
\@protected@testopt 06:89, 06:101, 06:628, 06:640
\@protecteddef *1209*
\@providesfile 50:430, 57:704, 01:82, 01:83
\@ptionlist 50:152, 50:224, 50:539, 50:968, 50:974, 50:1063, 50:1069, 50:1181
\@pushfilename 50:33, 50:905, 50:1050, *980*
\@put 42:452, 42:496, 42:527, 42:552, 42:620, 42:638
\@qend 14:236, 06:136, 06:776
\@qrelax 06:137, 06:776
\@raw@classoptionslist 50:11, 50:812, 51:97, 51:139
\@rc@ifdefinable 21:14, 06:130, 06:132, 06:250
\@reargdef 06:122, *1203*
\@refundefined 20:55, 20:125, 20:180, 35:3, 37:58, 37:123, 37:162, *1229*
\@reinserts 54:329, 54:332, 54:520
\@remove@eq@value 50:509, 50:576, 50:636
\@remove@tlig 21:996, 21:1004
\@remove@tlig@ 21:996, 21:997
\@remove@tlig@@ 21:997, 21:1000
\@remove@tlig@@@ 21:1010, 21:1029
\@removeelement . 13:32, 50:635, 50:644
\@removefromreset 22:94, 22:118, 22:119, 22:129, 22:152, 22:155
\@renewfontswitch *1203*
\@reqcolroom 54:1094, 54:1095, 54:1098, 54:1100, 54:1101, 54:1106, 54:1110, 54:1112, 54:1140, 54:1141, 54:1247, 54:1248, 54:1252, 54:1255, 54:1256, 54:1261, 54:1268, 54:1270, 54:1302, 54:1303, 54:1414, 54:1416, 54:1418, 54:1421, 54:1423, 54:1497, 54:1500, 54:1503, 54:1508, 54:1510, 54:2024, 54:2141, 54:2146, 54:2149
\@reserveda *562*
\@reset@ptions . . . 50:924, 50:961, 50:989, 50:1054, 50:1096, 50:1106
\@resetactivechars 54:579, 54:602, 54:660
\@resethfps 54:1209, 54:1378, 54:2091, *1210*
\@resetprotect *1224*
\@restorepar . 15:5, 18:388, 18:404, 18:422, 18:438, 39:127, 39:137, *1226*
\@reversemarginfalse . 45:388, 54:103
\@reversemargintrue 45:387
\@rightmark 49:16, 49:80
\@rightskip 37:456, 37:475, 37:493, 39:75, 40:392, 40:413
\@rjfieldfalse 41:34, 41:77
\@rjfieldtrue 41:125
\@rmfamilyhook 29:429, 29:451, 29:463, 29:490
\@roman 22:185, 22:191
\@rsbox 40:562, 40:569, 40:573
\@rtab 41:71, 41:86
\@rule 40:534, 40:539, 40:543
\@sanitize 46:7, 46:18, 46:24, 46:35, 06:800
\@savebox 40:161, 40:168, 40:174
\@savemarbox 45:326, 45:327, 45:330, 45:333

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@savepicbox . . . 40:161, 40:168, 40:178
\@savsf 17:28, 17:30, 17:50, 17:52, 18:124, 18:130, 18:139, 18:158, 18:172, 18:184, 18:198, 18:212
\@savsk 18:124, 18:129, 18:140, 18:159, 18:173, 18:185, 18:199, 18:213
\@scolelt 54:792, 54:857
\@sdblcolelt . . . 54:809, 54:829, 54:858
\@seccntformat . . . 44:60, 44:111, *1197*
\@secondoffive 35:24, 35:27, 35:41, 35:58
\@secondoftwo 03:87, 03:185, 17:5, 20:509, 20:539, 20:585, 20:602, 21:131, 22:238, 22:243, 29:512, 33:71, 33:836, 33:886, 33:902, 35:44, 05:36, 35:80, 05:40, 06:212, 49:17, 50:159, 50:193, 50:205, 50:238, 50:256, 50:305, 50:335, 06:457, 50:1800, 06:586, 06:637, 06:745, 06:752, 06:772, 06:819, 57:710, 01:72
\@secpenalty 18:18, 44:36, 44:50
\@sect 44:54, 44:55
\@seqncr 38:425
\@set@curr@file@aux 52:363, *1264*
\@setckpt 20:439, 20:446, 37:23, 37:89, 37:151, *1227*
\@setfloattypecounts 54:1078, 54:1228, 54:1399, 54:1480, 54:1575, 54:1696, 54:2038
\@setfontsize 29:637
\@setfps 45:34
\@setfpsbit . . . 45:73, 45:75, 45:77, 45:85, 45:143, 45:146, 45:149, 54:2082
\@setmarks . . 54:2252, 54:2254, 54:2269
\@setminipage 40:458, 45:21, 45:177, 45:185, 45:376, *1221*
\@setminpage *1234*
\@setnobreak 45:179, 45:375, *1221*
\@setpar 15:3, 39:78
\@setprotect *1224*
\@setref 35:10
\@setsize 29:637
\@settab 41:71, 41:93
\@settodim 23:17
\@settopoint 23:35
\@setupverbvisiblespace . . 37:585, 37:597, 37:601, 37:662, 37:676, 37:690
\@setupverbvisibletab 37:613, 37:634
\@sffamilyhook 29:429, 29:456, 29:464, 29:491
\@sharp 41:196, 41:235, 41:265, 41:280, 41:281, 41:301, 41:303, 41:305, 41:333
\@shipoutsetup 54:594
\@shortstack 42:132, 42:133
\@show@... . . . . *100*
\@show@DeclareRobustCommand 06:554, 06:568, 06:613, 06:667, 06:670
\@show@DeclareRobustCommand@env 06:691, 06:693
\@show@environment . . . 06:685, 06:686
\@show@environment@begin 06:694, 06:700, 06:708, 06:713
\@show@environment@end 06:698, 06:707, 06:712, *105*
\@show@environment@end@aux 06:716, 06:718
\@show@macro 06:613, 06:703
\@show@newcommand . 06:555, 06:568, 06:621, 06:648, 06:667, 06:673, *102*
\@show@newcommand@aux 06:648, 06:677, 06:695, *103*
\@show@newcommand@env 06:693, 06:692
\@show@nonstop . . . 06:703, 06:724, *105*
\@show@normalenv 06:689, 06:712
\@show@tokens 06:653, 06:658, 06:702, 06:719
\@show@typeout 06:697, 06:702, 06:724, *105*
\@showcommandlisthook 06:551, 06:553, 06:561, 07:1438, *100*
\@showenvironmentlisthook 06:688, 06:690, 07:1440, *1264*
\@skipping@modulefalse 03:154, 03:173, 03:200
\@skipping@moduletrue 03:153, 03:177
\@sline . 42:167, 42:179, 42:184, 42:269
\@slowromancap 22:192, 22:193
\@spaces 14:218
\@specialoutput 54:258
\@specialpagefalse 54:99, 54:611, 54:669
\@specialpagetrue 49:9
\@specialstyle . . 49:9, 54:611, 54:669
\@sptoken 06:785, 06:795
\@sqrt 38:355
\@ssect 44:53, 44:112
\@stackcr 42:139, 42:142
\@star@or@long 06:72, 06:77, 06:124, 06:146, 06:152, 06:178, 06:187, 06:233
\@startcolumn . . 54:265, 54:272, 54:779
\@startdblcolumn 54:779, 54:2274, 54:2277, 54:2314, 54:2320
\@startfield 41:28, 41:46, 41:92, 41:104, 41:125, 41:133
\@startline 41:20, 41:57, 41:62, 41:68, 41:83, 41:154
\@startpagehook *1200*
\@startpbox 41:193, 41:236, 41:266, 41:332, 41:384, 41:386
\@startsection 44:39

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@starttoc 44:149, *1164*
\@stopfield 41:32, 41:48, 41:86, 41:93, 41:125, 41:127, 41:140, 41:142, 41:154
\@stopline 41:30, 41:56, 41:85
\@stpelt 22:20, 22:23
\@string@makeletter 06:806, *109*
\@strip@args 21:76
\@strip@tex@ext 20:227, 20:237, 20:239, 20:244, 20:274, *450*
\@strip@tex@ext@aux . . 20:244, 20:275
\@svector 42:244, 42:259, 42:269
\@sverb 37:646, 37:725, 37:734, 37:737, *781*
\@svsec 44:57, 44:60, 44:66, 44:78, *1197*
\@svsechd 44:76, 44:101, 44:121
\@swaptwoargs 20:619, 20:621, 50:934, 52:153, 52:167, 52:187
\@sxverbatim . . . 37:518, 37:586, 37:593
\@tabacckludge 21:223, 21:225, 21:454, 21:455, 33:193, 33:200, 33:202, *1251*
\@tabacol 41:178, 41:258
\@tabarray 41:170, 41:180, 41:181
\@tabclassiv 41:180, 41:330
\@tabclassz 41:179, 41:282
\@tabcr 41:56, 41:73
\@tabfbox 41:16, 41:80, 41:82
\@tablab 41:72, 41:126
\@tabminus 41:72, 41:117
\@tabplus 41:72, 41:110
\@tabpush 41:11, 41:77, 41:85, 41:140, 41:143, 41:145
\@tabrj 41:72, 41:124
\@tabular 41:174, 41:177, 41:178
\@tabularcr 41:180, 41:208
\@tempa 35:151, 35:152, 35:153, 35:154, 35:178, 35:179, 35:189, 35:190, *1221*
\@tempboxa 12:13, 21:69, 23:21, 23:22, 23:23, 23:28, 23:29, 39:229, 39:235, 39:236, 39:238, 40:29, 40:30, 40:31, 40:32, 40:37, 40:38, 40:39, 40:40, 40:214, 40:248, 40:267, 40:276, 40:286, 40:441, 40:472, 40:579, 40:580, 40:581, 40:588, 40:589, 40:590, 40:591, 42:304, 42:305, 42:447, 42:448, 42:485, 42:490, 42:495, 42:496, 42:514, 42:519, 42:526, 42:527, 42:541, 42:544, 42:551, 42:552, 42:613, 42:614, 42:619, 42:620, 42:629, 42:630, 42:637, 42:638, 42:723, 42:741, 42:779, 42:797, 44:138, 44:139, 45:322, 45:380, 54:307, 54:379, 54:384, 54:385, 54:426, 54:431, 54:432, 54:570, 54:630, 54:637, 54:638, 54:689, 54:696, 54:697, 54:725, 54:729, 54:741, 54:747, 54:754, 54:755, 54:756, 54:757, 54:761, 54:769, *1196*
\@tempcnta 12:7, 28:954, 28:955, 28:956, 28:957, 28:961, 41:242, 41:243, 41:244, 41:245, 42:187, 42:188, 42:214, 42:215, 42:216, 42:229, 42:230, 42:231, 42:232, 42:239, 42:240, 42:254, 42:255, 42:270, 42:271, 42:276, 42:278, 42:279, 42:280, 42:281, 42:282, 42:285, 42:287, 42:288, 42:289, 42:290, 42:291, 42:292, 42:293, 42:294, 42:295, 42:296, 42:335, 42:336, 42:337, 42:338, 42:339, 42:360, 42:361, 42:362, 42:363, 42:364, 42:365, 42:392, 42:393, 42:394, 42:395, 42:396, 42:414, 42:416, 42:418, 42:419, 42:421, 42:423, 42:438, 42:439, 42:440, 42:442, 42:444, 42:446, 42:448, 42:561, 42:566, 42:574, 42:578, 42:615, 42:616, 42:617, 42:618, 42:631, 42:632, 42:634, 42:635, 42:657, 42:658, 42:659, 42:660, 42:661, 42:662, 42:710, 42:730, 42:770, 42:786, 45:62, 45:68, 45:70, 45:79, 45:80, 45:90, 45:91, 45:131, 45:137, 45:139, 45:152, 45:153, 45:159, 45:160, 50:1241, 50:1243, 50:1244, 50:1245, 50:1372, 50:1374, 50:1375, 50:1376, 50:1486, 50:1488, 50:1489, 50:1490, 54:16, 54:18, 54:20, 54:938, 54:939, 54:940, 54:941, 54:961, 54:962, 54:963, 54:964, 54:986, 54:989, 54:1022, 54:1025, 54:1136, 54:1298, 54:1578, 54:1581, 54:1699, 54:1702, 54:1817, 54:1819, 54:1822, 54:1824, 54:1826, 54:1848, 54:2072, 54:2073, 54:2077, 54:2083, 54:2087, 57:227, 57:232, 57:233, 57:234, 57:364, 57:366, 57:367, 57:368, 57:375, 57:377, 57:378, 57:379, 57:385, 57:387, 57:388, 57:389, 57:396, 57:398, 57:399, 57:400, 57:426, 57:428, 57:429, 57:430, 57:452, 57:454, 57:455, 57:456, 57:462, 57:464, 57:465, 57:466, 57:490, 57:495, 57:496, 57:497
\@tempcntb . . . 12:7, 28:955, 28:959, 28:961, 42:279, 42:280, 42:281,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

42:283, 42:284, 42:285, 42:561, 42:562, 42:566, 42:567, 42:574, 42:575, 42:578, 42:579, 45:88, 45:89, 45:90, 45:157, 45:158, 45:159, 54:17, 54:20, 54:21, 54:2083, 54:2084, 54:2085, 57:228, 57:232, 57:491, 57:495
\@tempdima 12:10, 24:219, 24:224, 38:186, 38:189, 38:195, 40:47, 40:48, 40:62, 40:63, 40:247, 40:248, 40:266, 40:267, 40:274, 40:275, 40:276, 40:278, 40:333, 40:334, 40:360, 40:361, 40:439, 40:443, 40:546, 40:549, 40:550, 40:577, 40:579, 40:585, 40:588, 41:35, 41:36, 41:37, 41:88, 41:89, 41:90, 41:91, 41:218, 41:219, 42:210, 42:211, 42:213, 42:214, 42:215, 42:216, 42:217, 42:218, 42:437, 42:438, 42:439, 42:448, 42:486, 42:487, 42:491, 42:492, 42:515, 42:516, 42:520, 42:522, 42:542, 42:543, 42:545, 42:547, 42:616, 42:618, 42:633, 42:635, 42:636, 42:656, 42:657, 42:658, 44:211, 44:212, 44:234, 44:235, 44:248, 45:196, 45:198, 45:218, 45:220, 45:258, 45:259, 45:260, 54:231, 54:232, 54:233, 54:491, 54:493, 54:539, 54:541, 54:542, 54:547, 54:552, 54:556, 54:561, 54:565, 54:921, 54:924, 54:944, 54:954, 54:966, 54:976, 54:1641, 54:1642, 54:1645, 54:1646, 54:1766, 54:1767, 54:1771, 54:1772, 54:1827, 54:1828, 54:1829, 54:1830, 54:1833, 54:1836, 54:1839, 54:1841, 54:2190, 54:2191, 54:2193, 54:2194, *1197*
\@tempdimb 12:10, 24:220, 24:225, 24:649, 24:653, 26:179, 26:180, 26:437, 26:460, 26:461, 26:470, 26:471, 26:475, 26:493, 26:496, 26:499, 26:501, 40:336, 40:337, 40:363, 40:364, 40:547, 40:550, 40:578, 40:580, 40:586, 40:589, 42:211, 42:212, 42:357, 42:359, 42:362, 42:365, 42:481, 42:483, 42:484, 42:509, 42:512, 42:513, 42:538, 42:539, 42:540, 42:611, 42:612, 42:621, 42:627, 42:628, 42:639, 42:647, 42:648, 42:653, 54:944, 54:945, 54:946, 54:947, 54:954, 54:966, 54:967, 54:968, 54:969, 54:976, *1197*
\@tempdimc . . 12:10, 26:454, 26:455, 26:457, 26:458, 26:460, 26:461, 40:74, 40:75, 40:85, 40:86, 40:548, 40:549, 40:550, 42:30, 42:31, 42:32, 42:33, 42:34, 42:35, 42:60, 42:61, 42:63, 42:64, 42:332, 42:333, 42:334
\@tempdimx *1197*
\@tempskipa 12:14, 18:45, 18:48, 18:49, 18:300, 18:307, 18:309, 18:312, 18:333, 18:340, 18:342, 18:345, 26:181, 26:182, 39:116, 39:117, 39:118, 39:174, 39:176, 39:177, 39:178, 39:246, 39:247, 39:248, 44:42, 44:44, 44:45, 44:50, 44:62, 44:63, 44:88, 44:89, 44:91, 44:103, 44:104, 44:113, 44:114, 54:1876, 54:1877, 54:1879, 54:1887
\@tempskipb 12:14, 18:221, 18:223, 18:225, 18:228, 18:230, 18:244, 18:259, 18:276, 18:298, 18:300, 18:301, 18:305, 18:307, 18:309, 18:310, 18:331, 18:333, 18:334, 18:338, 18:340, 18:342, 18:343, 18:365, 18:368, *433*
\@tempswa *1193*
\@tempswafalse . . . 20:303, 20:365, 20:412, 21:1537, 24:94, 24:699, 28:466, 28:541, 28:615, 28:696, 28:1216, 28:1222, 37:26, 37:91, 37:153, 37:533, 37:554, 47:17, 47:29, 50:1214, 50:1230, 50:1349, 50:1365, 52:286, 52:309, 52:330, 54:992, 54:1028, 54:1584, 54:1705, 02:262, 01:62
\@tempswatrue 20:301, 20:306, 20:363, 20:368, 20:410, 20:415, 21:1540, 21:1541, 24:97, 24:700, 24:701, 24:704, 24:707, 28:469, 28:544, 28:618, 28:699, 28:1179, 36:152, 37:196, 37:538, 37:559, 47:17, 47:29, 50:1211, 50:1346, 52:286, 52:309, 52:330, 54:1586, 54:1609, 54:1707, 54:1732, 54:2151, 54:2168, 02:268, 01:63
\@temptokena 12:16, 37:208, 37:212, 37:221, 37:234, 37:235, 49:27, 49:31, 49:38, 49:42, 49:55, 49:56, 49:63, 49:64, 49:77, 49:78, *767*
\@test... . . . . *1232*
\@test@opt *1228*
\@testdef . . 37:24, 37:90, 37:152, 37:194
\@testfalse 54:12, 54:14, 54:15
\@testfp . . . 54:886, 54:906, 54:942, 54:965, 54:2075, 54:2204, 54:2221, *1220*
\@testopt 18:7, 18:8, 18:9, 18:10, 38:396, 06:33, 06:79, 06:99, 06:103, 06:148, *1228*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@testpach 41:270, 41:348
\@testtrue . . . 54:13, 54:21, 54:358, 54:889, 54:908, 54:948, 54:970, 54:2079
\@testwrongwidth . . 54:347, 54:887, 54:943, 54:1116, 54:1430, 54:1635
\@text@composite 21:76, 21:1070, 21:1075, *1222*
\@text@composite@x 21:76
\@textbottom 49:83, 49:85, 54:515, 54:553, 54:567, 54:576
\@textfloatsheight 54:480, 54:1091, 54:1093, 54:1143, 54:1144, 54:1149, 54:1244, 54:1246, 54:1306, 54:1308, 54:1314, 54:2024
\@textmin 45:285, 45:286, 45:299, 45:300, 54:114, 54:1093, 54:1097, 54:1100, 54:1101, 54:1246, 54:1251, 54:1255, 54:1256, 54:1418, 54:1503, 54:1602, 54:1604, 54:1620, 54:1724, 54:1726, 54:1744, 54:2132, 54:2134, 54:2136
\@textsubscript 45:424, 45:432
\@textsuperscript 45:401, 45:403, 45:404, *1225*
\@texttop 49:83, 49:85, 54:511, 54:534, 54:576
\@tf@r 13:25, 13:26, *1206*
\@tfor 13:25, 20:586, 20:603, 20:755, 32:88, 40:78, 40:97, 41:268, 42:477, 42:505, 42:534, 45:63, 45:132
\@tforloop 13:27, 13:28, 13:30
\@thanks 44:11, 44:34
\@thefnmark 40:487, 40:504, 40:521, 45:400, 45:401, 45:453, 45:458, 45:473, 45:491, 45:509, 45:521, 45:526, 45:537, 45:542, 45:553
\@thefoot 54:126, 54:614, 54:617, 54:644, 54:673, 54:676, 54:703
\@thehead 54:125, 54:614, 54:616, 54:634, 54:673, 54:675, 54:693
\@themargin 54:74, 54:615, 54:617, 54:629, 54:674, 54:676, 54:688
\@themark 49:26, 49:27, 49:37, 49:38, 49:41, 49:54, 49:55, 49:62, 49:63, 49:78, 49:81
\@thirdofthree 21:197, 06:216
\@thm 43:12, 43:18, 43:24, 43:26
\@thmcounter 43:11, 43:17, 43:46
\@thmcountersep 43:10, 43:46
\@title 44:7, 44:31
\@tocrmarg 44:207, 44:230
\@toodeep . 14:253, 39:36, 39:256, 39:267
\@toplist 54:64, 54:386, 54:387, 54:433, 54:434, 54:720, 54:726, 54:736, 54:737, 54:1029, 54:1041, 54:1958, 54:1986
\@topnewpage 54:201, *1209*
\@topnum 45:271, 54:107, 54:1026, 54:1027, 54:1041, 54:1045, 54:1462, 54:1467, 54:1555, 54:1562, 54:1949, 54:1977, 54:2018
\@toproom 45:273, 54:108, 54:1029, 54:1041, 54:1950, 54:1978
\@topsep . . . 39:1, 39:71, 39:73, 39:195
\@topsepadd 39:1, 39:59, 39:61, 39:71, 39:124
\@totalleftmargin 37:529, 37:551, 39:9, 39:53, 39:54, 40:391, 40:412, 41:35, 41:76, 41:81, *802*
\@trivlist 39:48, 39:57, 39:92
\@tryfcolumn 54:782, 54:802, 54:820, 54:836, 54:2205, 54:2222
\@trylist 54:845, 54:848, 54:881, 54:901, 54:923
\@ttfamilyhook 29:429, 29:461, 29:465, 29:492
\@twoclasseserror 50:675, 50:1200
\@twocolumnfalse 54:101, 54:149
\@twocolumntrue 54:208
\@twoloadclasserror 50:959, 50:1094, 50:1195
\@twosidefalse 54:102
\@typein 06:32, 06:33, 06:40, 06:48, *1228*
\@typeset@protect 21:26, 21:32, 21:210, 21:218, 29:638, 37:280, 37:317, 06:102, 06:255, 06:262, 06:264, 57:358, *1220*
\@uclclist 21:1506, 21:1507, 21:1564, 57:571, *1235*
\@undefind 42:57, 42:71, 42:82, 42:91, 42:162, 42:174, 42:237, 42:252, 42:313, 42:371
\@undefined 01:92, 02:582, 01:93, 01:94, 02:650, 02:690, 02:705, 10:182, 10:183, 10:184, 10:186, 10:187, 10:188, 10:190, 10:191, 03:53, 03:62, 01:115, 14:28, 01:123, 16:170, 16:171, 16:172, 16:173, 16:174, 17:61, 04:2, 18:90, 04:15, 04:16, 04:17, 04:29, 04:30, 18:497, 18:546, 18:552, 01:131, 20:61, 20:62, 04:74, 20:130, 20:131, 20:185, 20:186, 04:84, 20:274, 20:275, 20:477, 20:478, 20:479, 20:484, 20:538, 20:551, 20:560, 20:575, 20:630, 01:138, 21:195, 21:197, 04:173, 21:333, 21:335, 21:337, 21:339, 21:341, 21:343, 21:345,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

