LATEX3: from local to global
A brief history and recent developments

Will Robertson and Frank Mittelbach
This is all Will’s fault

From: Frank Mittelbach
Date: 27 June 2012

Will wrote:
> I’m still marking exams today, but I can put time
> aside tomorrow to produce some solid material.
> Because this is largely my fault, and to take the
> worry off you here, would you like to assume that
> I’ll produce the whole talk?

I sure would :-)
Outline

History

Programming layer

Internal interfaces

New features of expl3
expl3 timeline

1991  Original kernel
1998  ‘Modern’ beginning
2004  Morten
2008  Will
2009  Joseph
2011  Bruno

Paulo Roberto Massa Cereda
expl3 history

2004
MORTEM

1993
KERNEL

2007

2012
JOSEPH+BRUNO
News items

2009/1  Test suites and reconsidering interface.
2010/1  Rewrite of xparse and xtemplate.
2010/2  siunitx and fontspec, xhead; xcoffin; l3fp; I/O.
2011/1  LPPL OSI; Stack Exchange. l3fp; l3coffin.
2011/2  Big Bang; xgalley.
2012/1  Native drivers; l3regex. LDB.
2012/2  l3fp (exp.); @@.
Code frequency
Aspects to $\LaTeX{}$3

Multiple overlapping concepts.

- Programming layer expl3: fontspect/siunitx users (etc.) are ‘using’ $\LaTeX{}$3
- l3in2e: xparse / xtemplate / xcoffins
- Typesetting research: xgalley / xor
- Kernel:
  All of the above

$\LaTeX{}$3 is not monolithic.
Outline

History

Programming layer

Internal interfaces

New features of expl3
expl3 is the foundation

- Supports pdf\TeX and Xe\TeX and Lua\TeX. All three are in active use.

- Abstraction for general programming concepts; avoid having to remember ‘special tricks’ and reinvent the wheel.

- Conceived in the 90s; too slow then. Iterated and tested over the next decade; Consolidated in the last 5 years.

- People are using it!
  
github.com/jcsalomon/xpeek
expl3 modules

Programming:

- basics / expan / quark / prg / token

Data types:

- int / dim / skip / box
- tl / seq / clist / prop

‘Complex’ data types/modules:

- msg / keys / file
- fp / coffins / (regex?)
Increasing complexity

This one...

And there is much worse! We want to mitigate complexity.
One of my favourite tricks

In plain:

\let\foo\bar
\expandafter\let\csname foo\expandafter
   \endcsname\csname bar\endcsname

In expl3:

\cs_set_eq:NN \foo \bar
\cs_set_eq:cc \{foo\} \{bar\}
\cs_set_eq:Nc \foo \{bar\}
\cs_set_eq:cN \{foo\} \bar
More expansion

A difficult case:

`\foobart{abc}\{\expandme}`

How to expand `\expandme` once before this is seen by `\foobar`?
More expansion

A difficult case:

`\foobar\{abc}\{\expandme}\expandafter\foobar\expandafter\{\expandafter a\expandafter b\expandafter c\expandafter}%\expandafter{\expandme}`
More expansion

A difficult case:

`\xfoobar{abc}\.expandme`
More expansion

A difficult case:

\xfoobar{abc}{\expandme}

\def\xfoobar#1#2{@foobar{#2}{#1}}
More expansion

A difficult case:

\xfoobar{abc}{\expandme}

\def\xfoobar#1#2{\@foobar{#2}{#1}}
\def\@foobar#1#2{\expandafter\@@foobar\expandafter{#1}{#2}}
More expansion

A difficult case:

\xfoobar{abc}\{\expandme}

\def\xfoobar#1#2{\@foobar{#2}{#1}}
\def\@foobar#1#2{\expandafter\@@foobar\expandafter{#1}{#2}}
\def\@@foobar#1#2{\foobar{#2}{#1}}
More expansion

A difficult case:

\xfoobar{abc}{\expandme}

\def\xfoobar#1#2{\@foobar{#2}{#1}}
\def\@foobar#1#2{\expandafter\@@foobar\expandafter{#1}{#2}}
\def\@@foobar#1#2{\foobar{#2}{#1}}

Now \foobar finally receives the arguments with correct prior expansion
More expansion

In expl3:

\foo_bar:no \{abc\} {\expandme}

\cs_new:Npn \foo_bar:nn \#{1#2} {...}
More expansion

In expl3:

\foo_bar:no \{abc\} {\expandme}

\cs_new:Npn \foo_bar:nn #1#2 {...}
\cs_generate_variant:Nn \foo_bar:nn {no}
Outline

History

Programming layer

Internal interfaces

New features of expl3
A problem with \LaTeX\ 2ε

- 99% of the 2e kernel used by packages
- We cannot change the internals of the kernel!
- If only people didn’t mess around with internals.
- Only documented interfaces should be used.
How?

— Can’t (reasonably) enforce this in code.
— Can indicate this with code conventions.
— Can use \texttt{tl\_count:n}.
— \texttt{Cannot} use \texttt{\_\_tl\_count:n}!

(Or if you do don’t blame us!)

— \TeX{} does not have name-spacing.
— Let’s help with \texttt{docstrip}.
— Makes code easier to write.
— Helps enforce conventions.
\LaTeX 2\varepsilon example

\begin{figure}...
\end{figure}

All floats defined with \@float...
\end@float.

*Internally*, uses \@xfloat.
\cs_new_protected:Npn \seq_remove_duplicates:N #1
\{
  \__seq_remove_duplicates:NN \seq_set_eq:NN #1
\}

\cs_new_protected:Npn \__seq_remove_duplicates:NN #1#2
\{
  \seq_clear:N \l__seq_remove_seq
  \seq_map_inline:Nn #2
  \seq_if_in:NnF \l__seq_remove_seq {##1}
  \seq_put_right:Nn \l__seq_remove_seq {##1}
\}
#1 #2 \l__seq_remove_seq

Internal code in expl3 (.dtx)

\cs_new_protected:Npn \seq_remove_duplicates:N #1
 { 
   \@@_remove_duplicates:NN \seq_set_eq:NN #1
 }

\cs_new_protected:Npn \@@_remove_duplicates:NN #1#2
 { 
   \seq_clear:N \l_@@_remove_seq
   \seq_map