21:347, 21:349, 21:351, 21:370, 21:372, 21:374, 04:189, 21:458, 04:197, 04:205, 21:700, 21:703, 04:237, 04:238, 04:239, 04:240, 04:241, 04:242, 04:243, 04:244, 04:245, 04:246, 04:247, 04:248, 04:249, 04:250, 04:251, 04:252, 04:253, 04:254, 04:255, 04:261, 22:252, 24:459, 24:499, 24:561, 24:594, 24:658, 24:665, 25:388, 25:411, 25:420, 25:509, 25:510, 25:511, 25:512, 25:513, 25:514, 25:515, 25:516, 25:517, 25:518, 25:529, 25:534, 25:539, 25:603, 25:604, 25:605, 25:606, 25:607, 25:608, 25:609, 25:634, 25:714, 25:716, 25:717, 25:719, 25:720, 25:721, 25:723, 26:552, 26:553, 27:4, 27:5, 27:6, 27:7, 27:8, 27:9, 27:10, 27:11, 27:12, 27:13, 27:14, 27:15, 27:16, 27:17, 27:18, 27:19, 27:20, 28:141, 28:509, 28:561, 01:189, 28:812, 28:919, 01:193, 29:35, 29:58, 29:71, 29:91, 29:104, 29:105, 29:106, 29:107, 29:108, 29:109, 29:110, 29:111, 29:112, 29:113, 29:114, 29:116, 29:117, 29:118, 29:231, 29:232, 29:233, 29:234, 29:235, 29:236, 29:237, 29:238, 29:239, 29:240, 29:242, 29:243, 29:244, 29:304, 29:305, 29:472, 29:490, 29:491, 29:492, 29:534, 29:535, 29:536, 29:537, 29:538, 29:578, 29:579, 29:580, 29:581, 29:582, 29:583, 29:594, 29:716, 30:15, 30:51, 30:58, 30:73, 32:37, 32:38, 32:39, 32:122, 33:159, 33:367, 33:617, 33:620, 33:621, 01:219, 33:651, 33:652, 33:653, 33:656, 33:657, 33:658, 33:659, 33:660, 33:661, 33:666, 33:667, 33:668, 33:669, 33:674, 33:675, 33:676, 33:677, 33:680, 33:681, 33:682, 33:684, 33:685, 33:690, 33:691, 33:692, 33:693, 33:694, 33:695, 33:696, 33:697, 33:698, 33:699, 33:700, 33:701, 33:702, 33:703, 33:704, 33:705, 33:707, 33:708, 33:710, 33:711, 33:712, 33:713, 33:714, 33:715, 33:716, 33:717, 33:718, 33:719, 33:721, 33:722, 33:723, 33:724, 33:725, 33:726, 33:727, 33:728, 33:729, 33:731, 33:732, 33:733, 33:734, 33:735, 33:736, 33:737, 33:738, 33:739, 33:740, 33:741, 33:742, 33:743, 33:744, 33:745, 33:746, 33:747, 33:748, 33:749, 33:750, 33:751, 33:752, 33:753, 33:754, 33:755, 33:756, 33:757, 33:762, 33:763, 33:765, 33:766, 33:767, 33:768, 33:769, 33:770, 33:771, 33:772, 33:774, 33:775, 33:777, 33:778, 33:780, 33:781, 33:782, 33:783, 33:785, 33:786, 33:787, 33:789, 33:791, 33:792, 33:793, 33:794, 33:795, 33:796, 33:797, 33:799, 33:800, 33:801, 33:802, 33:803, 33:804, 33:805, 33:806, 33:807, 05:4, 01:226, 05:13, 05:14, 05:15, 05:16, 35:161, 35:162, 35:163, 35:183, 35:199, 35:200, 36:267, 36:268, 36:270, 36:271, 36:272, 36:274, 36:275, 36:276, 36:278, 36:279, 37:176, 37:228, 37:229, 37:230, 37:231, 37:306, 37:385, 37:405, 37:406, 37:407, 37:408, 37:515, 05:150, 05:151, 05:152, 05:153, 05:154, 37:577, 05:155, 37:595, 37:596, 37:597, 37:598, 37:631, 37:643, 37:692, 05:172, 05:173, 05:174, 05:175, 38:236, 38:246, 38:247, 38:283, 38:286, 38:329, 38:342, 05:195, 38:436, 05:196, 40:21, 40:169, 40:233, 40:314, 06:34, 40:540, 40:570, 42:18, 42:466, 42:467, 44:197, 44:245, 45:5, 06:235, 45:429, 45:448, 45:562, 47:30, 47:53, 06:319, 06:320, 50:4, 50:25, 06:351, 50:146, 50:147, 50:148, 50:209, 06:379, 06:383, 06:384, 50:416, 06:401, 50:526, 50:576, 06:415, 50:981, 50:1019, 50:1021, 50:1027, 50:1084, 50:1099, 50:1100, 50:1115, 06:462, 06:463, 50:1254, 06:464, 06:465, 06:466, 06:467, 06:468, 50:1332, 50:1335, 50:1385, 50:1464, 50:1467, 50:1477, 50:1478, 50:1479, 50:1480, 50:1481, 50:1482, 50:1483, 01:310, 50:1554, 50:1557, 50:1573, 50:1593, 50:1600, 01:311, 06:504, 06:505, 06:506, 06:507, 06:508, 06:509, 06:540, 06:541, 06:542, 52:18, 52:19, 52:20, 52:21, 52:147, 52:189, 52:190, 52:197, 52:198, 52:199, 06:560, 06:561, 52:320, 52:341, 52:355, 52:359, 52:360, 52:492, 52:498, 52:499, 52:500,

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

53:438, 53:439, 53:440, 53:441, 53:442, 53:444, 53:445, 53:446, 53:453, 53:454, 53:456, 53:458, 53:459, 53:460, 53:463, 53:486, 54:36, 06:668, 06:669, 06:670, 06:671, 06:672, 06:673, 06:675, 06:676, 06:677, 54:370, 54:371, 06:732, 06:747, 06:754, 54:1928, 54:1929, 54:1930, 54:1931, 06:828, 06:829, 06:830, 06:839, 57:10, 57:18, 57:39, 57:54, 57:73, 57:82, 57:89, 57:98, 57:108, 57:160, 57:161, 57:270, 57:283, 57:296, 57:316, 57:341, 57:351, 57:352, 57:353, 57:382, 57:384, 57:423, 57:424, 57:443, 57:444, 57:445, 57:446, 57:447, 57:448, 57:449, 57:450, 57:451, 57:468, 57:484, 57:485, 57:486, 57:529, 57:530, 57:658, 57:693, 57:694, 57:695, 57:696, 57:697, 02:65, 02:81, 02:105, 02:106, 02:121, 02:122, 02:127, 02:136, 02:149, 02:184, 02:189, 02:222, 02:223, 02:246, 02:256, 01:52, 01:53, 02:291, 02:353, 02:363, 02:365, 02:475, 02:476, 02:486, 02:487, *1247*
\@undefinedfonterror *1210*
\@unexpandable@protect 20:211, 41:264, 06:232, 06:267, 06:273, 06:278, *1219*
\@unknownoptionerror 50:1110, 50:1170, 50:1183
\@unknownversion *1199*
\@unprocessedoptions 50:625, 50:734, 50:981, 50:1020, 50:1021, 50:1081, 50:1085, 50:1185, 51:85, *1233*
\@unused 14:15, 14:32, 14:59, 06:10, 06:17, 50:1578, 02:307, *1257*
\@unusedoptionlist 20:18, 20:20, 20:88, 20:90, 20:145, 20:147, 50:12, 50:518, 50:519, 50:529, 50:530, 50:637, 50:645, 51:127, 51:128, 51:134, 51:161, *1032*
\@upline 42:297, 42:298, 42:304
\@upordown 42:195, 42:196, 42:204, 42:225, 42:273
\@upvector 42:268, 42:304
\@use@ption . 50:549, 50:565, 50:583, 50:599, 50:601, 50:618, 50:620, 50:630
\@use@text@encoding 21:150, 33:16, 33:1138
\@vbsphack 18:220
\@verb 37:725, 37:734, 37:737
\@verbatim 37:523, 37:569, 37:584, 37:593
\@verbvisiblespacebox . . . 37:583, 37:598, 37:610, 37:611, 37:624, 37:625
\@vereq 30:464, 30:465
\@viiipt 24:742
\@viipt 24:741
\@vipt 24:740
\@vline 42:166, 42:178, 42:297
\@vobeyspaces 37:499, 37:569, 37:586, 37:662, 37:676, 37:690, 37:737
\@vobeytabs 37:499
\@vpt 24:739
\@vspace 18:377
\@vspace@calcify 18:80, 18:102, 18:242, 18:257, 18:384, 18:389, 18:399, 18:407, 37:435, 38:404, 41:62, 41:224, 42:148
\@vspacer 18:377
\@vtryfc 54:851, 54:859
\@vvector 42:243, 42:258, 42:268
\@warning 14:215
\@wckptelt 20:440, 20:443
\@whiledim 13:7, 42:123, 42:203
\@whilenoop 13:3, *1227*
\@whilenum 13:3, 41:244, 42:104, 42:116, 42:336, 42:338, 42:361, 42:364, 42:393, 42:395, 42:416, 42:421, 42:730, 42:786
\@whilesw . . . 13:10, 54:266, 54:396, 54:405, 54:443, 54:453, 54:2275, 54:2315
\@whileswnoop 13:10, *1227*
\@wholewidth 40:199, 40:201, 40:202, 40:204, 40:206, 40:207, 40:208, 40:209, 42:2, 42:126, 42:129, 42:131, 42:299, 42:302, 42:350, 42:359, 42:406, 42:413, 42:564, 42:577, 42:589, 42:599, 42:678, 42:679, 42:727, 42:783
\@width 18:528, 21:289, 21:294, 26:192, 30:625, 06:26, 40:204, 40:206, 40:282, 40:289, 40:550, 40:603, 40:610, 41:188, 41:219, 41:347, 41:366, 42:227, 42:299, 42:302, 42:325, 42:333, 42:350, 42:359, 42:384, 42:392, 42:406, 42:413, 42:564, 42:577, 42:727, 42:783, 45:395, 54:1855, 54:2264, 54:2298, 02:505, *1223*
\@wrglossary 46:25, 46:30, *1219*
\@wrindex 46:8, 46:13, *1219*
\@writeckpt 20:330, 20:386, 20:425, 20:437
\@writefile 20:32, 20:102, 20:160, 37:215, 44:183, *766*
\@writesetup 54:594

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@wrong@font@char 21:161, 24:562, 24:596, 24:609, *1230*
\@wtryfc 54:861, 54:871
\@x@protect 06:105, 06:254
\@x@sf 45:531, 45:533
\@xDeclareMathDelimiter 28:984, 28:1040
\@xaddvskip 18:220, 18:245, 18:260, 18:277
\@xarg 42:163, 42:166, 42:175, 42:178, 42:185, 42:189, 42:190, 42:226, 42:228, 42:238, 42:239, 42:243, 42:253, 42:254, 42:258, 42:266, 42:274, 42:663
\@xargarraycr . . 41:205, 41:214, 41:218
\@xargdef 06:80
\@xarraycr 41:202, 41:203
\@xbitor 54:15, 54:17
\@xcentercr 37:418, 37:425, 37:429
\@xdblarg 06:798
\@xdblfloat 45:264, *1218*
\@xdim . 42:84, 42:93, 42:105, 42:107, 42:117, 42:119, 42:667, 42:731, 42:732, 42:733, 42:734, 42:740, 42:787, 42:788, 42:789, 42:790, 42:796
\@xeqncr 38:388
\@xexnoop 41:238, 41:248
\@xexpast 41:239, 41:240
\@xfloat 45:28, 45:29, 45:34, 45:266, *1219*
\@xfootnote 45:452, 45:455, *1196*
\@xfootnotemark . 45:519, 45:523, *1196*
\@xfootnotenext 45:536, 45:539
\@xfootnotetext *1196*
\@xhline 41:360, 41:361
\@xifnch 06:786, 06:796
\@xiipt . 24:746, 30:171, 30:173, 30:174
\@xipt 24:745, 30:170
\@xivpt 24:747, 30:172, 30:174
\@xmpar 45:324, 45:325
\@xnewline 18:61, 18:62, 18:73, 18:74, 18:94
\@xnext 54:10, 54:11
\@xnthm 43:5, 43:6
\@xobeysp 18:467, 18:492, 37:504, 37:514, 37:518, 37:607, 37:611, 37:621, 37:625, 37:638, *438*
\@xobeytab 18:488, 37:506, 37:638
\@xprocess@ptions 50:540, 50:557, 50:559, 50:575, 50:577, 50:591
\@xpt . . . 24:744, 30:169, 30:172, 30:173
\@xsect 44:86, 44:87, 44:123
\@xtabcr 41:56, 41:57
\@xtabularcr 41:209, 41:210
\@xthm 43:32, 43:38, 43:41
\@xtryfc 54:848, 54:876
\@xtypein 06:33, 06:35, 06:42
\@xverbatim 37:518, 37:569
\@xviipt 24:748, 30:173, 30:175
\@xxDeclareMathDelimiter 28:969, 28:973
\@xxpt 24:749, 30:174, 30:175
\@xxvpt 24:750, 30:175
\@xxxii 12:2, 21:425, 21:427, 45:89, 45:158, 54:883, 54:884, 54:903, 54:904, 54:939, 54:940, 54:962, 54:963, 54:2041, *1230*
\@xympar 45:328, 45:332, 45:378
\@yarg 42:163, 42:167, 42:175, 42:179, 42:185, 42:186, 42:195, 42:238, 42:244, 42:253, 42:259, 42:268, 42:270, 42:297, 42:663
\@yargarraycr . . 41:206, 41:216, 41:220
\@yargd@f 06:107
\@yargdef 06:84, 06:94, 06:107, 06:123, *1232*
\@ydim . 42:85, 42:94, 42:105, 42:108, 42:117, 42:119, 42:667, 42:735, 42:736, 42:737, 42:738, 42:739, 42:791, 42:792, 42:793, 42:794, 42:795
\@yeqncr 38:388
\@ympar 45:324, 45:329
\@ynthm 43:5, 43:14
\@ythm 43:32, 43:38, 43:41
\@ytryfc 54:894, 54:913, 54:917
\@yyarg 42:185, 42:186, 42:187, 42:190, 42:274, 42:663
\@ztryfc 54:922, 54:933
\@@end *220*
\accent@spacefactor 21:70, 21:73, 21:74, *1238*
\active@math@prime 38:253, 38:254, 54:592, *1238*
\add@accent 21:65, 21:67, *1231*
\add@percent@to@temptokena 37:202, 37:218, 37:220, 37:229, *767*
\add@unicode@accent . 21:1039, 21:1053
\addto@hook 24:152, 24:154, 24:738, 28:438, 28:574, 28:578, 28:595, 28:719, 28:725, 28:733, 28:749, 28:752, 28:755, 28:1188, 28:1195, 28:1198, *1194*
\AddToHook *303*
\AddToHookNext *303*
\AddToHookNextWithArguments . . . *303*
\AddToHookWithArguments *303*
\alloc@ 04:22, 04:26, 04:38, 24:14, 02:90,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

02:91, 02:92, 02:93, 02:94, 02:95, 02:96, 02:97, 02:98, 02:99, 02:226, *1253*
\alpha@elt 28:45, 28:442, 28:669, 28:771, 28:1187, 28:1188
\alpha@list . . 28:41, 28:43, 28:451, 28:657, 28:669, 28:714, 28:769, 28:770, 28:1183, 28:1189, 28:1190
\apptocmd *304*
\atveryend@DEPRECATED 52:580, 52:582
\best@size 26:438, 26:462, 26:468, 26:474
\bf@def@ult 29:397
\bfdef@ult 29:200, 29:271, 29:272, 29:273, 29:314, 29:315, 29:316, 29:401, 29:442, *643*
\bfdefault@previous 29:267, 29:270, 29:310, 29:313, 30:113, 30:123, *643*
\bfseries@.. . . . . *638*
\bfseries@previous *1254*
\bfseries@rm 29:105, 29:128, 29:207, 29:210, 29:231, 29:258, 29:271, 29:314, 29:319, 29:355, *638*
\bfseries@rm@kernel 29:108, 29:131, 29:207, 29:234, *641*
\bfseries@sf 29:106, 29:128, 29:211, 29:214, 29:232, 29:259, 29:272, 29:315, 29:320, 29:356, *641*
\bfseries@sf@kernel 29:109, 29:132, 29:211, 29:235
\bfseries@tt 29:107, 29:128, 29:215, 29:218, 29:233, 29:260, 29:273, 29:316, 29:321, 29:357, *641*
\bfseries@tt@kernel 29:110, 29:133, 29:215, 29:236
\bm@b 40:37
\bm@c 40:37, 40:50, 40:251, 40:346
\bm@l 40:37
\bm@r 40:37
\bm@s 40:37
\bm@t 40:37
\botmark *948*
\bx@ *1144*
\bx@A 54:30, 54:57
\bx@AA 54:40
\bx@B 54:30, 54:57
\bx@BB 54:40
\bx@C 54:30, 54:57
\bx@CC 54:40
\bx@D 54:30, 54:57
\bx@DD 54:40
\bx@E 54:30, 54:57
\bx@EE 54:40
\bx@F 54:31, 54:58
\bx@FF 54:41
\bx@G 54:31, 54:58
\bx@GG 54:41
\bx@H 54:31, 54:58
\bx@HH 54:41
\bx@I 54:31, 54:58
\bx@II 54:41
\bx@J 54:31, 54:58
\bx@JJ 54:41
\bx@K 54:32, 54:59
\bx@KK 54:42
\bx@L 54:32, 54:59
\bx@LL 54:42
\bx@M 54:32, 54:59
\bx@MM 54:42
\bx@N 54:32, 54:59
\bx@NN 54:42
\bx@O 54:33, 54:60
\bx@OO 54:43
\bx@P 54:33, 54:60
\bx@PP 54:43
\bx@Q 54:33, 54:60
\bx@QQ 54:43
\bx@R 54:33, 54:60
\bx@RR 54:43
\bx@S 54:38
\bx@SS 54:44
\bx@T 54:38
\bx@TT 54:44
\bx@U 54:38
\bx@UU 54:44
\bx@V 54:38
\bx@VV 54:44
\bx@W 54:39
\bx@WW 54:45
\bx@X 54:39
\bx@XX 54:45
\bx@Y 54:39
\bx@YY 54:45
\bx@Z 54:39
\bx@ZZ 54:25, 54:45, 54:55
\c@bottomnumber 45:269, 45:274, 54:2330
\c@dbltopnumber 45:268, 45:283, 45:297, 54:2337
\c@enumi 39:251
\c@enumii 39:251
\c@enumiv 39:251, *1196*
\c@equation 38:349, 38:383, 38:543
\c@errorcontextlines . . . 14:212, *1200*
\c@footnote 44:12, 45:397, 45:525
\c@localmathalphabets 28:137, 28:150, 28:282, 28:315, *607*
\c@mpfootnote 40:454, 45:399
\c@ncel 30:468, 30:469
\c@page 34:3, 34:6, 34:7, 48:403, 48:462, 54:140, 54:1822

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\c@secnumdepth 44:56, 44:71, 44:81, 44:140
\c@tocdepth 44:140, 44:205, 44:228
\c@topnumber . . 45:267, 45:271, 54:2326
\c@totalnumber 45:270, 45:276, 54:2333
\c@totalpages 53:348, 53:441
\calculate@math@sizes 24:645, 26:219
\catcodetable@atletter . 04:93, 04:246
\catcodetable@initex . . 04:93, 04:243
\catcodetable@latex . . . 04:93, 04:245
\catcodetable@string . . 04:93, 04:244
\cdp@elt 24:96, 24:116, 24:127, 24:128, 24:149, 24:152, 24:154, 28:352, 28:468, 28:543, 28:617, 28:698, 57:472, 57:473
\cdp@list 24:98, 24:114, 24:128, 24:156, 24:157, 28:370, 28:470, 28:545, 28:619, 28:700, 57:473
\cf@encoding 21:34, 21:41, 21:44, 21:51, 21:154, 24:256, 24:266, 24:276, 24:326
\ch@ck 50:1259, 50:1278, 50:1390, 50:1411, 50:1503, 02:206, 02:207, 02:208, 02:209, 02:210, 02:236, 02:248, 02:249, 02:250, 02:251, 02:279, 02:281, 02:293, 02:294, 02:295, 02:296, 02:302
\char@if@alph 06:806
\chardef@text@cmd 21:3
\check@command 06:187, 06:189
\check@icl 32:9, 32:44, 32:49, 32:55, 32:63, 32:70, 32:72
\check@icr 32:9, 32:44, 32:50, 32:56, 32:64, 32:73, 32:78, *1233*
\check@mathfonts 19:5, 21:302, 21:328, 21:360, 21:1254, 24:359, 24:363, 24:370, 24:372, 26:250, 28:320, 28:403, 33:672, *613*
\check@nocorr *1216*
\check@nocorr@ 32:46, *1235*
\check@range 26:379, 26:380
\check@single 26:378, 26:400
\cl@@ckpt 20:440, 22:35
\cl@page 34:4
\col@number 54:97, 54:150, 54:210, 54:222, *1209*
\color@begingroup 24:668, 24:729, 38:104, 38:134, 40:29, 40:110, 40:215, 40:442, 40:488, 40:505, 40:522, 41:47, 41:51, 45:475, 45:493, 45:511, 54:495, *1206*
\color@endbox 40:110, 45:253, 45:348, 45:366, 54:226, 54:635, 54:645, 54:694, 54:704
\color@endgroup 24:673, 24:735, 38:104, 38:134, 40:29, 40:119, 40:134, 40:173, 40:194, 40:217, 40:470, 40:492, 40:509, 40:525, 41:49, 45:479, 45:497, 45:514, 54:499, *1206*
\color@hbox 40:110, 54:632, 54:642, 54:691, 54:701, *1220*
\color@setgroup 40:110, 40:173, 40:192, *1206*
\color@vbox . 40:110, 45:96, 45:165, 45:339, 45:357, 45:381, 54:217, *1221*
\conditionally@traceoff 14:291, 07:285, 07:297, *398*
\conditionally@traceon 14:291, 07:321
\contionally@traceon *398*
\copy@kernel@robust@command 06:588, 06:675
\count@ 03:14, 03:15, 03:16, 03:17, 03:18, 03:20, 01:163, 01:164, 01:165, 24:703, 24:709, 24:711, 01:170, 26:22, 26:302, 26:304, 26:326, 26:327, 28:435, 28:437, 28:441, 28:800, 28:801, 28:802, 28:848, 28:849, 28:850, 28:909, 28:910, 28:911, 28:957, 28:958, 28:959, 28:997, 28:998, 28:999, 28:1005, 28:1006, 28:1007, 28:1051, 28:1052, 28:1053, 28:1059, 28:1060, 28:1061, 28:1120, 28:1121, 28:1122, 28:1128, 28:1129, 28:1130, 32:115, 32:118, 42:729, 42:730, 42:731, 42:734, 42:735, 42:738, 42:742, 06:169, 42:785, 42:786, 42:787, 42:790, 42:791, 42:794, 06:173, 42:798, 06:287, 06:312, 50:1302, 50:1304, 50:1305, 50:1306, 50:1435, 50:1437, 50:1438, 50:1439, 50:1524, 50:1526, 50:1527, 50:1528, 57:239, 57:240, 57:247, 57:249, 57:534, 57:535, 57:542, 57:544, 02:41, 02:191, 02:192, 02:197, 02:199, 02:205, 02:206, 02:207, 02:208, 02:209, 02:210, 02:211, 01:50, 02:505, 02:506, *1168*
\counterwithin@s 22:161, 22:162, 22:180
\counterwithin@x 22:161, 22:164, 22:181
\counterwithout@s 22:150, 22:151, 22:177
\counterwithout@x 22:150, 22:153, 22:178
\curr@fontshape 21:180, 24:88, 24:388, 24:396, 24:400, 24:402, 24:544, 24:550, 24:553, 24:562, 24:569, 24:571, 24:579, 24:585, 24:588, 24:596,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

24:603, 24:605, 25:454, 26:92, 26:100, 26:142, 26:169, 26:477, 26:497, 26:529, 26:545, 26:560, 28:374, 28:379, 29:556, 29:565, *1194*
\curr@math@size 24:377, 26:256, 26:262, 26:267, 26:284
\declare@command@copy *1253*
\declare@commandcopy 06:476, 06:480, 06:485, 06:488, 06:507, *98*
\declare@commandcopy@do 06:488, 06:534, 06:535
\declare@commandcopy@let . 06:495, 06:499, 06:509, 06:610, 06:611, *99*
\declare@environmentcopy 06:517, 06:521, 06:526, 06:529
\declare@file@substitution 52:247, 52:567, 53:517, *1040*
\declare@robustcommand 06:233
\DeclareEncodingSubset@aux 33:41, 33:43, 33:59
\DeclareFontEncoding@ 24:122, 24:124, 24:139, 57:383, 57:403, 57:469, *1247*
\DeclareFontEncoding@saved 57:383, 57:403, 57:485
\DeclareFontShape@ 24:21, 24:22
\DeclareRobustCommand *312*
\DeclareSymbolFont@m@dropped 28:460, 28:465, 28:508, 28:509
\DeclareSymbolFontAlphabet@ 28:1175, 28:1178
\DeclareUnicodeAccent@ 21:1046, 21:1048, 21:1052
\def *322*
\default@ds 50:505, 50:536, 50:600, 50:619, 50:1108, 50:1110, *1198*
\default@family 24:129, 24:161, 24:512, 24:526, 24:529, 24:554, 24:589, 57:474, *1193*
\default@M 24:136, 24:176, 24:179, 24:183, 57:481, *1198*
\default@mextra 27:10, 27:89
\default@series 24:129, 24:162, 24:513, 24:527, 24:530, 24:551, 24:586, 57:474
\default@shape 24:130, 24:163, 24:514, 24:528, 24:531, 24:549, 24:584, 57:475
\default@T 24:170, 24:173, 24:183, 24:272, *1198*
\define@mathalphabet 27:18, 27:131, *1193*
\define@mathgroup . 27:19, 27:135, *1193*
\define@newfont 24:380, 24:389
\delayed@f@adjustment 24:290, 25:395, 25:396, 25:397, 25:399, 25:400, 25:411, 25:616, 25:617, 25:619, 25:620, 26:122, 26:126, 26:134, 26:138, 29:662, *657*
\delayed@merge@font@series 25:396, 25:460, 25:475, 25:514, 26:133, 26:136
\delayed@merge@font@shape 25:617, 25:666, 25:721, 26:132, 26:135
\development@branch@name 03:11, 03:39, 03:53, 03:54, 03:55, 03:62, 03:63, 03:64
\dimen@ 14:28, 14:29, 18:396, 18:401, 18:430, 18:435, 21:429, 21:430, 21:432, 21:433, 21:796, 21:797, 24:214, 24:216, 24:222, 24:235, 24:238, 24:242, 24:648, 24:649, 24:650, 24:654, 26:451, 26:452, 26:453, 26:454, 26:458, 33:102, 33:104, 33:928, 33:930, 38:72, 38:73, 38:199, 38:200, 38:201, 38:202, 40:587, 40:590, 41:176, 41:177, 54:512, 54:514, 54:535, 54:537, 02:41, 02:502, 02:503, 02:539, 02:540, 02:542, 02:544, *1232*
\dimen@i 02:41
\dimen@ii 24:218, 24:223, 02:41
\disable@package@load 52:477, 53:475, *1055*
\displ@y 38:204, 38:208, 38:209
\do@add@percent@to@temptokena 37:206, 37:212, 37:230
\do@emfont@update 29:557, 29:561, 29:581
\do@noligs 37:739, 37:744
\do@subst@correction 24:84, 26:482, 26:537
\document@default@language 20:51, 20:52, 20:121, 20:122, 54:601, 57:286
\document@select@group 28:139, 28:145, 28:387, *607*
\dont@add@percent@to@temptokena 37:205, 37:207, 37:231
\dorestore@version . . . 28:114, 28:119
\ds@ 50:538, 50:1112, *1198*
\dt@pfalse 38:205
\dt@ptrue 38:204
\e@alloc 04:15, 04:49, 04:79, 04:89, 04:178, 04:182, 04:186, 04:194, 04:202, 04:210, 57:12, 57:47, 02:51, 02:52, 02:53, 02:55, 02:56, 02:63, 02:64, 02:66, 02:68, 02:79, 02:82, 02:84, 02:138, 02:230, *1253*
\e@alloc@attribute@count 04:66, 04:74, 04:75, 04:76, 04:80, 04:237

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\e@alloc@bytecode@count . . . 04:70, 04:197, 04:198, 04:199, 04:203, 04:253
\e@alloc@ccodetable@count 04:67, 04:84, 04:85, 04:86, 04:90, 04:241
\e@alloc@chardef 04:48, 04:178, 04:194, 04:202, 04:210, 57:12, 02:60, 02:102, 02:211, 02:212
\e@alloc@intercharclass@top . . 57:35
\e@alloc@luachunk@count . . . 04:71, 04:205, 04:206, 04:207, 04:211, 04:255
\e@alloc@luafunction@count 04:68, 04:173, 04:174, 04:175, 04:179, 04:183, 04:187, 04:247, 04:249
\e@alloc@top 04:47, 04:80, 04:179, 04:183, 04:187, 04:195, 04:203, 04:211, 57:12, 02:55, 02:63, 02:102, 02:188, 02:259
\e@alloc@whatsit@count 04:69, 04:189, 04:190, 04:191, 04:195, 04:251
\e@ch@ck . . 04:51, 04:55, 02:142, 02:152
\e@insert@top 02:257, 02:259, 02:276, 02:291
\e@mathgroup@top 28:56, 28:149, 28:150, 28:187, 28:217, 02:79, 02:124, *607*
\em@currfont 29:556, 29:567, *654*
\em@force 29:565, 29:570, 29:573, 29:583, *654*
\emfontdeclare@clist 29:546, 29:548, 29:552, 29:557, 29:562, 29:568, 29:579, *653*
\empty@sfcnt 26:490, 26:491, 26:492, 26:506, 26:511, 26:563
\ENC@cmd *1213*
\enc@update 24:257, 24:259, 24:275, 24:278, 26:147, 26:173, *1213*
\end@dblfloat 45:205
\end@float 45:189, 45:227, 45:243, 45:383, *1241*
\endlinechar *322*
\enlargethispage *952*
\err@rel@i . 27:12, 27:99, 27:132, 27:136
\error@fontshape . . 24:507, 24:522, 24:547, 24:582, 26:107, 26:527, 28:373
\escapechar *315*
\et@xmaxfam . 04:22, 04:26, 04:30, 04:38
\et@xmaxregs 04:29, 04:31, 04:32, 04:33, 04:34, 04:35, 04:36, 04:37
\every@math@size 24:78, 26:235, 26:247, *1217*
\every@size *1223*
\execute@size@function 26:362, 26:390, 26:404, 26:421
\expand@font@defaults 29:37, 29:148, 29:256, 29:279, 29:309, 29:330, 29:354, 29:365, 29:396, 29:397, 29:436, 29:438, 29:470, 29:472, 29:501, 33:26, 33:83, *221*
expand@font@defaults 29:420
\expandafter *1055*
\external@font 26:84, 26:87, 26:98, 26:102, 26:104, 26:391, 26:405, 26:467, 26:501, 26:569, 26:571, 26:573
\extra@def 27:9, 27:84, *1193*
\extract@alph@from@version 24:622, 24:628, 28:162, 28:193, 28:223, 28:255, *1196*
\extract@default@composite 21:1062, 21:1069
\extract@default@composite@a 21:1071, 21:1075
\extract@default@composite@b 21:1073, 21:1077
\extract@font 24:403, 26:81
\extract@fontinfo 26:358, 26:365
\extract@rangefontinfo 26:375, 26:382, 26:401, 26:434
\extract@sizefn 26:350, 26:372
\f@... . . . . *1248*
\f@baselineskip 21:870, 21:1220, 24:317, 24:324, 24:533, 26:121, 26:167, 26:182, 26:186, 26:201, 26:215, 26:226, 26:240
\f@depth 45:291, 54:347
\f@encoding 21:178, 04:260, 04:275, 04:283, 24:251, 24:270, 24:273, 24:274, 24:276, 24:326, 24:383, 24:388, 24:407, 24:409, 24:411, 24:416, 24:418, 24:449, 24:511, 24:543, 24:578, 25:439, 25:443, 25:652, 25:656, 26:91, 26:128, 26:307, 26:517, 28:358, *1196*
\f@family 24:279, 24:287, 24:299, 24:309, 24:320, 24:384, 24:388, 24:407, 24:409, 24:411, 24:416, 24:418, 24:450, 24:529, 24:554, 24:589, 25:439, 25:443, 25:652, 25:656, 26:91, 26:128, 28:358, 28:389, 29:151, 29:178, 29:179, 29:258, 29:259, 29:260, 29:281, 29:282, 29:283, 29:319, 29:320, 29:321, 29:339, 29:340, 29:341, 29:355, 29:356, 29:357, 29:366, 29:367, 29:368, 29:504, 29:517, 29:659, 29:677, 29:693, 33:23, 33:27, 33:29, 33:31, 33:63, 33:66, 33:80, 33:84, 33:86,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

33:88, 33:93, 33:100, 33:894, 33:897, 33:911, 33:920, 33:926, 33:1141, *651*
\f@linespread 24:320, 26:120, 26:166, 26:183, 26:184, 26:187, 26:195, 26:198, 26:209, 26:212, *1213*
\f@series 19:14, 24:279, 24:310, 24:321, 24:385, 24:388, 24:530, 24:551, 24:586, 25:400, 25:410, 25:419, 25:429, 25:463, 25:482, 25:483, 25:652, 25:656, 26:125, 26:128, 26:131, 28:390, 29:144, 29:160, 29:162, 29:166, 29:167, 29:168, 29:189, 29:195, 29:198, 29:200, 29:222, 29:520, 29:524, 29:660, 29:678, 29:694, 29:739, 33:7, 33:613, *560*
\f@series@saved 26:125, 26:131
\f@shape 24:279, 24:289, 24:301, 24:311, 24:322, 24:386, 24:388, 24:531, 24:549, 24:584, 25:439, 25:443, 25:620, 25:627, 25:633, 25:643, 25:650, 25:654, 25:657, 25:660, 25:669, 25:676, 25:678, 25:713, 26:124, 26:128, 26:130, 28:391, 29:661, 29:679, 29:695, 29:740, *576*
\f@shape@saved 26:124, 26:130
\f@size 21:180, 21:869, 21:1219, 24:88, 24:317, 24:323, 24:387, 24:532, 24:571, 24:605, 24:647, 24:648, 24:651, 24:652, 26:121, 26:142, 26:167, 26:169, 26:180, 26:200, 26:215, 26:218, 26:221, 26:226, 26:233, 26:240, 26:252, 26:255, 26:261, 26:267, 26:284, 26:285, 26:288, 26:293, 26:359, 26:366, 26:385, 26:387, 26:402, 26:453, 26:455, 26:457, 26:473, 26:474, 26:479, 26:493, 26:505, 26:510, 26:522, 26:530, 26:535, 26:561, 26:575, 29:556, 29:565, 33:102, 33:928
\f@user@size 26:473, 26:478, 26:522, 26:535
\f@warn@break *1215*
\famdef@ult 29:444
\filec@ntents 50:1209, 50:1212, 50:1215, 50:1226, 50:1251, 50:1344, 50:1347, 50:1350, 50:1361, 50:1382, 50:1475, 50:1497, 50:1585, 50:1808, *1225*
\filec@ntents@checkdir 50:1232, 50:1234, 50:1252, 50:1367, 50:1369, 50:1383, 50:1482
\filec@ntents@force 50:1228, 50:1363, 50:1478
\filec@ntents@noheader 50:1230, 50:1365, 50:1480
\filec@ntents@nosearch 50:1231, 50:1366, 50:1481
\filec@ntents@nowarn 50:1236
\filec@ntents@opt 50:1212, 50:1215, 50:1217, 50:1347, 50:1350, 50:1352, 50:1477
\filec@ntents@OPTION *1010*
\filec@ntents@overwrite 50:1229, 50:1364, 50:1479
\filec@ntents@warning 50:1237, 50:1239, 50:1284
\filec@ntents@where 50:1233, 50:1235, 50:1264, 50:1368, 50:1370, 50:1395, 50:1483
\filename@area ... 20:642, 20:657, 20:667, 20:699, 20:700, 20:704, 20:729, 20:732, 20:757, 20:773, 20:775, 01:230, 01:236, 01:243, 01:249, 01:256, 01:262, 01:269, 50:941
\filename@base 20:642, 20:657, 20:667, 20:699, 20:701, 20:704, 20:729, 20:732, 20:764, 20:773, 01:279, 01:286, 01:299, 50:942
\filename@dot 01:297, 01:303
\filename@dots . 01:281, 01:283, 01:288
\filename@ext 20:643, 20:658, 20:668, 20:695, 20:696, 20:699, 20:702, 20:704, 20:725, 20:726, 20:729, 20:732, 20:765, 01:278, 01:285, 01:295, 01:297, 50:943, *1046*
\filename@parse 01:94, 20:640, 20:655, 20:665, 20:694, 20:724, 20:762, 01:226, 50:940
\filename@path 01:231, 01:232, 01:237, 01:244, 01:245, 01:250, 01:257, 01:258, 01:263
\filename@simple 01:234, 01:247, 01:260, 01:270, 01:274, 01:276, 01:291, 01:293
\finish@module@release .. 03:89, 03:93
\finph@nt 38:104, 38:106, 38:110, 38:111, 38:119, 38:120
\finsm@sh ... 38:134, 38:141, 38:147, 38:153, 38:154, 38:158, 38:159, *788*
\fix@penalty 32:101
\fixed@sfcnt ... 26:565, 26:566, 26:567
\fl@ShowFloat 54:1908, 54:1914, 54:1929
\fl@trace 54:242, 54:269, 54:325, 54:353, 54:360, 54:381, 54:428, 54:476, 54:529, 54:544, 54:545, 54:546, 54:547, 54:558,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

54:559, 54:560, 54:561, 54:562, 54:572, 54:785, 54:804, 54:823, 54:841, 54:843, 54:982, 54:986, 54:998, 54:999, 54:1000, 54:1001, 54:1007, 54:1010, 54:1018, 54:1022, 54:1033, 54:1038, 54:1043, 54:1044, 54:1045, 54:1046, 54:1053, 54:1056, 54:1064, 54:1075, 54:1081, 54:1086, 54:1091, 54:1097, 54:1098, 54:1103, 54:1108, 54:1109, 54:1110, 54:1118, 54:1122, 54:1127, 54:1131, 54:1136, 54:1147, 54:1148, 54:1150, 54:1168, 54:1177, 54:1183, 54:1192, 54:1195, 54:1201, 54:1211, 54:1215, 54:1225, 54:1231, 54:1237, 54:1243, 54:1250, 54:1252, 54:1258, 54:1263, 54:1265, 54:1267, 54:1275, 54:1280, 54:1286, 54:1291, 54:1297, 54:1311, 54:1312, 54:1315, 54:1336, 54:1345, 54:1351, 54:1360, 54:1363, 54:1370, 54:1380, 54:1384, 54:1396, 54:1402, 54:1407, 54:1412, 54:1416, 54:1420, 54:1421, 54:1428, 54:1433, 54:1437, 54:1444, 54:1453, 54:1457, 54:1461, 54:1462, 54:1466, 54:1467, 54:1477, 54:1483, 54:1489, 54:1495, 54:1499, 54:1505, 54:1507, 54:1515, 54:1520, 54:1525, 54:1533, 54:1542, 54:1547, 54:1552, 54:1554, 54:1559, 54:1561, 54:1572, 54:1578, 54:1588, 54:1594, 54:1598, 54:1599, 54:1604, 54:1605, 54:1611, 54:1614, 54:1615, 54:1616, 54:1623, 54:1624, 54:1625, 54:1633, 54:1638, 54:1650, 54:1651, 54:1658, 54:1661, 54:1669, 54:1673, 54:1677, 54:1678, 54:1682, 54:1683, 54:1693, 54:1699, 54:1709, 54:1715, 54:1719, 54:1720, 54:1726, 54:1727, 54:1734, 54:1737, 54:1738, 54:1739, 54:1747, 54:1748, 54:1749, 54:1758, 54:1763, 54:1776, 54:1778, 54:1785, 54:1788, 54:1797, 54:1801, 54:1805, 54:1806, 54:1810, 54:1811, 54:1863, 54:1868, 54:1874, 54:1884, 54:1891, 54:1905, 54:1906, 54:1910, 54:1921, 54:1933, 54:2031, 54:2044, 54:2045, 54:2049, 54:2052, 54:2054, 54:2057, 54:2060, 54:2062, 54:2103, 54:2110, 54:2115, 54:2121, 54:2126, 54:2130, 54:2136, 54:2144, 54:2146, 54:2153, 54:2158, 54:2163, 54:2165, 54:2171, 54:2173, 54:2180, 54:2209, 54:2211, 54:2226, 54:2228, 54:2242, 54:2267, 54:2271, 54:2276, 54:2288, 54:2305, 54:2310, 54:2318

`\fl@tracemessage` 54:1905, 54:1922, 54:1931, 54:1933

`\fl@traceval` 54:1915, 54:1916, 54:1917, 54:1918, 54:1921, 54:1930, 54:1933

`\float@count` 02:51, 02:52, 02:53, 02:62, 02:188, 02:205, 02:211, 02:213, 02:214, 02:222, 02:230

`\fmtversion@topatch` 57:655, 57:657, 57:669, 57:670, 57:682, 57:690

`\font@info` 26:99, 26:365, 26:434, 26:439

`\font@name` . . 21:179, 21:182, 24:86, 24:194, 24:196, 24:379, 24:394, 24:570, 24:604, 26:84, 26:88, 26:90, 26:105, 26:141, 26:144, 26:154, 26:168, 26:171, 26:330, 26:331, 26:332, 26:333, 26:334, 26:339, *1193*

`\font@submax` 26:441, 26:470, 26:471, 37:53, 37:55, 37:118, 37:120, 37:157, 37:159, 57:299, 57:301, 57:310, *1208*

`\fps@dbl` 45:34

`\freeze@math@version` . 28:154, 28:273

`\frozen@everydisplay` 24:349, 24:353, 24:368, 24:370, *611*

`\frozen@everymath` 24:349, 24:361, 24:372, *611*

`\g@addto@macro` 09:61, 25:731, 28:403, 50:133, 50:460, 50:1120, 50:1136, 50:1137, 53:352, 53:358, 53:408, 06:918, *229*

`\G@refundefined` *1230*

`\G@refundefinedfalse` 35:5, *739*

`\G@refundefinedtrue` . . 35:3, 35:16, 35:33, 35:50, 36:203, 36:210, 36:220, 36:226, 47:41, 47:68, 47:85, *1230*

`\gen@sfcnt` 26:502, 26:503, 26:504

`\genb@sfcnt` 26:507, 26:508, 26:509, *1229*

`\genb@x` 26:510, 26:512

`\genb@y` 26:512

`\get@cdp` 28:571, 28:579, 28:612

`\get@external@font` 26:83, 26:96, 26:536

`\getanddefine@fonts` 24:617, 24:635, 26:320, 28:59, 28:87, 28:132, 28:159, 28:190, 28:220, 28:251, 28:294, 28:333, 28:438, 28:522, 28:576, 28:578, 28:595, 28:718, 28:719, 28:751, 28:752, 28:1194, 28:1195, *610*

`\glb@currsize` 20:41, 20:111, 20:169, 24:346, 26:217, 26:252, 26:256, 26:262, 26:285

`\glb@settings` 24:347, 26:217, 26:264, 26:295

`\gobble@finish@module@release` 03:109, 03:111, 03:170

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

\gobble@font@spec *1227*
\group@elt 28:35, 28:436, 28:483, 28:484, 28:515, 28:519, 28:1226
\group@list 28:440, 28:490, 28:513, 28:518, 28:519, 28:568, 28:794, 28:842, 28:903, 28:988, 28:991, 28:1042, 28:1045, 28:1111, 28:1114, 28:1181, 28:1232, *1236*
\h@false 38:81
\h@true 38:82, 38:83
\hb@xt@ 21:425, 38:210, 38:376, 38:455, 38:470, 38:482, 38:509, 38:540, 06:29, 40:48, 40:63, 40:86, 40:104, 40:248, 40:267, 40:614, 40:618, 40:619, 40:620, 41:37, 42:31, 42:43, 42:62, 42:74, 42:105, 42:117, 42:265, 42:299, 42:302, 42:305, 42:307, 42:314, 42:373, 42:452, 42:587, 42:597, 42:740, 42:796, 44:218, 44:241, 44:248, 54:634, 54:644, 54:693, 54:703, 54:1847, 54:2261, 54:2262, 54:2266, 54:2293, 54:2294, 54:2300, 02:549, *1224*
\hexnumber@ . 28:815, 28:823, 28:858, 28:866, 28:887, 28:897, 28:923, 28:931, 28:939, 28:948, 28:951, 28:960, 28:961, 28:1000, 28:1008, 28:1054, 28:1062, 28:1081, 28:1082, 28:1092, 28:1093, 28:1098, 28:1124, 28:1132, 28:1137, 28:1139, 29:645
\hgl@ 02:504, 02:505
\hmode@bgroup 21:67, 21:75, 21:327, 21:356, 21:390, 21:396, 21:427, 21:438, 21:445, 21:476, 21:483, 21:486, 21:488, 21:496, 21:512, 21:730, 21:760, 21:766, 21:798, 21:805, 21:833, 21:836, 21:893, 21:1253, 32:7, 33:625, 33:632, *1238*
\hmode@start@before@group 21:68, 21:151, 21:153, 21:159, 21:184
\hyper@nopatch@longtable 55:241
\if@afterindent 44:124, 44:131
\if@compatibility 50:2, 50:676
\if@endpe 37:338, 37:353, 37:370, 37:380, 39:140, *1269*
\if@eqnsw 38:357, 38:423
\if@fcolmade 54:97, 54:266, 54:396, 54:405, 54:443, 54:453, 54:783, 54:803, 54:821, 54:850, 54:930, 54:2208, 54:2225, 54:2275, 54:2315
\if@filesw 20:5, 20:36, 20:106, 20:164, 20:298, 20:310, 20:331, 20:360, 20:372, 20:387, 20:407, 20:419, 20:426, 20:438, 37:21, 37:59, 37:87, 37:124, 37:149, 37:163, 44:153, 47:4, 47:8, 47:39, 47:48, 47:56, 47:67, 47:84, 50:1262, 50:1279, 50:1393, 50:1412, 53:361, *1219*
\if@firstamp 41:251
\if@firstcolumn 54:97, 54:248, 54:281, 54:398, 54:446, 54:1819, 54:2239, 54:2284, *949*
\if@font@series@context 29:509, 29:528, 29:536, *651*
\if@forced@series 25:401, 29:143, *1250*
\if@ignore 28:301, 28:340, 37:4, 37:339, 37:356, 37:372, 37:381
\if@in@minipage@env . . 40:423, 40:445
\if@includeinrelease 03:73, 03:76, 03:132, 06:872
\if@inlabel 39:28, 39:65, 39:102, 39:184, 39:207, 54:163, 54:190
\if@inmath *1232*
\if@insert 54:97, 54:1061, 54:1173, 54:1207, 54:1341, 54:1376, 54:1450, 54:1539, 54:1666, 54:1794
\if@listfiles@hashes 20:743, 20:778, 20:785, 20:789
\if@listfiles@sizes 20:743, 20:779, 20:783
\if@minipage 16:26, 18:240, 18:255, 18:272, 18:291, 18:324, 18:358, 37:528, 37:550, 39:173, 40:420, 41:79, 45:20
\if@mparswitch 54:97, 54:1821
\if@multiplelabels 35:93
\if@mypkg@draft *1027*
\if@negarg 42:157, 42:198, 42:212, 42:273
\if@newlist 37:570, 39:29, 39:33, 39:69, 39:78, 39:106, 39:190, 54:603, 54:648, 54:661, 54:707
\if@nmbrlist 39:33, 39:225
\if@no@font@opt . 27:16, 27:110, 27:129
\if@nobreak 18:121, 18:148, 18:293, 18:326, 18:360, 20:203, 20:215, 39:191, 39:216, 40:383, 40:404, 44:47, 44:128, 45:180, 45:373, 48:241, 49:33, 49:44, 49:58, 49:66, 54:167, 54:194, 54:337, 54:1152, 54:1318, *1234*
\if@noitemarg 39:32, 39:223
\if@noparitem 39:30, 39:181
\if@noparlist 39:31, 39:114
\if@noskipsec 18:148, 39:58, 40:384, 40:405, 44:38, 44:40, 44:97, 45:374, 54:157, 54:184
\if@ovb 42:427, 42:494, 42:525, 42:550, 42:561, 42:574

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\if@ovhline 42:459, 42:589
\if@ovl 42:427, 42:492, 42:521, 42:546, 42:591, 42:600
\if@ovr 42:427, 42:491, 42:520, 42:545, 42:588, 42:598
\if@ovt 42:427, 42:493, 42:524, 42:549, 42:566, 42:578
\if@ovvline 42:459, 42:564
\if@partsw . 20:5, 20:302, 20:364, 20:411
\if@pboxsw 40:351, 40:373, 40:529
\if@reversemargin 54:97, 54:1824
\if@rjfield 41:19, 41:33
\if@skipping@module 03:137, 03:149, 03:152
\if@specialpage . 54:97, 54:610, 54:668
\if@tempswa . . 12:9, 20:308, 20:370, 20:417, 21:1542, 24:99, 24:713, 28:471, 28:546, 28:620, 28:701, 28:1225, 37:61, 37:126, 37:165, 37:535, 37:556, 47:95, 50:1291, 50:1424, 50:1513, 52:292, 52:293, 52:315, 52:316, 52:336, 52:337, 54:994, 54:1030, 54:1630, 54:1755, 02:270, 01:62, 01:63, 01:64, *1241*
\if@test 54:12, 54:13, 54:891, 54:910, 54:950, 54:972, 54:1036, 54:1120, 54:1129, 54:1278, 54:1289, 54:1431, 54:1518, 54:1636, 54:1761
\if@twocolumn 20:26, 20:96, 20:153, 45:32, 45:210, 45:235, 54:97, 54:141, 54:269, 54:280, 54:397, 54:445, 54:469, 54:785, 54:841, 54:1818, 54:2210, 54:2227
\if@twoside 54:97, 54:140, 54:613, 54:671
\ifdt@p 38:203, 38:205
\IfFileExists@ 20:471, 20:498
\IfFileExists@@ 20:498
\ifG@refundefined 35:3, 35:4, 35:5
\ifh@ 38:76, 38:114, 38:123
\ifin@ 21:1562, 21:1565, 27:50, 27:52, 28:1, 28:22, 28:425, 28:567, 28:569, 28:630, 28:643, 28:713, 28:715, 28:743, 28:795, 28:809, 28:843, 28:855, 28:904, 28:920, 28:989, 28:992, 28:1013, 28:1043, 28:1046, 28:1109, 28:1112, 28:1115, 28:1182, 28:1184, 28:1213, 29:210, 29:214, 29:218, 50:236, 50:254, 50:548, 50:582
\ifmath@fonts 24:204, 26:222
\ifmaybe@ic 32:82, 32:91, *1241*
\ifnot@nil 26:343, 26:360, 26:381
\ifrestore@version *1232*
\iftc@forced . . 33:868, 33:879, 33:1147
\ifv@ 38:75, 38:113, 38:122
\ifx *323*
\in@ 21:1560, 21:1563, 27:49, 27:51, 28:1, 28:21, 28:424, 28:566, 28:568, 28:626, 28:639, 28:712, 28:714, 28:741, 28:793, 28:804, 28:841, 28:852, 28:902, 28:916, 28:987, 28:990, 28:1010, 28:1041, 28:1044, 28:1106, 28:1110, 28:1113, 28:1180, 28:1183, 28:1211, 29:208, 29:212, 29:216, 50:235, 50:252, 50:545, 50:581, *564*
\in@@ 25:489, 25:490, 25:492, 25:493, 28:5, 28:6, 28:7, 28:9
\in@false 28:10
\in@true 28:12
\init@restore@glb@settings 26:265, 26:268, 26:270
\init@restore@version 28:62, 28:91, 28:108, 28:123, 28:124, *1232*
\init@series@setup 25:732, 25:737, 29:120, 29:206, 29:246
\input@path 01:93, 01:115, 01:117, 01:123, 01:125, 01:131, 01:133, 20:484, 20:538, 20:560, 20:587, 20:604, 01:138, 01:140, 01:150, 01:217, 52:275, *1052*
\insc@unt 54:61, 02:37, 02:51, 02:52, 02:53, 02:62, 02:90, 02:91, 02:92, 02:94, 02:247, 02:248, 02:249, 02:250, 02:251, 02:252, 02:263, 02:264, 02:265, 02:266, 02:267, 02:271, 02:273, 02:292, 02:293, 02:294, 02:295, 02:296, 02:297
\install@mathalphabet 24:612, 24:629, 24:636, 28:444, 28:447, 28:573, 28:574, 28:671, 28:723, 28:726, 28:733, 28:748, 28:749, 28:756, 28:1196, 28:1198
\is@range 26:376, 26:377
\kernel@ifnextchar . . 03:90, 06:81, 06:100, 06:150, 50:440, 06:783, 06:798
\kernel@make@fragile 18:25, 18:26, 18:27, 18:28, 18:29, 21:172, 21:173, 37:487, 37:488, 37:489, 38:90, 38:91, 38:92, 38:93, 38:179, 38:180, 38:181, 41:160, 41:161, 41:162, 42:822, 42:823, 42:824, 42:825, 42:826, 42:827, 42:828, 42:829, 42:830, 42:831, 42:832, 42:833, 44:23, 44:24, 44:25, 44:26, 44:27, 49:72, 49:73, 06:387, 06:899, 06:900, 06:901, 06:902, 06:903, 06:904, 06:905, 06:906, 06:907, 06:908, 06:909, 06:910,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

06:911, 06:912, 06:913, 06:914, *29*
\l@ngrel@x 06:74, 06:75, 06:76, 06:120, 06:167
\l@nohyphenation 37:532, 37:724, 57:283, *1246*
\last@fontshape 24:545, 24:563, 24:580, 24:597
\latexrelease@postltcmd 07:3014
\latexrelease@postltexpl 05:132
\leavevmode@ifvmode 18:537, 18:538, 18:546, 30:631, 30:633, 30:635, 30:637, 38:115, 38:154, 38:219, 38:239, 38:240
\load@onefile@withoptions 50:945, 50:987, 50:1099, *1002*
\load@onefilewithoptions 50:892, 50:1049, 50:1600, *1001*
\lower@bound . . . 26:386, 26:387, 26:398
\lst@vskip *414*
\LT@cols *1168*
\ltx@sh@ft . . 21:390, 21:397, 21:476, 21:484, 21:760, 21:767, 02:541, *1241*
\m@ne 02:39, *382*
\m@th 19:13, 30:344, 30:466, 30:468, 30:469, 30:472, 30:513, 30:537, 30:540, 30:544, 30:547, 30:554, 30:557, 30:564, 30:567, 30:649, 38:68, 38:71, 38:106, 38:140, 38:147, 38:167, 38:169, 38:185, 38:204, 38:367, 38:470, 38:482, 38:509, 38:520, 40:351, 40:373, 40:555, 41:181, 44:214, 44:237, 45:400, 45:409, 45:416, 45:437, 45:444, 02:521, 02:533, *1230*
\makeph@nt 38:101, 38:103
\makesm@sh 38:131, 38:133
\mandatory@arg 26:414, 26:501, 26:505, 26:510, 26:517, 26:519, 26:524, 26:526, 26:531, 26:533, 26:546, 26:562, 26:569, 26:571
\math@bgroup 24:643, 26:306, 26:312, 28:53, 28:81, 28:146, 28:175, 28:184, 28:206, 28:214, 28:245, 32:130, 32:131, 32:138, *608*
\math@egroup 24:643, 26:310, 26:311, 32:131, 32:132, 32:139, *1194*
\math@famname *1193*
\math@fonts 24:613, 24:618, 26:232, 26:336, 28:60, 28:89, 28:160, 28:191, 28:221, 28:253
\math@fontsfalse 19:7, 21:302, 21:329, 21:360, 21:1255, 24:77, 24:206, 24:216, 24:239, 33:103, 33:673, 33:929
\math@fontstrue 24:204, 24:655
\math@group *1193*
\math@version 24:7, 24:336, 24:617, 24:621, 24:623, 24:624, 24:626, 26:230, 28:56, 28:59, 28:64, 28:65, 28:69, 28:84, 28:88, 28:93, 28:94, 28:98, 28:111, 28:112, 28:113, 28:126, 28:127, 28:128, 28:149, 28:151, 28:153, 28:154, 28:159, 28:163, 28:165, 28:167, 28:171, 28:187, 28:190, 28:194, 28:196, 28:198, 28:202, 28:217, 28:220, 28:224, 28:226, 28:228, 28:232, 28:248, 28:252, 28:256, 28:258, 28:260, 28:264, 29:617, *607*
\math@version. *1193*
\mathchar@type 28:887, 28:897, 28:948, 28:951, 28:960, 28:976, 28:1081, 28:1092, 28:1155
\mathph@nt 38:99, 38:105
\mathsm@sh 38:129, 38:138, 38:139, 38:145, 38:146
\maybe@ic 32:63, 32:64, 32:83
\maybe@ic@ 32:83
\maybe@icfalse 32:97
\maybe@ictrue 32:87
\maybe@load@fontshape 21:71, 25:476, 25:515, 26:127, 29:164, *473*
\maybe@update@bfseries@defaults 29:38, 29:257, 29:266, 29:304
\maybe@update@mdseries@defaults 29:39, 29:280, 29:289, 29:305
\mb@b 40:76, 40:87, 40:95, 40:105
\mb@l 40:76, 40:80, 40:86, 40:95, 40:99, 40:104, 42:137, 42:141
\mb@r 40:76, 40:80, 40:86, 40:95, 40:99, 40:104, 42:137, 42:141
\mb@t 40:77, 40:84, 40:96, 40:103
\md@def@ult 29:397
\mddef@ult 29:198, 29:293, 29:294, 29:295, 29:334, 29:335, 29:336, 29:402, 29:443, *1251*
\mddefault@previous 29:290, 29:292, 29:331, 29:333, 30:114, 30:124, *1254*
\mdseries@.. . . . . *636*
\mdseries@previous *1254*
\mdseries@rm 29:111, 29:127, 29:128, 29:229, 29:237, 29:281, 29:293, 29:334, 29:339, 29:366
\mdseries@sf 29:112, 29:128, 29:238, 29:282, 29:294, 29:335, 29:340, 29:367
\mdseries@tt 29:113, 29:128, 29:239, 29:283, 29:295, 29:336, 29:341, 29:368
\meaning *324*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\merge@font@series 25:408, 25:425, 25:426, 25:507, 25:509, 26:133, *560*
\merge@font@series@ 25:428, 25:433, 25:510
\merge@font@series@without@substitution . . 25:460, 25:475, 25:512, 26:136, *562*
\merge@font@series@without@substitution@ 25:460, 25:513
\merge@font@shape 25:626, 25:639, 25:640, 25:711, 25:716, 26:132
\merge@font@shape@ 25:642, 25:647, 25:717
\merge@font@shape@without@substitution 25:666, 25:719, 26:135
\merge@font@shape@without@substitution@ 25:666, 25:720
\mv@⟨*version*⟩ *610*
\mv@⟨*version*⟩@frozen *607*
\mv@⟨*version*⟩@reset *610*
\n@space 30:632, 30:634, 30:636, 30:638, 30:643, 30:644, 30:645, 30:646, 30:649
\new@command 06:78, 06:77, 06:131, 06:165, 06:184, 06:251
\new@environment 06:147, 06:146, 06:159
\new@fontshape 27:2, 27:4, 27:22, 27:24
\new@label@record 36:57, 36:128, 36:278, 37:25, *749*
\new@mathalphabet 28:624, 28:645, 28:656
\new@mathgroup 04:27, 24:14, 28:474, 02:78, 02:80, 02:98, 02:100, *1226*
\new@mathversion 28:20, 28:407, 28:415, 28:420, 28:423
\new@module@skip 03:138, 03:150, 03:152
\new@moduledate 03:82, 03:101, 03:104, 03:152
\new@modulename 03:96, 03:152
\new@symbolfont 28:475, 28:517
\newcommand *305*
\NewCommandCopy *312*
\NewDocumentCommand *305*
\newlinechar *322*
\newmathalphabet@ 27:14
\newmathalphabet@@ 27:109
\newmathalphabet@@@ . . . 27:15, 27:109
\nfss@catcodes 24:19, 24:120, 24:412, 24:413, 24:420, 24:467, 30:40, 30:45, 30:53, 30:142, 54:3, *1229*
\nfss@text . . 21:315, 21:317, 29:648, 32:5, 32:122, 35:17, 35:34, 35:51, *1216*
\no@alphabet@error 24:4, 28:443, 28:445, 28:661, 28:662, 28:676, 28:685, 28:771, 28:772, *1208*
\no@alphabet@help *1194*
\no@version@warning *1194*
\noaccents@ 24:658, 30:136
\noexpand *324*
\non@alpherr 24:637, 24:639, 28:72, 28:101, 28:117, 28:174, 28:205, 28:235, 28:267, 28:1233
\not@base . . . 29:720, 29:724, 29:725, 29:726, 29:727, 29:728, 29:729, 29:730, 29:731, 29:732, 29:733, 29:734
\not@math@alphabet 25:527, 25:532, 25:537, 25:683, 25:687, 25:690, 25:693, 25:696, 25:699, 25:702, 25:705, 29:5, 29:8, 29:11, 29:14, 29:17, 29:20, 29:23, 29:26, 29:29, 29:255, 29:278, 29:308, 29:329, 29:353, 29:364, 29:381, 29:384, 29:406, 29:411, 29:416, 29:449, 29:454, 29:459, 29:475, 29:478, 29:481, 29:484, 29:487, 29:597
\now@and@everyjob 04:215, 04:221, 21:1011
\o@lign 21:390, 21:397, 21:476, 21:484, 21:760, 21:767, 02:535
\on@line 14:8, 14:15, 14:214, 29:144, 29:147, 37:252, 37:273, 37:291, 37:301, 40:189, 50:953, 50:1088, 08:534, 08:607, 08:1478, 08:1513, *407*
\operator@font . 30:650, 38:3, 38:4, 38:5, 38:6, 38:7, 38:8, 38:9, 38:10, 38:11, 38:12, 38:13, 38:14, 38:15, 38:16, 38:17, 38:18, 38:19, 38:20, 38:21, 38:22, 38:23, 38:24, 38:25, 38:26, 38:27, 38:28, 38:29, 38:30, 38:31, 38:32, 38:33, 38:34, 38:37, 38:40
\optional@arg 26:415, 26:494, 26:496, 26:568, 26:571
\outer@nobreak 45:181, 45:251, 45:255, 45:346, 45:364, *1221*
\p@ 02:311
\p@... . . . . *1249*
\p@enum *816*
\p@equation 38:364, 38:518
\p@renwd *1204*
\p@selectfont 26:119, *1194*
\par *169*
\par@deathcycles 39:56, 39:77, 39:79, 39:80
\patch@level 03:1, 03:39, 03:44, 03:46, 03:48, 03:52, 03:61, 57:658, 57:670, 57:672, *1244*
\patchcmd *305*
\ph@nt 38:81, 38:82, 38:83, 38:97
\pickup@font 21:181, 24:195, 24:378, 24:572, 24:606,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

26:143, 26:170, 26:331, 26:333, 26:335
\pictur@ 42:21
\pkgcls@arg 50:1611, 50:1742
\pkgcls@candidate 50:1598, 50:1620, 50:1696, 50:1700, 50:1704, 50:1773, 50:1776
\pkgcls@debug . . . 50:1588, 50:1604, 50:1605, 50:1606, 50:1607, 50:1608, 50:1672, 50:1673, 50:1674, 50:1675, 50:1684, 50:1689, 50:1707, 50:1716, 50:1732, 50:1766, 50:1767, 50:1768
\pkgcls@ext . . . 50:1612, 50:1660, *1268*
\pkgcls@innerdate 50:1593, 50:1645, 50:1648, 50:1654, 50:1794
\pkgcls@mindate 50:1625, 50:1634, 50:1650, 50:1655, *1020*
\pkgcls@name 50:1610, 50:1660
\pkgcls@parse@date@arg 50:1619, 50:1630
\pkgcls@parse@date@arg@ 50:1636, 50:1639
\pkgcls@parse@date@arg@version 50:1646, 50:1667
\pkgcls@releasedate 50:1598, 50:1701, 50:1705, 50:1777
\pkgcls@rollbackdate@error 50:1697, 50:1756, 50:1774
\pkgcls@show@selection 50:1724, 50:1730, 50:1780, 50:1785
\pkgcls@targetdate 50:1593, 50:1632, 50:1640, 50:1643, 50:1644, 50:1648, 50:1656, 50:1657, 50:1670, 50:1678, 50:1693, 50:1695, 50:1725, 50:1737, 50:1739, 50:1764, 50:1770, 50:1772, *1020*
\pkgcls@targetlabel 50:1593, 50:1633, 50:1653, 50:1668, 50:1680, 50:1712, 50:1746, 50:1784, 50:1787, *1020*
\pkgcls@use@this@release 50:1681, 50:1698, 50:1700, 50:1713, 50:1723, 50:1776
\pr@@@s 38:259, 38:267
\pr@@@t 38:262, 38:268
\pr@m@s 38:256, 38:257
\preload@sizes 27:11, 27:94, *1193*
\prepare@family@series@update 29:117, 29:141, 29:243, 29:407, 29:412, 29:417, 29:450, 29:455, 29:460, *648*
\pretocmd *304*
\prim@s 38:253, 38:255, 38:267
\prime@s 38:254
\process@table 20:40, 20:110, 20:168, 28:351
\propagate@doendpe 39:140, *811*
\protectd@edf *1262*
\protected *283*
\protected@cmd *1226*
\protected@edef . . 22:240, 29:546, 35:126, 35:136, 35:143, 35:151, 35:153, 35:174, 35:178, 35:185, 35:189, 35:196, 36:40, 36:45, 36:83, 36:84, 40:486, 40:503, 40:520, 44:60, 06:222, 45:472, 45:490, 45:508, 06:265, 50:539, 50:914, 50:1181, *1219*
\protected@file@percent 37:197, 37:204, 37:219, 37:227, 37:228, 44:165, 44:172, *766*
\protected@wlog 50:386, 50:388, 50:402, 50:416
\protected@write 20:202, 20:207, 35:105, 35:165, 36:55, 44:181, 46:14, 46:31, *1267*
\protected@xdef 44:11, 45:453, 45:521, 45:537, 06:265, 50:378, 50:397, 50:450, 50:811, *1222*
\provide@command 06:179, 06:178
\ps@empty 49:10, 57:158
\ps@plain 49:13
\q@curr@file 50:1209, 50:1253, 50:1255, 50:1260, 50:1286, 50:1384, 50:1386, 50:1391, 50:1419, *1253*
\quote@@name 20:462, 20:478
\quote@name 20:462, 20:475, 20:477, 20:588, 20:590, 20:599, 50:1384
\r@@t 38:66
\reenable@package@load 52:477, 53:464, *1055*
\reinstall@nfss@defs 25:685, 25:726, 25:730, 25:732, 25:736, 25:737, 25:740
\relax *324*
\rem@pt 24:329
\remove@angles 26:347, 26:370
\remove@nil 28:36
\remove@star 26:347, 26:353
\remove@tlig 21:1001, 21:1003, 21:1005, 21:1029, 21:1034, 21:1081, 21:1082, 21:1083, *1253*
\remove@to@nnil 24:328, 26:347, 26:373, 26:486
\renew@command 06:125, 06:124, 06:185, 06:193
\renew@environment . . . 06:153, 06:152
\requested@test@context 29:503, 29:520, 29:524, *651*
\reserved@@b *624*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

`\reserved@a` 01:105, 03:13, 03:19, 03:34, 01:109, 01:110, 03:181, 03:182, 13:33, 13:37, 14:234, 18:456, 18:459, 20:212, 20:213, 20:235, 20:244, 20:253, 20:306, 20:368, 20:415, 20:485, 20:487, 20:492, 20:494, 20:506, 20:509, 20:512, 20:515, 20:516, 20:517, 20:539, 20:541, 20:546, 20:548, 20:549, 20:550, 20:561, 20:563, 20:568, 20:570, 20:585, 20:591, 20:595, 20:602, 20:608, 20:612, 20:641, 20:644, 20:645, 20:647, 20:656, 20:659, 20:666, 20:669, 20:691, 20:692, 20:693, 20:697, 20:705, 20:722, 20:723, 20:727, 20:733, 20:763, 20:767, 20:770, 20:775, 21:81, 21:82, 21:86, 21:89, 21:97, 21:107, 21:110, 21:119, 21:138, 21:143, 21:995, 21:999, 21:1043, 21:1045, 21:1046, 21:1048, 21:1050, 21:1058, 21:1067, 24:30, 24:31, 24:32, 24:41, 24:44, 24:47, 24:63, 24:66, 24:69, 24:105, 24:108, 24:110, 24:147, 24:151, 24:414, 24:417, 24:544, 24:545, 24:560, 24:563, 24:568, 24:579, 24:580, 24:593, 24:597, 24:602, 24:629, 24:632, 24:633, 24:641, 25:435, 25:436, 25:439, 25:440, 25:455, 25:456, 25:468, 25:469, 25:648, 25:649, 25:652, 25:653, 25:674, 25:675, 26:196, 26:198, 26:200, 26:210, 26:212, 26:215, 26:344, 26:345, 26:358, 26:359, 01:179, 01:180, 27:53, 27:57, 01:183, 28:460, 28:463, 28:535, 28:538, 28:571, 28:580, 28:582, 28:626, 28:629, 28:639, 28:642, 28:740, 28:742, 28:804, 28:808, 28:852, 28:854, 28:915, 28:918, 28:1010, 28:1012, 28:1106, 28:1108, 28:1210, 28:1212, 28:1228, 28:1230, 28:1231, 28:1236, 29:40, 29:41, 29:73, 29:74, 29:174, 29:175, 01:201, 29:516, 29:517, 01:205, 32:47, 32:48, 32:53, 32:54, 32:65, 32:68, 32:88, 32:95, 01:227, 05:22, 05:23, 05:24, 05:25, 05:26, 05:47, 05:52, 36:40, 36:41, 36:45, 36:46, 36:83, 36:85, 01:234, 05:89, 05:95, 05:105, 37:195, 37:196, 01:237, 37:250, 37:251, 37:256, 37:271, 37:272, 37:276, 37:289, 37:290, 37:294, 37:299, 37:300, 37:304, 37:389, 37:390, 01:239, 01:240, 01:247, 38:418, 38:419, 38:420, 38:421, 38:423, 01:250, 01:252, 01:253, 40:78, 40:79, 40:82, 40:97, 40:98, 40:101, 40:184, 40:190, 01:260, 41:241, 01:263, 41:245, 41:250, 41:269, 41:360, 41:361, 01:265, 42:199, 42:201, 06:117, 42:205, 06:120, 06:133, 06:134, 06:135, 06:137, 42:477, 42:478, 42:505, 42:506, 42:534, 42:535, 06:184, 06:185, 06:186, 06:192, 06:193, 06:194, 06:195, 06:198, 45:29, 45:30, 45:32, 45:33, 45:63, 45:67, 45:72, 45:74, 45:76, 45:78, 45:83, 45:84, 45:132, 45:136, 45:142, 45:145, 45:148, 06:222, 45:151, 06:228, 06:238, 06:242, 06:300, 06:304, 06:338, 06:342, 50:131, 50:134, 50:136, 06:366, 06:370, 50:230, 50:238, 50:242, 50:248, 50:256, 50:260, 50:448, 50:450, 50:451, 50:452, 50:456, 50:471, 50:473, 50:474, 50:475, 50:479, 50:656, 50:660, 50:666, 50:670, 50:748, 50:749, 50:752, 50:813, 50:817, 50:829, 50:830, 50:832, 50:841, 50:845, 50:857, 50:858, 50:860, 50:868, 50:872, 50:884, 50:885, 50:886, 50:888, 50:900, 50:903, 50:904, 50:906, 50:990, 50:1039, 50:1056, 50:1097, 50:1220, 50:1221, 50:1223, 50:1355, 50:1356, 50:1358, 50:1400, 50:1401, 50:1403, 50:1407, 06:481, 06:489, 50:1622, 50:1627, 06:490, 50:1679, 50:1680, 50:1711, 50:1712, 50:1783, 50:1784, 50:1808, 50:1810, 06:522, 01:315, 06:530, 06:531, 06:532, 06:533, 01:316, 01:317, 52:161, 52:167, 52:168, 52:169, 06:573, 06:576, 06:591, 06:593, 06:596, 06:605, 06:627, 06:634, 06:659, 06:661, 54:37, 54:46, 54:48, 54:50, 54:881, 54:901, 06:780, 06:789, 54:2001, 54:2003, 54:2004, 54:2093, 54:2095, 54:2101, 54:2104, 57:226, 57:243, 57:244, 57:245, 57:252, 57:253, 57:254, 57:489, 57:520, 57:526, 57:527, 57:538, 57:539, 57:540, 57:547, 57:548, 57:549, 57:574, 57:575, 57:579, 57:582, 57:584, 57:586, 57:591, 57:594, 57:602, 57:603, 57:604, 57:671,

File Key: 01=`ltdirchk.dtx`, 02=`ltplain.dtx`, 03=`ltvers.dtx`, 04=`ltluatex.dtx`, 05=`ltexpl.dtx`, 06=`ltdefns.dtx`, 07=`ltcmd.dtx`, 08=`lthooks.dtx`, 09=`ltcmdhooks.dtx`, 10=`ltsockets.dtx`, 11=`lttemplates.dtx`, 12=`ltalloc.dtx`, 13=`ltcntrl.dtx`, 14=`lterror.dtx`, 15=`ltpar.dtx`, 16=`ltpara.dtx`, 17=`ltmeta.dtx`, 18=`ltspace.dtx`, 19=`ltlogos.dtx`, 20=`ltfiles.dtx`, 21=`ltoutenc.dtx`, 22=`ltcounts.dtx`, 23=`ltlength.dtx`, 24=`ltfssbas.dtx`, 25=`ltfssaxes.dtx`, 26=`ltfsstrc.dtx`, 27=`ltfsscmp.dtx`, 28=`ltfssdcl.dtx`, 29=`ltfssini.dtx`, 30=`fontdef.dtx`, 31=`preload.dtx`, 32=`ltfntcmd.dtx`, 33=`lttextcomp.dtx`, 34=`ltpageno.dtx`, 35=`ltxref.dtx`, 36=`ltproperties.dtx`, 37=`ltmiscen.dtx`, 38=`ltmath.dtx`, 39=`ltlists.dtx`, 40=`ltboxes.dtx`, 41=`lttab.dtx`, 42=`ltpictur.dtx`, 43=`ltthm.dtx`, 44=`ltsect.dtx`, 45=`ltfloat.dtx`, 46=`ltidxglo.dtx`, 47=`ltbibl.dtx`, 48=`ltmarks.dtx`, 49=`ltpage.dtx`, 50=`ltclass.dtx`, 51=`ltkeys.dtx`, 52=`ltfilehook.dtx`, 53=`ltshipout.dtx`, 54=`ltoutput.dtx`, 55=`lttagging.dtx`, 56=`lthyphen.dtx`, 57=`ltfinal.dtx`

57:674, 57:675, 57:692, 02:193, *1221*
\reserved@b 01:106, 01:107, 13:33, 13:34, 13:37, 18:457, 18:458, 18:465, 20:304, 20:306, 20:366, 20:368, 20:413, 20:415, 20:586, 20:588, 20:590, 20:603, 20:605, 20:607, 20:691, 20:692, 20:693, 20:766, 20:768, 20:769, 20:776, 21:90, 21:97, 21:112, 21:119, 21:998, 21:999, 21:1044, 21:1045, 21:1067, 21:1076, 21:1078, 24:31, 24:32, 24:38, 24:95, 24:97, 24:150, 24:151, 24:630, 24:641, 25:454, 25:455, 25:457, 27:47, 27:54, 27:71, 27:73, 28:461, 28:462, 28:463, 28:467, 28:469, 28:536, 28:537, 28:538, 28:542, 28:544, 28:579, 28:580, 28:581, 28:616, 28:618, 28:697, 28:699, 28:744, 28:745, 28:746, 28:753, 28:913, 28:917, 28:919, 29:181, 29:186, 29:190, 29:191, 29:198, 29:199, 32:52, 32:53, 32:66, 32:68, 32:95, 32:96, 05:48, 05:54, 36:84, 36:85, 41:246, 41:248, 41:250, 06:109, 06:111, 06:118, 06:135, 06:136, 45:43, 45:44, 45:112, 45:113, 06:239, 06:240, 06:242, 06:301, 06:302, 06:304, 06:339, 06:340, 06:342, 06:367, 06:368, 06:370, 50:231, 50:232, 50:233, 50:235, 50:250, 50:253, 50:448, 50:471, 50:821, 50:827, 50:830, 50:849, 50:855, 50:858, 50:876, 50:882, 50:886, 50:900, 50:907, 50:1269, 50:1270, 50:1273, 50:1274, 50:1309, 50:1310, 50:1312, 50:1339, 50:1402, 50:1403, 50:1406, 50:1407, 50:1442, 50:1443, 50:1445, 50:1471, 50:1531, 50:1532, 50:1534, 50:1561, 06:574, 06:576, 06:592, 06:596, 06:631, 06:634, 54:790, 54:793, 54:807, 54:810, 54:827, 54:830, 06:781, 06:791, 57:229, 57:231, 57:235, 57:492, 57:494, 57:498, 57:692, *524*
\reserved@c 01:107, 01:112, 20:756, 24:96, 24:97, 24:631, 24:634, 27:48, 27:55, 27:61, 27:68, 28:33, 28:37, 28:468, 28:469, 28:543, 28:544, 28:617, 28:618, 28:698, 28:699, 28:721, 28:730, 28:745, 28:759, 28:1000, 28:1017, 28:1026, 28:1054, 28:1065, 28:1123, 28:1136, 28:1138, 29:183, 29:186, 29:196, 29:197, 29:200, 29:201, 32:67, 32:69, 32:76, 50:901, 50:903, 50:904, 50:1254, 50:1259, 50:1260, 50:1278, 50:1286, 50:1292, 50:1314, 50:1322, 50:1385, 50:1390, 50:1391, 50:1411, 50:1419, 50:1425, 50:1447, 50:1454, 50:1502, 50:1503, 50:1504, 50:1514, 50:1536, 50:1543, 50:1571, 06:786, 06:789, 06:791, 06:794, 57:233, 57:238, 57:246, 57:496, 57:517, 57:518, 57:519, 57:521, 57:522, 57:523, 57:524, 57:525, 57:533, 57:541, 57:694, *1235*
\reserved@d 01:110, 01:113, 20:754, 20:756, 27:61, 27:68, 27:70, 27:74, 28:1008, 28:1017, 28:1026, 28:1062, 28:1065, 28:1131, 28:1136, 28:1140, 29:188, 29:189, 29:194, 29:195, 05:46, 05:51, 06:779, 06:788, 57:695, *1251*
\reserved@e 18:58, 18:60, 18:70, 18:72, 18:101, 18:108, 18:116, 27:39, 27:45, 27:70, 27:73, 27:74, 28:34, 28:39, 57:696, *1228*
\reserved@f 18:59, 18:60, 18:71, 18:72, 18:116, 21:1543, 21:1544, 21:1545, 21:1546, 21:1548, 21:1559, 24:190, 24:192, 24:198, 24:199, 26:382, 26:393, 26:397, 26:401, 26:407, 26:410, 26:449, 26:486, 26:489, 27:27, 27:38, 27:45, 27:71, 27:73, 57:697, *1221*
\reset@font . 29:219, 29:653, 29:684, 29:699, 29:714, 35:17, 35:34, 35:51, 40:482, 40:500, 40:517, 45:175, 45:371, 45:466, 45:485, 45:503, 47:40, 49:14, 54:620, 54:679, *1226*
\restglb@settings 26:268, 26:278
\restore@mathversion 28:107, 28:110, 28:125, 28:133
\restore@protect 06:265
\rlh@ 30:471, 30:472
\rm@def@ult 29:397
\rmdef@ult 29:258, 29:281, 29:319, 29:339, 29:355, 29:366, 29:398, 29:439, 33:27, 33:84, *643*
\robust@command@act 09:90, 06:422, 06:423, 06:425, 06:493, 06:550, 06:687, *96*
\robust@command@act@chk@args 06:447, 06:468, *96*
\robust@command@act@do 06:433, 06:465, *97*
\robust@command@act@end . . 06:430, 06:431, 06:443, 06:446, 06:466, *97*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\robust@command@act@loop 06:427, 06:433, 06:463, *97*
\robust@command@act@loop@aux ... 06:433, 06:464
\robust@command@chk@safe 09:137, 06:315, 06:426, 06:447, 06:467, 06:607, 06:619, *312*
\s@fct@ 26:426, 26:490
\s@fct@alias 26:552
\s@fct@fixed 26:565
\s@fct@gen 26:502
\s@fct@genb 26:507
\s@fct@sgen 26:502
\s@fct@sgenb 26:507
\s@fct@sub 26:514
\s@fct@subf 26:557
\saved@space@catcode . 57:355, 57:424
\scan@@fontshape ... 27:7, 27:40, 27:43
\scan@fontshape ... 27:6, 27:26, 27:37
\scantokens *320*
\scriptfont@name 26:333, 26:338
\section *305*
\select@group 24:614, 24:633, 28:48, 28:387, 28:448, 28:626, 28:679, 28:688, 28:726, 28:758, *607*
\series@change@debug 29:137, 29:144, 29:147, 29:158, 29:161, 29:165, 29:177, 29:185, 29:190, 29:196, 29:199, 29:201
\series@check@toks 25:490, 25:492, 25:499
\series@drop@one@m 25:496, 25:500, 25:518
\series@maybe@drop@one@m 24:31, 25:483, 25:485, 25:517, 28:462, 28:537, 29:168, 29:187, 29:193, 29:401, 29:402
\series@maybe@drop@one@m@x 25:486, 25:488
\seriesdefault@kernel 29:220, 29:773, *660*
\set@@mathdelimiter . 28:1063, 28:1097
\set@color 40:109
\set@curr@file ... 20:226, 20:235, 20:262, 20:270, 20:452, 20:470, 50:932, 50:1252, 50:1383, 52:266, *1002*
\set@curr@file@aux 52:272, 52:278, 52:279
\set@curr@file@nosearch . 52:266, *1261*
\set@display@protect 14:7, 14:14, 14:34, 14:61, 06:8, 06:16, 06:263, 50:389, *1219*
\set@fontsize 24:317, 24:319, 26:121, 26:167, 26:177, *1211*
\set@mathaccent 28:813, 28:821, 28:856, 28:864, 28:882
\set@mathchar 28:937, 28:947
\set@mathdelimiter 28:1014, 28:1023, 28:1075
\set@mathradical 28:395, 28:1133
\set@mathsymbol 28:921, 28:929, 28:950
\set@simple@size@args 26:348, 26:361, 26:368, 26:389, 26:403
\set@size@funct@args 26:351, 26:353, 26:411
\set@size@funct@args@ 26:411
\set@target@series 24:288, 24:300, 25:437, 25:441, 25:444, 25:447, 25:470, 25:472, 25:481, 25:516
\set@typeset@protect 41:197, 41:235, 06:263, 06:282, 53:84, 53:128, 54:607, 54:609, 54:665, 54:667, *1219*
\SetMathAlphabet@ 28:633, 28:702, 28:711
\SetSymbolFont@ 28:493, 28:547, 28:565
\SetSymbolFont@m@dropped 28:535, 28:540, 28:560, 28:561
\sf@def@ult 29:397
\sf@size 19:6, 21:302, 24:224, 24:243, 24:653, 26:328, 26:332, 33:672, 45:409, 45:416, 45:437, 45:444
\sfdef@ult 29:259, 29:282, 29:320, 29:340, 29:356, 29:367, 29:399, 29:440, 33:29, 33:86, *643*
\sh@ft 02:539, *1222*
\show@kernel@robust@command 06:613, 06:676, 06:704
\show@release@info 03:38, 03:41, 03:46, 03:51, 37:45, 37:46, 37:48, 37:110, 37:111, 37:113, *38*
\ShowCommand *312*
\sixt@@n 04:30, 24:14, 28:84, 28:248, 28:799, 28:801, 28:847, 28:849, 28:908, 28:910, 28:956, 28:958, 28:996, 28:998, 28:1004, 28:1006, 28:1050, 28:1052, 28:1058, 28:1060, 28:1119, 28:1121, 28:1127, 28:1129, 42:278, 42:293, 42:295, 45:62, 45:80, 45:131, 45:153, 02:16, 54:1009, 54:1055, 54:1194, 54:1362, 54:1596, 54:1660, 54:1717, 54:1787, 54:2047, 54:2056, 54:2112, 54:2128, 54:2161, 02:64, 02:66, 02:96, 02:97, 02:98, 01:55
\sixt@@n␣ *382*
\size@update 26:146, 26:172, 26:185, 26:204, 26:206
\sizefn@info 26:352, 26:354, 26:362, 26:390, 26:404

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\skip@ 32:105, 32:108, 02:41, 02:501, 02:503, 02:504, 02:506
\sp@ce@skip 18:84, 18:517, 18:518
\sp@n 41:379
\split@name 24:382, 24:394, 24:508, 24:523, 26:519, 26:533
\ssf@size . . 21:328, 21:360, 21:1254, 24:225, 24:244, 24:654, 26:328, 26:334
\string@makeletter 50:906, 50:941, 50:942, 50:943, 52:169, 06:806, *1254*
\strip@meaning *1204*
\strip@prefix 01:95, 24:611, 01:212, 06:240, 06:302, 06:340, 06:368, 01:307, 06:803
\strip@pt . . . 24:216, 24:222, 24:223, 24:224, 24:225, 24:238, 24:242, 24:329, 24:653, 24:654, 26:180, 02:543
\sub@sfcnt 26:514, 26:515, 26:516, 26:543
\subf@sfcnt 26:557, 26:558, 26:559
\subst@correction 24:85, 24:91
\subst@fontshape 27:8, 27:80
\subst@size 26:465
\sw@slant 32:91, 32:101
\t@st@ic 32:90, 32:94, *1199*
\target@meta@family@value 29:150, 29:175, 29:182, 29:184
\target@series@value 29:149, 29:157, 29:160, 29:162, 29:166, 29:167, 29:168, 29:191, 29:197, 29:198, 29:200, *647*
\tc@check@accent 33:109, 33:161, 33:163, 33:165, 33:167, 33:169, 33:171, 33:173, 33:175, 33:177, 33:179, 33:181, 33:183, 33:185, 33:187, 33:189, 33:191, 33:935, 33:1011, 33:1013, 33:1015
\tc@check@symbol . . 33:109, 33:211, 33:213, 33:215, 33:217, 33:219, 33:221, 33:223, 33:225, 33:227, 33:229, 33:231, 33:233, 33:235, 33:237, 33:239, 33:241, 33:243, 33:245, 33:247, 33:249, 33:251, 33:253, 33:255, 33:257, 33:259, 33:261, 33:263, 33:265, 33:267, 33:269, 33:271, 33:273, 33:275, 33:277, 33:279, 33:281, 33:283, 33:285, 33:287, 33:289, 33:291, 33:293, 33:295, 33:297, 33:299, 33:301, 33:303, 33:305, 33:307, 33:310, 33:312, 33:314, 33:316, 33:318, 33:320, 33:322, 33:324, 33:326, 33:328, 33:330, 33:332, 33:334, 33:336, 33:338, 33:340, 33:342, 33:344, 33:346, 33:348, 33:350, 33:354, 33:356, 33:358, 33:360, 33:362, 33:364, 33:366, 33:935, 33:1005, 33:1007, 33:1009, 33:1017, 33:1019, 33:1021, 33:1023, 33:1025, 33:1027, 33:1029, 33:1031, 33:1033, 33:1035, 33:1037, 33:1039, 33:1041, 33:1043, 33:1045, 33:1047, 33:1049, 33:1051, 33:1053, 33:1055, 33:1057, 33:1059, 33:1061, 33:1063, 33:1065, 33:1067, 33:1069, 33:1071, 33:1073, 33:1075, 33:1077, 33:1079, 33:1081, 33:1083, 33:1085, 33:1087, 33:1089, 33:1091, 33:1093, 33:1095, 33:1097, 33:1099, 33:1101, 33:1103, 33:1105, 33:1107, 33:1109, 33:1111, 33:1113, 33:1115, 33:1117, 33:1119
\tc@error 33:110, 33:915, 33:936
\tc@errorwarn 33:21, 33:76, 33:78, 33:823, 33:825, 33:827, 33:828, 33:874, 33:875, 33:876, 33:909
\tc@fake@euro 33:97, 33:352, 33:923, 33:1004
\tc@forcedfalse 33:868
\tc@forcedtrue 33:873
\tc@oldstylesubst 33:16, 33:20
\tc@subst . 33:77, 33:109, 33:908, 33:935
\tc@swap@accent 33:112, 33:113
\test@font@series@context 29:505, 29:515, 29:535, *651*
\test@next *1227*
\text@command 32:8, 32:46, *1216*
\textfont@name 26:331, 26:337
\tf@size 24:223, 24:243, 24:652, 26:328, 26:330, *1197*
\thr@@ 02:563, 02:573, 02:607, 26:58, 26:254, 26:260, 26:273, 26:280, 26:287, 26:292, 38:376, 38:539, 39:256, 39:267, 42:287, 42:288, 42:290, 42:291, 42:329, 42:355, 42:388, 42:411, 02:16, 57:84, 57:92, *1224*
\toks@ 03:94, 03:106, 03:108, 03:115, 03:119, 03:122, 03:127, 18:455, 18:456, 18:461, 24:148, 24:152, 24:154, 24:157, 24:221, 24:226, 28:6, 28:7, 28:434, 28:438, 28:444, 28:447, 28:452, 28:518, 28:519, 28:521, 28:522, 28:572, 28:574, 28:578, 28:595, 28:598, 28:657, 28:669, 28:670, 28:671, 28:717, 28:719, 28:725, 28:733, 28:737, 28:749, 28:752, 28:755, 28:763, 28:765, 28:1186, 28:1188, 28:1190, 28:1193, 28:1195, 28:1198, 28:1201, 28:1233,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

28:1234, 50:501, 50:502, 50:504, 50:505, 54:2247, 54:2248, 54:2249, 54:2250, 06:920, 06:921, 02:41, *1224*
\topmark *948*
\topmark(s) *948*
\try@load@font@shape *1236*
\try@load@fontshape 24:397, 24:405, 24:451, 24:556, 25:479, 26:520, 28:359, 28:376, *563*
\try@simple@size 26:356, 26:481
\try@simples . . . 26:439, 26:445, 26:449
\try@size@range 26:101, 26:356, 26:432
\try@size@substitution 26:103, 26:436
\tryif@simple 26:447, 26:448
\tryis@simple 26:448
\tt@def@ult 29:397
\ttdef@ult 29:260, 29:283, 29:321, 29:341, 29:357, 29:368, 29:400, 29:441, 33:31, 33:88, *643*
\tw@ 02:16, *1208*
\two@digits 01:169, 01:170, 26:512, 06:2, 50:1205, 50:1298, 50:1431, 50:1520, 01:70
\type@restoreinfo 26:202, 26:207
\undeclare@... . . . . *1084*
\undeclare@file@substitution 52:247, *1040*
\unexpandable@noexpand *1230*
\unexpanded *944*
\unqu@tefilef@und 52:143
\unquote@name 20:456, 20:462, 20:479, 52:349
\unrestored@protected@xdef 45:458, 45:526, 45:542, 45:553, 06:265, 48:223, 49:26, 49:54, 49:78
\unskip *953*
\update@series@target@value 29:118, 29:152, 29:173, 29:244
\update@uclc@with@cyrillic 21:1505, 21:1533, 21:1567, 21:1575
\upper@bound 26:383, 26:384, 26:385, 26:398
\use@mathgroup . . . 24:620, 24:638, 24:640, 26:299, 28:63, 28:92, 28:639, 28:741, 28:744, 28:1211, 28:1235, *1194*
\UTF@two@octets@noexpand *1261*
\UTFviii@four@octets 57:410, 57:415, 57:421
\UTFviii@four@octets@@ 57:410, 57:421
\UTFviii@four@octets@combine 57:445
\UTFviii@four@octets@noexpand 57:451
\UTFviii@four@octets@string . 57:448
\UTFviii@invalid 57:349, 57:442
\UTFviii@invalid@err 57:407, 57:412, 57:418
\UTFviii@invalid@err@@ 57:407, 57:418
\UTFviii@three@octets 57:409, 57:414, 57:420
\UTFviii@three@octets@@ 57:409, 57:420
\UTFviii@three@octets@combine 57:444
\UTFviii@three@octets@noexpand 57:450
\UTFviii@three@octets@string 57:447
\UTFviii@two@octets 57:408, 57:413, 57:419
\UTFviii@two@octets@@ 57:408, 57:419
\UTFviii@two@octets@combine . 57:443
\UTFviii@two@octets@noexpand 57:449
\UTFviii@two@octets@string . . 57:446
\UTFviii@undefined@err 57:406, 57:411, 57:417
\UTFviii@undefined@err@@ 57:406, 57:417
\v@false 38:82
\v@true 38:81, 38:83
\vbox *953*
\ver@... . . . . *464*
\ver@@... . . . . *464*
\ver@⟨*file*⟩.⟨*ext*⟩ *1001*
\verb@balance@group 37:654, 37:656, 37:671, 37:673, 37:685, 37:687, 37:700, 37:702, 37:708, 37:709
\verb@egroup 37:654, 37:657, 37:671, 37:674, 37:685, 37:688, 37:700, 37:703, 37:709, 37:713
\verb@eol@error 37:710, 37:722, 37:732
\verbatim@font 37:542, 37:563, 37:571, 37:723, 37:733, *1228*
\verbatim@nolig@list . 37:738, 37:744
\verbatim@out *1235*
\version@elt 28:18, 28:31, 28:32, 28:431, 28:432, 28:491, 28:521, 28:632, 28:670, 28:762, 28:1191
\version@list 28:16, 28:21, 28:32, 28:424, 28:432, 28:496, 28:527, 28:566, 28:637, 28:682, 28:712, 28:767, 28:1204
\vgl@ 02:501, 02:502
\voidb@x . . 23:23, 23:29, 02:311, 02:532
\vsplit *952*
\warn@rel@i 27:5, 27:25, 27:29, 27:81, 27:85, 27:90, 27:95, 27:119, 27:140
\wrong@fontshape . 24:401, 24:538, *1193*
\x@protect . . 06:243, 06:254, 06:305, 06:343, 06:371, 06:577, 06:597, *1219*
\xe@alloc@ 57:56, 57:66
\xe@alloc@intercharclass . 57:35, *1243*
\xe@ch@ck 57:57, 57:61

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\XXX@argdef *1206*
\z@ 02:311, *382*
\z@skip 02:311, *414*
\zap@space 20:262, 20:282, 29:546, 47:49, 50:231, 50:453, 50:476, 50:488, 50:655, 50:771, 50:811, 50:829, 50:840, 50:857, 50:867, 50:884, 50:916, 50:1218, 50:1353
\zref@labelbyprop *751*

tex commands:
\tex_afterassignment:D 53:49, 53:138, 07:2541
\tex_aftergroup:D . 53:57, 53:144, *1068*
\tex_currentgrouplevel:D 10:33, 53:48, 53:56, 53:137, 53:143
\tex_deadcycles:D 53:91
\tex_endlinechar:D 09:447, 07:1857, 07:1955, 07:1956, 07:1962, 07:1987, *1266*
\tex_escapechar:D 09:202, 09:277, 48:105, 48:106, 48:108, 52:212, 52:226, 07:1097, 07:1108, 07:1316, 07:1326, 07:1856, 07:2415, 07:2441, 08:1164, *256*
\tex_everypar:D 16:20, 16:24, 16:29, 16:53, 16:57, 16:61, 16:76, 16:76, 16:116, 16:118, 16:128, 16:129, 16:180, 16:181, *1260*
\tex_gdef:D 09:192, 09:267, *317*
\tex_hskip:D 16:102
\tex_indent:D 16:123
\tex_interactionmode:D 48:103, 48:104, 48:109
\tex_lastnodetype:D 16:101, 48:79, *415*
\tex_lastxpos:D 36:243
\tex_lastypos:D 36:244
\tex_lowercase:D 07:2238, 07:2421
\tex_marks:D 48:232
\tex_newlinechar:D 09:448
\tex_noindent:D 16:27, 16:59, 16:133, *410*
\tex_par:D 16:18, 16:51, 16:104, 16:111, 16:135, 16:176, 16:177, 16:178, *415*
\tex_parskip:D . . . 16:25, 16:26, 16:58
\tex_savepos:D *747*
\tex_setbox:D . . . 48:110, 53:50, 53:139
\tex_shipout:D . 53:126, 53:380, 53:510
\tex_splitbotmarks:D 48:129, 48:160, *956*
\tex_splitfirstmarks:D 48:141, 48:178, 48:185, *956*
\tex_splitmaxdepth:D 48:67
\tex_unskip:D 16:97, 48:77, *414*
\tex_vbadness:D 48:68
\tex_vfuzz:D 48:69
\tex_vsplit:D 48:110, *955*
\tex_vss:D 53:334
\tex_xdef:D 09:198, 09:273, *317*
\text *1196*

text commands:
\l_text_case_exclude_arg_tl . 57:621
\text_case_switch:nnnn 57:624
\text_declare_case_equivalent:Nn 57:626
\text_declare_lowercase_mapping:nn 57:630
\text_declare_lowercase_mapping:nnn 57:631
\text_declare_titlecase_mapping:nn 57:637
\text_declare_titlecase_mapping:nnn 57:638
\text_declare_uppercase_mapping:nn 57:644
\text_declare_uppercase_mapping:nnn 57:645
\text_lowercase:n *1262*
\textacutedbl 21:915, 21:1153, 33:234, 33:235, 33:721, 33:968, 33:1349
\textascendercompwordmark 21:854, 33:154, 33:685, 33:951, 33:1380
\textasciiacute 21:965, 21:1114, 33:236, 33:237, 33:722, 33:992, 33:1368
\textasciibreve 21:913, 21:1152, 33:238, 33:239, 33:723, 33:965, 33:1346
\textasciicaron 21:914, 21:1151, 33:240, 33:241, 33:724, 33:966, 33:1347
\textasciicircum 21:286, 21:558, 21:1088, *1246*
\textasciidieresis . . 21:953, 21:1101, 33:242, 33:243, 33:725, 33:982, 33:1365
\textasciigrave 21:904, 21:1082, 33:244, 33:245, 33:726, 33:963, 33:1337
\textasciimacron . . . 21:960, 21:1109, 33:246, 33:247, 33:727, 33:987, 33:1348
\textasciitilde 21:287, 21:559, 21:1093, *1230*
\textasteriskcentered 21:267, 21:717, 21:864, 21:865, 21:1215, 22:212, 22:218, 33:120, 33:577, 33:642, 33:958, 33:1383, *1246*
\textbackslash 21:268, 21:560, 21:718, 21:1087, *1223*
\textbaht 21:939, 21:1156, 33:252, 33:253, 33:731, 33:1098, 33:1099, 33:1359
\textbar 21:269, 21:561, 21:719, 21:1091, *1217*

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\textbardbl 21:270, 21:720, 21:919, 21:1170, 22:217, 33:127, 33:365, 33:366, 33:578, 33:650, 33:797, 33:971, 33:1264, *1239*
\textbf 32:19, 36:238, *638*
\textbigcircle 21:729, 21:892, 21:1232, 33:254, 33:255, 33:732, 33:1050, 33:1051, 33:1334, *1240*
\textblank 21:861, 21:1229, 33:345, 33:346, 33:785, 33:1020, 33:1021, 33:1271
\textborn 21:905, 33:256, 33:257, 33:387, 33:733, 33:1056, 33:1057, 33:1338
\textbraceleft 21:271, 21:308, 21:562, 21:721, 21:1090, *1218*
\textbraceright 21:272, 21:309, 21:563, 21:722, 21:1092, *1217*
\textbrokenbar 21:951, 21:1099, 33:128, 33:651, 33:980, 33:1392
\textbullet 21:273, 21:723, 21:921, 21:1179, 33:121, 33:579, 33:643, 33:973, 33:1387
\textcapitalcompwordmark 21:853, 33:153, 33:684, 33:950, 33:1379
\textcelsius 21:922, 21:1198, 33:129, 33:353, 33:354, 33:652, 33:791, 33:974, 33:1265
\textcent 21:947, 21:1095, 33:130, 33:653, 33:977, 33:1389
\textcentoldstyle 21:924, 33:258, 33:259, 33:392, 33:395, 33:734, 33:1072, 33:1073, 33:1352, *706*
\textcircled . . . 21:279, 21:283, 21:300, 21:301, 21:730, 21:893, 33:155, 33:156, 33:655, 33:671, 33:687, 33:855, 33:1120, 33:1122, 33:1373, *1218*
\textcircledP 21:958, 21:1200, 33:260, 33:261, 33:735, 33:1114, 33:1115, 33:1367
\textcolonmonetary 21:926, 21:1191, 33:313, 33:314, 33:765, 33:1074, 33:1075, 33:1295
\textcommaabove 21:353, 21:355, 21:369, 21:370, 21:458, 21:459, 21:700, 21:701, *1243*
\textcommabelow 21:324, 21:326, 21:332, 21:333, 21:703, 21:704, 21:705, 21:706, 21:707, 21:708, 21:709, 21:710, 21:711, 21:712, 21:1252, 21:1455, 21:1456, 21:1457, 21:1458, *1243*
\textcompsubstdefault 33:34, 33:39, 33:91, 33:618, 33:913, *1250*
\textcompwordmark 21:290, 21:291, 21:564, 21:1158, *1217*
\textcopyleft 21:956, 33:262, 33:263, 33:368, 33:736, 33:1112, 33:1113, 33:1366
\textcopyright 21:283, 21:317, 21:954, 21:1102, 33:131, 33:654, 33:983, 33:1394, *1235*
\textcurrency 21:949, 21:1097, 33:335, 33:336, 33:778, 33:850, 33:854, 33:1008, 33:1009, 33:1284
\textdagger 21:275, 21:312, 21:725, 21:917, 21:1177, 22:213, 22:219, 33:123, 33:581, 33:645, 33:969, 33:1384, *1218*
\textdaggerdbl 21:274, 21:313, 21:724, 21:918, 21:1178, 22:214, 22:220, 33:122, 33:580, 33:644, 33:970, 33:1385, *1218*
\textdblhyphen 21:876, 33:266, 33:267, 33:369, 33:738, 33:1022, 33:1023, 33:1322
\textdblhyphenchar 21:912, 33:264, 33:265, 33:370, 33:737, 33:1068, 33:1069, 33:1345
\textdegree 21:961, 21:1110, 33:132, 33:656, 33:988, 33:1398
\textdied 21:907, 33:268, 33:269, 33:388, 33:739, 33:1060, 33:1061, 33:1340
\textdiscount 21:941, 21:1190, 33:270, 33:271, 33:740, 33:1102, 33:1103, 33:1360
\textdiv 21:978, 21:1135, 33:133, 33:657, 33:1002, 33:1407
\textdivorced 21:906, 21:1235, 33:272, 33:273, 33:741, 33:1058, 33:1059, 33:1339
\textdollar 21:255, 21:307, 21:438, 21:565, 21:798, 21:862, 21:1084, 33:114, 33:115, 33:623, 33:625, 33:956, 33:1128, 33:1130, 33:1381, *1217*
\textdollaroldstyle 21:923, 33:274, 33:275, 33:393, 33:394, 33:742, 33:1070, 33:1071, 33:1351, *706*
\textdong 21:935, 21:1195, 33:315, 33:316, 33:766, 33:1092, 33:1093, 33:1298
\textdownarrow 21:903, 21:1210, 33:317, 33:318, 33:767, 33:1054, 33:1055, 33:1294
\texteightoldstyle 21:886, 33:214, 33:215, 33:384, 33:710, 33:1040, 33:1041, 33:1331
\textellipsis 21:296, 21:321, 21:1180

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\textemdash 21:256, 21:407, 21:411, 21:566, 21:570, 21:785, 21:1162, *1217*
\textendash 21:257, 21:408, 21:410, 21:567, 21:569, 21:786, 21:1161, *1217*
\textestimated 21:942, 21:1206, 33:329, 33:330, 33:774, 33:853, 33:1006, 33:1007, 33:1288
\texteuro 21:976, 21:1196, 33:351, 33:352, 33:789, 33:851, 33:1003, 33:1004, 33:1274
\textexclamdown 21:258, 21:412, 21:414, 21:571, 21:787, 21:1094, *1217*
\textfiguredash 21:410, 21:569, 21:1160, 21:1166, *1256*
\textfiveoldstyle 21:883, 33:216, 33:217, 33:381, 33:711, 33:1034, 33:1035, 33:1328
\textfloatsep 54:732, 54:745, 54:2142, 54:2192, 54:2341
\textflorin 21:925, 21:1149, 33:333, 33:334, 33:777, 33:975, 33:1283
\textfont 26:337, 38:251
\textfouroldstyle 21:882, 33:218, 33:219, 33:380, 33:712, 33:1032, 33:1033, 33:1327
\textfraction 54:1954, 54:1957, 54:1982, 54:1985, 54:2134, 54:2335, *1236*
\textfractionsolidus 21:877, 21:1187, 33:337, 33:338, 33:780, 33:959, 33:1277, 33:1509, *709*
\textgravedbl 21:916, 21:1154, 33:248, 33:249, 33:728, 33:967, 33:1350
\textgreater 21:281, 21:572, 21:740, 21:1086, *1230*
\textguarani 21:929, 33:276, 33:277, 33:391, 33:743, 33:1080, 33:1081, 33:1354
\textheight 20:22, 20:23, 20:92, 20:93, 20:149, 20:150, 45:257, 45:258, 45:261, 45:287, 45:301, 53:380, 54:78, 54:227, 54:228, 54:276, 54:401, 54:449, 54:478, 54:649, 54:708, 54:767, 54:819, 57:156, 57:157, *1208*
\texthorizontalbar 21:411, 21:570, 21:1163, 21:1168, *1256*
\texthyphen 21:260, 21:417, 21:574, 21:789, *1217*
\texthyphenchar 21:259, 21:416, 21:573, 21:788, *1223*
\textindent *1224*
\textinterrobang 21:933, 21:1186, 33:349, 33:350, 33:787, 33:1088, 33:1089, 33:1272
\textinterrobangdown 21:934, 21:1236, 33:347, 33:348, 33:786, 33:1090, 33:1091, 33:1273
\textit 32:21
\textlangle 21:888, 21:1227, 33:309, 33:310, 33:762, 33:1044, 33:1045, 33:1301
\textlbrace *1217*
\textlbrackdbl 21:900, 33:210, 33:211, 33:389, 33:707, 33:961, 33:1335
\textleaf 21:908, 33:278, 33:279, 33:373, 33:744, 33:1062, 33:1063, 33:1341
\textleftarrow 21:859, 21:1207, 33:319, 33:320, 33:768, 33:1016, 33:1017, 33:1291
\textlegacyasteriskcentered 33:589, 33:800
\textlegacybardbl 33:589, 33:801
\textlegacybullet 33:589, 33:802
\textlegacydagger 33:589, 33:804
\textlegacydaggerdbl 33:589, 33:803
\textlegacyparagraph 33:589, 33:805
\textlegacyperiodcentered 33:589, 33:806
\textlegacysection 33:589, 33:807
\textless 21:280, 21:575, 21:739, 21:1085, *1230*
\textlira 21:931, 21:1192, 33:321, 33:322, 33:769, 33:1084, 33:1085, 33:1297
\textlnot 21:957, 21:1107, 33:134, 33:658, 33:985, 33:1396
\textlquill 21:945, 21:1188, 33:280, 33:281, 33:745, 33:1108, 33:1109, 33:1363
\textmacron *1237*
\textmarried 21:909, 21:1234, 33:282, 33:283, 33:746, 33:1064, 33:1065, 33:1342
\textmd 32:19
\textmho 21:891, 21:1205, 33:284, 33:285, 33:747, 33:1048, 33:1049, 33:1333
\textminus ... 21:889, 21:1211, 33:343, 33:344, 33:783, 33:960, 33:1278, *711*
\textmu 21:966, 21:1115, 33:341, 33:342, 33:782, 33:993, 33:1280
\textmusicalnote 21:910, 21:1233, 33:286, 33:287, 33:748, 33:1066, 33:1067, 33:1343
\textnaira 21:928, 21:1193, 33:288, 33:289, 33:749, 33:1078, 33:1079, 33:1353
\textnineoldstyle 21:887, 33:220, 33:221, 33:385, 33:713, 33:1042, 33:1043, 33:1332

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\textnonbreakinghyphen 21:409, 21:568, 21:1159, 21:1164, *1256*
\textnormal 32:15
\textnumero 21:940, 21:1199, 33:331, 33:332, 33:775, 33:1100, 33:1101, 33:1287
\textogonekcentered 21:487, 21:698, 21:699, *1240*
\textohm 21:899, 21:1204, 33:339, 33:340, 33:781, 33:852, 33:1005, 33:1279
\textonehalf 21:974, 21:1124, 33:135, 33:659, 33:999, 33:1404
\textoneoldstyle 21:879, 33:222, 33:223, 33:377, 33:714, 33:1026, 33:1027, 33:1324
\textonequarter 21:973, 21:1123, 33:136, 33:660, 33:998, 33:1403
\textonesuperior 21:970, 21:1118, 33:137, 33:355, 33:356, 33:661, 33:792, 33:996, 33:1268
\textopenbullet 21:943, 21:1231, 33:290, 33:291, 33:750, 33:1104, 33:1105, 33:1361
\textordfeminine 21:305, 21:955, 21:1103, 33:138, 33:662, 33:984, 33:1395
\textordmasculine . . . 21:306, 21:971, 21:1119, 33:139, 33:664, 33:997, 33:1402
\TextOrMath 22:209, 22:212, 22:213, 22:214, 22:215, 22:216, 22:217, 22:218, 22:219, 22:220, 22:225, 22:232, *1242*
\textparagraph . 21:276, 21:310, 21:726, 21:967, 21:1116, 22:216, 33:124, 33:582, 33:646, 33:994, 33:1400, *1218*
\textperiodcentered 21:277, 21:727, 21:968, 21:1117, 33:125, 33:583, 33:647, 33:995, 33:1401
\textpermill *1236*
\textpertenmill *1236*
\textpertenthousand 21:492, 21:937, 21:1182, 33:306, 33:307, 33:308, 33:759, 33:1094, 33:1095, 33:1132, 33:1357, *704*
\textperthousand 21:490, 21:920, 21:1181, 33:118, 33:119, 33:639, 33:972, 33:1131, 33:1386, *1236*
\textpeso 21:930, 21:1197, 33:292, 33:293, 33:751, 33:1082, 33:1083, 33:1355
\textpilcrow 21:938, 33:294, 33:295, 33:386, 33:752, 33:1096, 33:1097, 33:1358
\textpm 21:962, 21:1111, 33:140, 33:666, 33:989, 33:1399
\textquestiondown . . . 21:261, 21:413, 21:415, 21:576, 21:790, 21:1126, *1240*
\textquotedbl 21:579, 21:1083, *1217*
\textquotedblleft 21:262, 21:418, 21:577, 21:791, 21:1174, *1217*
\textquotedblright 21:263, 21:419, 21:578, 21:792, 21:1175, *1217*
\textquoteleft 21:264, 21:420, 21:580, 21:793, 21:1171, *1217*
\textquoteright 21:265, 21:421, 21:581, 21:794, 21:1172, *1217*
\textquotesingle 21:863, 21:1081, 33:141, 33:667, 33:957, 33:1382
\textquotestraightbase 21:855, 33:142, 33:371, 33:668, 33:952, 33:1377
\textquotestraightdblbase . . . 21:856, 33:143, 33:372, 33:669, 33:953, 33:1378
\textrangle 21:890, 21:1228, 33:311, 33:312, 33:763, 33:1046, 33:1047, 33:1302
\textrbrace *1217*
\textrbrackdbl 21:901, 33:212, 33:213, 33:390, 33:708, 33:962, 33:1336
\textrecipe 21:932, 21:1201, 33:296, 33:297, 33:753, 33:1086, 33:1087, 33:1356
\textreferencemark 21:969, 21:1185, 33:298, 33:299, 33:754, 33:1116, 33:1117, 33:1369
\textregistered 21:300, 21:301, 21:959, 21:1108, 33:144, 33:670, 33:986, 33:1397, *1230*
\textrightarrow 21:860, 21:1209, 33:323, 33:324, 33:770, 33:1018, 33:1019, 33:1292
\textrm 32:15
\textrquill 21:946, 21:1189, 33:300, 33:301, 33:755, 33:1110, 33:1111, 33:1364
\textsc 32:21, *1235*
\textsection 21:278, 21:311, 21:582, 21:728, 21:952, 21:1100, 22:215, 33:126, 33:584, 33:585, 33:648, 33:981, 33:1393, *1218*
\textservicemark 21:944, 21:1202, 33:302, 33:303, 33:756, 33:1106, 33:1107, 33:1362
\textsevenoldstyle 21:885, 33:224, 33:225, 33:383, 33:715, 33:1038, 33:1039, 33:1330
\textsf 32:15
\textsixoldstyle 21:884, 33:226, 33:227, 33:382, 33:716, 33:1036, 33:1037, 33:1329

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\textsl 32:21
\textssc 25:536, 32:25
\textsterling . . 21:266, 21:319, 21:445, 21:583, 21:805, 21:948, 21:1096, 33:116, 33:117, 33:624, 33:632, 33:978, 33:1127, 33:1129, 33:1390, *1217*
\textstyle 19:15, 30:476, 38:63
\textsubscript 45:419
\textsuperscript 21:303, 21:305, 21:306, 33:663, 33:665, 33:679, 45:402, *1225*
\textsurd 21:972, 21:1226, 33:304, 33:305, 33:757, 33:1118, 33:1119, 33:1370
\textsw 25:531, 32:25
\TextSymbolUnavailable 21:3, 21:758, *1230*
\textthreeoldstyle 21:881, 33:228, 33:229, 33:379, 33:717, 33:1030, 33:1031, 33:1326
\textthreequarters . . 21:975, 21:1125, 33:146, 33:675, 33:1000, 33:1405
\textthreequartersemdash 21:858, 33:145, 33:357, 33:358, 33:375, 33:674, 33:793, 33:955, 33:1263
\textthreesuperior 21:964, 21:1113, 33:147, 33:359, 33:360, 33:676, 33:794, 33:991, 33:1267
\texttildelow 21:911, 21:1155, 33:250, 33:251, 33:729, 33:964, 33:1344
\texttimes 21:977, 21:1129, 33:148, 33:677, 33:1001, 33:1406
\texttrademark 21:303, 21:936, 21:1203, 33:149, 33:678, 33:976, 33:1388, *1230*
\texttt 32:15
\texttwelveudash 21:857, 33:150, 33:361, 33:362, 33:374, 33:680, 33:795, 33:954, 33:1262
\texttwooldstyle 21:880, 33:230, 33:231, 33:378, 33:718, 33:1028, 33:1029, 33:1325
\texttwosuperior 21:963, 21:1112, 33:151, 33:363, 33:364, 33:681, 33:796, 33:990, 33:1266
\textulc 25:526, 32:25
\textunderline *1217*
\textunderscore 21:288, 21:315, 21:584, 21:1089, *1217*
\textup 32:21, *566*
\textuparrow 21:902, 21:1208, 33:325, 33:326, 33:771, 33:1052, 33:1053, 33:1293
\textvisiblespace 21:292, 21:585, 21:1230, *1217*
\textwidth 20:24, 20:94, 20:151, 40:444, 45:266, 54:79, 54:146, 54:203, 54:220, 54:634, 54:644, 54:693, 54:703, 54:2261, 54:2293, 57:157, *1208*
\textwon 21:927, 21:1194, 33:327, 33:328, 33:772, 33:1076, 33:1077, 33:1296
\textyen 21:950, 21:1098, 33:152, 33:682, 33:979, 33:1391
\textzerooldstyle 21:878, 33:232, 33:233, 33:376, 33:719, 33:1024, 33:1025, 33:1323
\TH 21:535, 21:1131, 57:650, *1213*
\th 21:586, 21:1137, 57:650, *1213*
\thanks *898*
\thanks 44:10, 44:26, *1219*
\the *412*
\the... . . . . *1267*
\the#1 . . . . . *514*
thebibliography (env.) *937*
\theenum *816*
\theequation 38:352, 38:364, 38:457, 38:518
\thefootnote 45:396, 45:521, 45:526, 45:546
\thempfn 40:454, 45:453, 45:458, 45:537, 45:542, 45:545
\thempfootnote 40:454, 45:398
\thepage 20:209, 34:6, 35:18, 35:35, 35:52, 35:106, 35:166, 36:211, 36:222, 36:227, 36:234, 44:164, 44:171, 44:177, 46:15, 46:32, 47:43, 49:14, 54:246, 54:277, 54:1831, *750*
\Theta 30:306
\theta 30:282
\thetotalpages 53:350, 53:442, *1080*
\thicklines 42:124
\thickmuskip 30:653, 38:228, 38:230, 38:243
\thickspace 38:214
\thinlines 42:124, 42:815, 42:832
\thinmuskip 30:651, 38:220, 38:222, 38:238, 38:244
\thinspace 18:536, 18:542, 18:543, 38:189, 38:214, 38:251, *1251*
\thispagestyle 49:6
\tilde 18:472, 18:483, 30:526
\time 01:163, 01:167, 02:383
\times 30:403
\title *898*
\title 44:6, 44:7, 44:21, 44:23, 44:31
title 36:237, *751*
tl commands:
  \c_empty_tl 08:2194, 08:65
  \c_novalue_tl 07:525, 07:909, 07:965, 07:1026, 07:1502, 07:1602, 07:1718, 07:1823, 07:1968, 07:1975, 07:2034, 07:2086, 07:2167, 07:2194, 07:2276, *235*
  \c_space_tl . 09:365, 11:543, 11:547, 11:590, 11:788, 11:791, 16:42,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

16:73, 16:82, 52:553, 52:556, 07:24, 57:600, 57:605, 57:610, 57:615, 07:120, 07:136, 07:139, 07:153, 07:164, 07:165, 07:181, 07:189, 07:199, 07:202, 07:204, 07:206, 07:208, 07:216, 07:222, 07:290, 07:291, 07:1152, 07:1167, 07:1199, 07:1200, 07:1377, 07:1384, 07:1510
\tl_clear:N 09:196, 09:271, 11:284, 11:323, 11:706, 48:148, 48:149, 07:305, 07:350, 07:379, 07:446, 07:447, 07:663, 07:777, 07:788, 07:789, 07:790, 07:792, 07:950, 07:1018, 07:1212, 07:1445, 07:1446, 07:1518, 07:1539, 07:1629, 07:1630, 07:1646, 07:1858, 07:2226
\tl_const:Nn . . . 09:11, 09:12, 11:9, 11:10, 11:11, 11:12, 11:13, 11:14, 11:15, 53:311, 53:316, 07:1468, 07:2546, 08:35, 08:36, 08:110, 08:113, 08:919, 08:920, 08:922, 08:923, 08:924, 08:925, 08:928, 08:929, 08:931, 08:952, 08:953, 08:1297
\tl_count:N . . . 52:546, 07:397, 07:2250
\tl_count:n 07:602, 07:607, 07:864, 07:971, 07:1025, 07:1252, 07:1402, 07:1410
\tl_gclear:N *282*
\tl_gclear_new:N . 36:25, 36:145, 08:270
\tl_gput_left:Nn 51:235
\tl_gput_right:Nn 16:38, 16:69, 16:78, 28:320, 48:173, 48:182, 07:1127, 07:1438, 07:1440, 08:455, *265*
\tl_gremove_once:Nn 08:37, 08:37
\tl_gset:Nn 09:124, 16:15, 16:48, 17:34, 17:38, 28:284, 28:317, 36:26, 36:138, 36:146, 48:128, 48:139, 48:159, 48:208, 48:209, 48:210, 52:51, 55:105, 55:106, 55:126, 55:130, 55:176, 55:184, 55:230, 08:182, 08:194, 08:301, 08:319, 08:417, 08:429, 08:445, 08:484, 08:749, 08:803, 08:837, 08:841, 08:864, 08:868
\tl_gset_eq:NN 48:126, 48:127, 48:132, 48:134, 48:138, 48:197, 48:199, 48:201, 48:432, 48:434, 48:436, 48:438, 48:444, 48:448, 48:451, 55:207, 55:208, 08:63, 08:452, 08:463
\tl_head:N 07:2478
\tl_head:w 11:79, 11:92
\tl_if_blank:nTF 11:331, 11:335, 11:435, 11:438, 20:521, 55:53, 57:580, 07:598, 07:1612, 07:2048, 07:2089, 07:2300, 07:2989, 07:2990, 07:2991, 08:435
\tl_if_blank_p:n 07:116
\tl_if_empty:N *289*
\tl_if_empty:NTF . 08:2177, 08:2179, 11:245, 11:258, 11:311, 48:130, 48:161, 48:235, 07:316, 07:317, 07:465, 07:713, 07:1452, 08:235, 08:255, 08:366, 08:368, 08:637, 08:1038, 08:1428, 08:1988, 08:1996
\tl_if_empty:nTF . 08:2503, 09:120, 09:305, 09:324, 11:1148, 11:1153, 49:30, 49:40, 52:40, 52:46, 52:58, 52:107, 52:114, 52:116, 52:118, 52:416, 07:2045, 07:2416, 07:2442, 07:2477, 07:2669, 07:2759, 08:338, 08:353, 08:356, 08:824, 08:856, 08:910
\tl_if_empty_p:N 08:2453, 08:2454, 53:68, 53:114, 53:378
\tl_if_empty_p:n 08:887
\tl_if_eq:NNTF 48:267, 48:274, 48:441, 07:339
\tl_if_eq:NnTF 36:151
\tl_if_eq:nnTF 07:725, 07:984, 07:2421
\tl_if_exist:N *283*
\tl_if_exist:NTF . 08:2222, 08:2242, 08:2328, 08:2476, 08:2517, 36:106, 36:111, 36:130, 36:172, 36:187, 07:1453, 08:135, 08:233, 08:253, 08:1275
\tl_if_exist_p:N 08:276
\tl_if_head_eq_charcode:nNTF 20:523, 20:528
\tl_if_head_is_group:nTF 09:332, 07:2298, 07:2342, 07:2374
\tl_if_head_is_N_type:nTF 09:329, 07:2310, 07:2322, 07:2339, 07:2371
\tl_if_in:NnTF 07:1643
\tl_if_in:nnTF 11:280, 11:304, 11:326, 11:738, 07:614, 07:1609, 07:1635, *170*
\tl_if_novalue:nTF 07:370, 07:389, 07:436, 07:954, 07:2118, 07:2981, 07:2982, 07:2983, 07:2984, 07:2985, 07:2986, 08:332
\tl_if_single:nTF 11:486, 07:2215, 07:2964
\tl_if_single_token:nTF 09:315, 07:673, 07:2413, 07:2966, *191*
\tl_log:n 08:37, 08:39, 08:1821
\tl_map_function:nN 07:97, 07:390, 07:599, 07:600, 07:969, 07:1019, 07:3005
\tl_map_inline:Nn 07:723
\tl_map_inline:nn . . . 07:1481, 07:1715

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\tl_new:N 09:6, 09:8, 09:9, 09:10, 11:19, 11:20, 11:23, 11:24, 11:25, 11:26, 11:27, 11:37, 16:14, 36:137, 48:24, 48:25, 48:26, 48:27, 48:28, 48:29, 48:30, 48:31, 48:32, 48:33, 48:34, 48:35, 48:36, 48:37, 48:38, 48:39, 48:40, 48:41, 48:63, 48:64, 48:116, 48:192, 48:193, 52:8, 52:9, 52:10, 52:11, 52:59, 53:53, 53:205, 55:102, 55:103, 55:107, 55:108, 55:109, 55:110, 07:11, 07:12, 07:13, 07:14, 07:17, 07:21, 07:28, 07:29, 07:30, 07:33, 07:38, 07:39, 07:41, 07:42, 07:50, 07:51, 07:1624, 07:1625, 07:1841, 07:2212, 07:3000, 08:25, 08:26, 08:27, 08:29, 08:32, 08:78, 08:96, 08:124, 08:138, 08:141, 08:165, 08:166, 08:299, 08:317, 08:1294, 08:1550, 08:1551, 08:1552

\tl_put_left:Nn 07:830, 07:839, 07:1537

\tl_put_right:Nn 09:168, 09:182, 09:243, 09:257, 11:299, 11:306, 11:770, 11:778, 11:805, 11:816, 11:833, 11:858, 11:865, 57:621, 07:365, 07:391, 07:535, 07:568, 07:591, 07:605, 07:622, 07:627, 07:755, 07:855, 07:868, 07:895, 07:903, 07:917, 07:919, 07:926, 07:943, 07:959, 07:965, 07:1034, 07:1059, 07:1227, 07:1235, 07:1251, 07:1258, 07:1513, 07:1516, 07:1633, 07:1634, 07:1641, 07:1651, 07:1719, 07:1753, 07:1982, 07:1998, 07:2009, 07:2228, 07:2273

\tl_replace_all:Nnn . . 11:279, 07:2241

\tl_rescan:nn 09:20, 09:20, 09:462, 09:538, 09:566

\tl_set:Nn 09:166, 09:179, 09:180, 09:183, 09:185, 09:206, 09:241, 09:254, 09:255, 09:258, 09:260, 09:281, 09:373, 09:378, 09:459, 09:504, 09:535, 09:563, 11:255, 11:278, 11:322, 11:333, 11:337, 11:416, 11:542, 11:546, 11:589, 11:620, 11:727, 11:745, 11:876, 11:888, 11:910, 48:42, 48:43, 48:44, 48:45, 48:46, 48:47, 48:48, 48:49, 48:50, 48:51, 48:52, 48:53, 48:54, 48:55, 48:56, 48:57, 48:58, 48:59, 48:101, 52:81, 52:82, 52:83, 52:84, 53:47, 53:136, 53:210, 53:236, 53:261, 53:288, 55:42, 55:112, 55:113, 55:114, 55:115, 55:159, 55:160, 07:22, 07:88, 07:119, 07:306, 07:307, 07:308, 07:309, 07:337, 07:388, 07:874, 07:1071, 07:1089, 07:1291, 07:1509, 07:1544, 07:1606, 07:1712, 07:1739, 07:1746, 07:1759, 07:1765, 07:1771, 07:1777, 07:1783, 07:1789, 07:1795, 07:1801, 07:1832, 07:1854, 07:1877, 07:2208, 07:2209, 07:2235, 07:2259, 07:2290, 07:2302, 07:2329, 07:2395, 07:2397, 07:2516, 07:2517, 08:37, 08:40, 08:557, 08:1148, 08:1155, 08:1159, 08:1559, 08:1579, 08:1580, 08:1585, 08:1586, 08:1601, 08:1608, 08:1610, 08:1641, 08:1662, 08:1663, 08:1669, 08:1671, 08:1690, 08:1697, 08:1699, 08:1746, 08:1753

\tl_set_eq:NN 09:197, 09:272, 48:153, 48:154, 55:43, 55:156, 55:161, 07:327, 07:333, 07:341, 07:384, 07:1538, 07:1645, 07:1718, 08:1589, 08:1613, 08:1674, 08:1702

\tl_show:n 07:1333, 07:1346, 07:1355, 07:1391, 07:1414, 08:37, 08:38, 08:1826, 08:1929, 08:2030

\tl_tail:N 07:1911

\tl_to_str:N 07:1966, 07:1973, 07:2554

\tl_to_str:n 09:113, 09:118, 09:457, 09:476, 09:505, 09:516, 09:552, 11:302, 11:417, 11:621, 11:876, 11:943, 11:944, 11:945, 11:954, 11:955, 36:10, 36:19, 36:51, 36:64, 36:92, 36:93, 36:100, 36:101, 36:145, 36:146, 36:187, 48:230, 48:245, 51:121, 51:150, 51:152, 52:580, 07:70, 07:81, 07:248, 07:252, 07:460, 07:468, 07:481, 07:492, 07:496, 07:532, 07:563, 07:644, 07:668, 07:676, 07:691, 07:704, 07:716, 07:735, 07:772, 07:773, 07:1225, 07:1299, 07:1305, 07:1513, 07:1517, 07:1967, 07:1974, 07:2162, 07:2189, 07:2269, 07:2270, 07:2430, 07:2434, 08:384, 08:534, 08:607, 08:1977, 08:1990, 08:2061, *1032*

\tl_trim_spaces:n . 09:307, 11:301, 11:322, 11:334, 11:417, 11:441, 11:621, 11:876, 50:514, 51:111, 51:142, 52:41, 52:42, 07:244, 07:417, 07:535, 07:569, 07:1527, 07:1529, 07:2161, 07:2188, 07:2290, 07:2844, 07:2848, 07:2852, 07:2856, 08:413, *318*

\tl_trim_spaces_apply:nN . 07:673, 07:2293, 07:2317, 07:2407, 08:334

\tl_use:N . . . 28:288, 28:292, 28:327,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

28:331, 36:94, 36:107, 07:1370, 07:1458, 08:1056, 08:1119, 08:1812
\l_tmpa_tl *346*
tl internal commands:
\g__mark_first_marks_tl 48:148, 48:153, 48:182, 48:192, *958*
\g__mark_last_marks_tl 48:149, 48:154, 48:173, 48:192, *958*
\g__mark_new_top_tl 48:61, 48:126, 48:127, 48:133, 48:135
\l__mark_saved_parameters_tl 48:101, 48:114, 48:116
\g__mark_tmp_tl 48:61, 48:128, 48:130, 48:138, 48:159, 48:161, 48:171, 48:174, 48:223, 48:230, 48:235, 48:237, 48:238, *960*
\tmspace 38:214, *791*
\to 30:447, 30:449
\today 01:168, 01:172, 01:180, 01:183, 44:33, *1189*
token commands:
\c_space_token 07:2506
\token_if_active:NTF 07:1984, 07:2000
\token_if_cs:NTF 07:2217, *180*
\token_if_eq_catcode:NNTF 09:317, 07:1663
\token_if_eq_charcode:NNTF 07:412, 07:1899, 07:1931, 07:1942
\token_if_eq_meaning:NNTF 09:187, 09:262, 09:463, 52:454, 52:466, 07:682, 07:685, 07:693, 07:1292, 07:2520, 07:2968, 07:2970
\token_if_long_macro_p:N . . . 07:1386
\token_if_macro:NTF 09:84, 08:1204, 08:1230, *311*
\token_if_math_toggle_p:N 07:2363, 07:2385
\token_if_protected_long_macro_p:N 07:1387, 07:1396
\token_if_protected_macro:NTF 07:1155, 07:1180, 07:1347
\token_if_protected_macro_p:N 07:1395
\token_to_meaning:N 09:80, 09:460, 09:472, 09:536, 09:564, 16:41, 16:72
\token_to_str:N 09:79, 09:80, 09:102, 09:522, 09:529, 09:557, 09:560, 11:279, 11:280, 11:297, 11:304, 11:307, 11:314, 11:1079, 36:57, 52:241, 07:70, 07:78, 07:81, 07:417, 07:808, 07:1066, 07:1220, 07:1369, 07:1393, 07:1416, 07:1676, 07:1822, 07:1989, 07:2001, 07:2161, 07:2188, 07:2417, 07:2430, 07:2434, 07:2443, 07:2811, 07:2812, 07:2825, 07:2826, 07:2930, 07:2931, 07:2944, 07:2945, 08:381, *160*
\toks 16:40, 16:71, 16:80, 04:36, 28:668, 28:669, 28:679, 28:688, 02:31, 57:698, 02:63, 02:95, *412*
\toksdef 04:230, 02:46, 02:63, 02:95
\tokszero 04:230
\tolerance 24:672, 24:718, 24:733, 49:87, 49:95, 02:317
\top 30:327
\topfigrule 54:731, 54:2363
\topfraction 45:273, 54:2329
\topmargin 54:71, 54:628, 54:687
\TopMark 48:366, 48:486, *947*
\topmark 54:2247, 54:2256, *947*
\topsep 38:524, 39:1, 39:59, *803*
\topskip 20:59, 20:128, 20:183, 54:130, 02:418, *1227*
\totalheight 40:33, 40:34, 40:35
totalpages *1064*
trace commands:
trace_stack_levels 57:96
\tracefloats 54:1933
\tracefloatsoff 54:1933
\tracefloatvals 54:1933, *1258*
\traceoff *398*
\traceon *398*
\tracingall 02:557, *398*
\tracingassigns 02:575, 02:609, 02:626, 02:667
\tracingcommands 02:573, 02:591, 02:607, 02:616, 02:629, 02:653, 02:670, 02:347
\tracingfonts 26:17, 26:54, 26:58, 26:86, 26:118, 26:153, 26:194, 26:224, 26:238, 26:254, 26:260, 26:273, 26:280, 26:287, 26:292, 26:301, 26:314, 26:322, 26:325, *1194*
\tracinggroups 02:565, 02:600, 02:638, 02:678
\tracingifs . 02:566, 02:601, 02:637, 02:677
\tracinglostchars . . . 02:563, 02:586, 02:598, 02:617, 02:641, 02:661, 02:681, 24:682, 02:346, 02:558, *1258*
\tracingmacros . 02:572, 02:590, 02:606, 02:618, 02:640, 02:660, 02:680, 02:341
\tracingnesting 02:568, 02:603, 02:635, 02:675
\tracingnone 02:622, *1267*
\tracingoff 26:118, 26:322
\tracingon 26:119, 26:323, *1195*
\tracingonline . 02:628, 02:652, 02:669, 02:699, 24:681, 54:1919, 02:340, 02:552
\tracingoutput 02:632, 02:656, 02:673, 02:696, 02:345, 02:553

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\tracingpages .. 02:562, 02:585, 02:597, 02:617, 02:642, 02:662, 02:682, 02:344
\tracingparagraphs 02:564, 02:587, 02:599, 02:618, 02:639, 02:659, 02:679, 02:343
\tracingrestores 02:574, 02:592, 02:608, 02:618, 02:627, 02:658, 02:668, 02:348
\tracingscantokens 02:567, 02:582, 02:602, 02:636, 02:650, 02:676
\tracingstacklevels 02:570, 02:634, 02:349, 02:558, *24*
\tracingstats 02:584, 02:596, 02:616, 02:643, 02:663, 02:683, 57:2, 02:342, 02:561
\triangle 30:329
\triangleleft 30:365, 30:489
\triangleright 30:366, 30:489
\TrimSpaces 07:3001
trivlist (env.) 39:89
\trivlist 37:444, 37:494, 37:496, 37:527, 37:549, 38:508, 39:89, 41:78, 43:52, 43:54, 43:59, 43:61, *810*
TS1 commands:
  \TS1:? *728*
  \TS1:⟨*family*⟩ *728*
\tt *1199*
\ttdefault 29:12, 29:216, 29:400, 29:417, 29:441, 29:460, 29:488, 30:58
\ttfamily 29:10, 29:11, 29:410, 29:458, 29:459, 29:486, 29:487, 32:17, 37:571, *221*
ttfamily 29:420
\ttsubstdefault 30:20, 30:32, 33:32, 33:89
\twocolumn 54:201, *1233*
\twocolumn[...] *1228*
\twocolumn[] *446*
\typein *82*, *82*
\typein 33:1620, 33:1625, 06:31
\typeout *82*
\typeout 01:100, 03:21, 10:54, 10:57, 10:59, 10:61, 10:63, 10:66, 10:69, 03:38, 14:74, 20:201, 20:673, 20:674, 20:680, 20:714, 20:760, 20:774, 20:795, 01:156, 24:391, 25:478, 01:181, 01:183, 01:195, 29:138, 29:563, 29:744, 29:754, 29:764, 30:9, 30:132, 01:210, 01:217, 33:1216, 33:1253, 33:1260, 33:1413, 33:1416, 33:1419, 33:1617, 33:1618, 33:1619, 33:1622, 33:1623, 33:1624, 33:1629, 33:1633, 01:228, 01:241, 06:3, 01:254, 06:36, 06:43, 01:267, 46:8, 46:25, 48:326, 48:327, 48:331, 48:335, 48:339, 48:343, 48:347, 50:384, 50:400, 50:412, 01:305, 50:1589, 50:1796, 50:1799, 06:614, 06:615, 53:104, 53:115, 53:148, 06:649, 06:650, 06:655, 06:709, 06:710, 06:725, 06:727, 54:1922, 54:1934, 55:118, 57:276, 57:652, 57:659, 57:671, 57:672, 57:680, 01:57, *103*
\typeoutdetails 33:1216, 33:1217, 33:1220, 33:1226, 33:1228, 33:1232, 33:1233, 33:1239, 33:1257, 33:1258, 33:1412, 33:1627, 33:1629, 33:1633

U

\u 21:237, 21:386, 21:472, 21:589, 21:596, 21:616, 21:623, 21:754, 21:1242, 21:1317, 21:1318, 21:1333, 21:1334, 21:1343, 21:1344, 21:1357, 21:1358, 21:1359, 21:1383, 21:1384, 21:1409, 21:1410, 33:163, 33:194
\uccode 57:240, 57:248, 57:255, 57:257, 57:260, 57:262, 57:535, 57:543, 57:550, 57:552, 57:555, 57:557
\Ucharcat 29:624
\uchyph 02:370
\ulcdefault 25:526, 25:605
\ulcshape 25:526, 25:604, 25:698, 25:699, 29:549, 32:29, *565*
\Umathcode 04:30, 18:552, 30:15, 30:51, 30:58, 30:73, 33:159, 33:367, 37:577, 57:160, 57:351, 57:529, 02:127
\unboldmath 29:615
\UndeclareTextCommand 21:191, 33:115, 33:117, 33:119, 33:308, 33:585, 33:1127, 33:1128, 33:1129, 33:1130, 33:1131, 33:1132, *1209*
\undefined 22:119, 22:120, 22:176, 22:177, 22:178, 22:179, 22:180, 22:181, 01:12, 30:123, 30:124, 01:14, 35:40, 35:41, 35:57, 35:58, 35:59, 35:60, 01:20, 57:168, 57:180, 57:181, 57:201, 01:41, *1229*
\undefinedpagestyle 49:4, 49:8
\underbar 06:893, 06:914, 02:522
\underbrace 30:547
\underline *819*
\underline 40:551, 40:552, 02:522
\underscore *1235*
\unexpanded 32:47, 37:279, 37:315, 50:824, 50:1624, 50:1626, 06:632, 06:643, *1259*
\unhbox *1196*
\unhcopy ... 41:345, 42:741, 42:797, 02:524
\unicodedataline 04:143, 04:146, 04:160, 04:161, 04:162

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\UnicodeEncodingName 21:981, 21:987, 21:1038, 21:1044, 21:1048, 21:1059, 21:1063, 21:1079, 21:1080, 33:376, 33:377, 33:378, 33:379, 33:380, 33:381, 33:382, 33:383, 33:384, 33:385, 33:386, 33:387, 33:388, 33:389, 33:390, 33:391, 33:392, 33:393, *500*
\UnicodeFontFile 21:1036
\UnicodeFontName 21:1037
\UnicodeFontTeXLigatures . 21:993, 21:1033
\unicoderead 04:143, 04:157, 04:158, 04:159, 04:160, 04:165
uninstall 04:999
\unitlength 40:74, 40:85, 40:94, 40:104, 42:5, 42:29, 42:30, 42:32, 42:34, 42:42, 42:43, 42:44, 42:45, 42:60, 42:63, 42:73, 42:74, 42:84, 42:85, 42:93, 42:94, 42:107, 42:108, 42:119, 42:164, 42:176, 42:241, 42:256, 42:316, 42:318, 42:332, 42:340, 42:342, 42:357, 42:375, 42:377, 42:392, 42:397, 42:399, 42:414, 42:417, 42:422, 42:479, 42:480, 42:507, 42:508, 42:536, 42:537, 42:611, 42:627, 42:647, 42:653, 42:689, 42:690, 42:692, 42:693, 42:696, 42:697, 42:699, 42:700, 42:711, 42:712, 42:714, 42:715, 42:717, 42:718, 42:720, 42:721, 42:749, 42:750, 42:752, 42:753, 42:756, 42:757, 42:759, 42:760, 42:771, 42:773, 42:775, 42:777, 53:328, 57:149, *1061*
\unkern 24:701
\unless 04:151, 04:159, 04:161
\unlhd 29:732
\unpenalty 24:704, 24:708, 32:116, 37:543, 37:565, *409*
\unrhd 29:734
\unsetattribute *43*
\unsetattribute 04:82, 04:240
\unskip *1196*
\unvcopy 38:193, *949*
\Uparrow 30:591
\uparrow 30:585
\upbracefill 30:550, 30:567
\updefault 25:688, 29:21, 30:102, 30:109, 30:110, 30:118, 30:120, *668*
\Updownarrow 30:595
\updownarrow 30:589
\uplus 30:385
\uppercase *1209*
\upshape 21:442, 21:732, 21:802, 21:895, 25:686, 25:687, 29:19, 29:20, 29:549, 29:560, 29:593, 29:651, 32:24, 33:629, *565*
\Upsilon 30:311
\upsilon 30:293
use commands:
  \use:N 08:2289, 08:2359, 08:2375, 10:136, 10:143, 11:273, 11:489, 11:630, 11:756, 11:905, 28:295, 28:334, 36:73, 05:181, 48:112, 48:129, 48:142, 48:160, 48:179, 48:186, 48:232, 52:442, 52:446, 52:448, 52:465, 57:591, 07:804, 07:1455, 08:1629, 08:1762, 08:1763, 08:1803, *339*
  \use:n . . 08:2104, 08:2341, 08:2347, 08:2356, 09:105, 09:172, 09:175, 09:203, 09:212, 09:247, 09:250, 09:278, 09:284, 09:350, 09:383, 09:406, 09:425, 09:454, 09:484, 09:506, 09:513, 09:549, 11:294, 11:338, 11:372, 20:523, 48:111, 51:180, 07:186, 07:683, 07:686, 07:862, 07:1095, 07:1222, 07:1314, 07:1423, 07:2350, 07:2392, 07:2965, 07:2967, 07:2972, 08:238, 08:388, 08:511, 08:1152, 08:1627, 08:2070, *955*
  \use:nn 08:2120, 08:2679, 09:217, 09:289, 08:1733
  \use:nnn 07:2811, 07:2825, 07:2930, 07:2944, *188*
  \use_i:nn . . 08:2206, 08:2551, 49:41, 07:946, 07:1634, 07:2445, 07:2498, *346*
  \use_i:nnn 07:2423, 08:2069, *263*
  \use_i:nnnn 09:348, *319*
  \use_i_delimit_by_q_recursion_stop:nw 09:413, 09:432
  \use_i_delimit_by_q_stop:nw . 07:426
  \use_ii:nn 08:2557, 09:57, 09:139, 55:24, 55:34, 07:1536, 07:1556, 07:1561, 07:1566, 07:1571, 07:1816, 07:2025, 07:2038, 07:2077, 07:2095, 07:2120, 07:2153, 07:2180, 07:2444, 07:2976, 08:637, *169*
  \use_ii:nnn 07:1617, 07:2094, 07:2427, 07:2969
  \use_ii:nnnn 09:353
  \use_ii_i:nn 09:216, 09:288
  \use_ii_iii:nnn 52:216, 52:230, 52:242, 52:242
  \use_iii:nn *263*
  \use_iii:nnn 52:47, 07:2054, 07:2422, 07:2425, 07:2497, 07:2971

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\use_iii:nnnn 07:2053
\use_iv:nnnn 07:412, 07:418
\use_none:n 08:2208, 08:2297, 09:355, 09:413, 09:432, 10:7, 10:8, 48:305, 52:131, 52:134, 53:7, 55:3, 55:4, 55:23, 55:33, 55:121, 55:123, 07:95, 07:687, 07:1577, 07:1583, 07:1589, 07:1595, 07:1612, 07:1613, 07:1616, 07:1619, 07:1633, 07:1660, 07:1797, 07:2089, 07:2090, 07:2093, 07:2096, 07:2300, 08:7, 08:638, 08:1348, 08:1419, 08:1466, 08:1731, *283*
\use_none:nn 08:2102, 28:294, 28:333, 48:263, 07:1112, 07:1791, 07:2048, 07:2049, 07:2052, 07:2055, 07:2329, 08:2061, *961*
\use_none:nnn . . . 07:873, 07:1785, 07:2046, 07:2488, 07:2863, 07:2869, 07:2880, 07:2886, 08:744, 08:2067, *170*
\use_none:nnnn 08:2928, 07:1779, 07:2303
\use_none:nnnnn 07:1773, 08:1840, 08:1958
\use_none:nnnnnn 07:1767
\use_none:nnnnnnn 07:1761
\use_none:nnnnnnnnn 08:1348, 08:1840, 08:1958
\use_none_delimit_by_q_recursion_stop:w 07:340
\use_none_delimit_by_q_stop:w 07:2264, 07:2283, 07:2287
\usebox 40:195
\usecounter 39:249, 39:262
\UseExpandableTaggingSocket *1161*
\UseExpandableTaggingSocket . 55:7, *1161*
\usefont 21:1574, 24:80, 24:282, 24:677, 29:709, 33:7, 33:613, 37:581, *1228*
\UseHook 08:2784, 08:2902, 09:290, 09:557, 09:560, 20:316, 20:322, 20:327, 20:338, 20:377, 20:381, 20:384, 26:145, 29:263, 29:274, 29:286, 29:296, 29:317, 29:324, 29:337, 29:344, 29:403, 29:408, 29:413, 29:418, 29:663, 29:680, 37:248, 37:260, 37:263, 37:351, 37:355, 37:368, 37:371, 50:1003, 50:1007, 50:1031, 50:1035, 52:170, 52:171, 52:174, 52:175, 53:118, 53:170, 53:382, *226*
\UseHookWithArguments 08:2784, 09:219, 09:522, 09:529, 35:104, *196*
\UseInstance . 11:478, 11:1072, 11:1156, *348*
\UseLegacyTextSymbols . . . 33:576, 33:799
\UseName *78*
\UseName 05:177, 05:195
\UseOneTimeHook 08:2784, 08:2903, 20:15, 20:57, 20:70, 20:317, 20:321, 20:326, 20:339, 20:378, 20:380, 20:383, 37:14, 37:18, 37:29, 37:30, 37:32, 37:80, 37:84, 37:94, 37:95, 37:97, 50:1004, 50:1008, 50:1030, 50:1034, *212*
\UseOneTimeHookWithArguments 08:2784, *197*
\usepackage . . 50:676, 50:738, 50:1604, *200*
\UseRawInputEncoding 57:381, 57:437, 57:484
\UseSocket 10:170, 10:188, 35:123, 35:128, 35:133, *330*
\UseTaggingSocket *1161*
\UseTaggingSocket 17:40, 35:129, 55:7, 55:173, 55:209, 55:224, *1162*
\UseTemplate 11:92, 11:1156, *354*
\UseTextAccent 21:149, 21:150, 21:188, 33:110, 33:111, 33:113, 33:156, 33:158, 33:936, 33:1121, 33:1122, 33:1124, 33:1125, *476*
\UseTextSymbol 21:150, 21:186, 33:109, 33:352, 33:935, 33:1004, *476*
\usetikzlibrary *297*
\ushape *1236*

V

\v 21:238, 21:387, 21:471, 21:592, 21:593, 21:594, 21:598, 21:600, 21:603, 21:605, 21:607, 21:613, 21:619, 21:620, 21:621, 21:625, 21:627, 21:630, 21:632, 21:634, 21:640, 21:755, 21:1247, 21:1327, 21:1328, 21:1329, 21:1330, 21:1339, 21:1340, 21:1373, 21:1374, 21:1379, 21:1380, 21:1391, 21:1392, 21:1399, 21:1400, 21:1403, 21:1404, 21:1426, 21:1427, 21:1428, 21:1429, 21:1430, 21:1431, 21:1432, 21:1433, 21:1434, 21:1435, 21:1436, 21:1445, 21:1446, 21:1447, 21:1448, 21:1451, 21:1452, 33:165, 33:195
\vadjust 18:38, 18:60, 18:72, 18:101, 18:108, 18:388, 18:404, 18:422, 18:438, 45:201, 45:223, *404*
\valign 33:101, 33:927
\value *512*
\value 22:14, 22:47, 22:84, 36:235, 47:9, *1196*
\varbigtriangledown 30:369, 30:372
\varbigtriangleup 30:370, 30:371
\varepsilon 19:15, 30:298
\varphi 30:303
\varpi 30:300
\varrho 30:301
\varsigma 30:302
\vartheta 30:299

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\vbadness 53:212, 53:214, 53:263, 53:265, 54:2244, 02:319, *1076*
\vbox *1075*
vbox commands:
  \vbox_set:Nn 48:74
  \vbox_set_to_ht:Nnn . . 53:215, 53:266
  \vbox_to_zero:n 53:326
  \vbox_unpack:N 48:82, 48:95, 48:96, 48:385, 48:416, 53:226, 53:276
\vdash 30:411
\vdots 30:517
\vec 30:531
\vector 14:269, 42:233, 42:816, 42:833
\vee 30:374, 30:376
\verb . . . 37:575, 37:591, 37:604, 37:618, 37:630, 37:636, 37:642, 37:655, 37:667, 37:672, 37:680, 37:686, 37:696, 37:701, 37:714, 37:716, *1198*
\verb* *1266*
verbatim (env.) 37:569
\verbatim 37:569
verbatim* (env.) 37:584
\verbvisiblespace 37:575, 37:577, 37:591, 37:596, 37:606, 37:610, 37:620, 37:624, *1248*
\Vert 30:578, 30:580
\vert 30:583
\vfil 33:102, 33:105, 33:928, 33:931, 42:565, 42:577, 53:393, 53:405, 54:177, 54:196, 54:414, 54:461, 54:631, 54:690, 02:511
\vfilneg 02:511
\vfuzz 49:90, 49:97, 53:211, 53:213, 53:262, 53:264, 02:391, *1076*
\vglue 02:501
\vline 41:366
\voffset 02:407
\vphantom 21:497, 21:513, 38:75
\vrule . . 18:528, 21:293, 21:295, 21:502, 21:518, 26:190, 30:565, 30:566, 30:568, 30:569, 40:204, 40:206, 40:282, 40:289, 40:550, 40:603, 40:610, 41:186, 41:219, 41:347, 41:366, 42:227, 42:299, 42:302, 42:324, 42:333, 42:350, 42:359, 42:383, 42:391, 42:406, 42:413, 42:564, 42:577, 42:725, 42:781, 54:1855, 54:2264, 54:2297, 02:505, *1268*
\vsize *1231*
\vskip *1196*
\vspace 18:377, 18:447, 18:448, 18:449
\vsplit 54:384, 54:431, 54:2246, *957*

W

\wedge 30:373, 30:375
\whatsit 04:194, *43*
\widehat 30:534
\widetilde 30:533
\widowpenalties 02:106
\widowpenalty 24:691, 02:326
\width 40:30, *1216*
\wlog 04:6, 04:7, 04:8, 04:54, 37:45, 37:110, 50:414, 57:60, 02:40, 57:705, 02:145, 02:239, 02:254, 02:284, 02:299, 01:84, *1206*
\wp 30:319
\wr 30:389
\write *29*

X

\x 03:96, 03:99, 24:333, 24:334, 57:330, 57:332
\xdef *1193*
\XeTeXcharclass 24:665, 57:39, 57:47, 57:54, 57:67, 57:73, 57:82, 57:89
\XeTeXcharclassCL 57:173
\XeTeXcharclassCM 57:177
\XeTeXcharclassEX 57:174
\XeTeXcharclassID 57:171
\XeTeXcharclassIS 57:175
\XeTeXcharclassNS 57:176
\XeTeXcharclassOP 57:172
\XeTeXcharglyph 21:1034
\XeTeXdashbreakstate 57:273
\XeTeXglyph 21:1034
\XeTeXintercharclasses . . 57:167, 57:200
\XeTeXinterchartoks 57:168, 57:182, 57:183, 57:184, 57:185, 57:186, 57:187, 57:188, 57:189, 57:190, 57:191, 57:192, 57:193, 57:194, 57:195, 57:196, 57:201, 57:206, 57:207, 57:208, 57:209, 57:210, 57:211, 57:212, 57:213, 57:214, 57:215, 57:216, 57:217, 57:218, 57:219, 57:220
\XeTeXmathcode 57:161, 57:530
\XeTeXrevision 57:41
\XeTeXuseglyphmetrics . . . 57:270, 57:272
\XeTeXversion 29:623, 57:41
\Xi 30:308
\xi 30:288
xpos 36:243, *751*
\xpt *1199*
\xspaceskip 02:422
\xtxHanGlue 57:180, 57:204, 57:212, 57:213, 57:214, 57:215, 57:216, 57:217, 57:218, 57:219, 57:220

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\xtxHanSpace . . . 57:181, 57:205, 57:206, 57:207, 57:208, 57:209, 57:210, 57:211

Y

\y 03:97, 03:99
\year 03:14, 01:169, 50:1298, 50:1431, 50:1520, 02:386
ypos 36:243, *751*

Z

\Z 57:252, 57:526, 57:547
\z 57:243, 57:527, 57:538
\zeta 30:280

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfsstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx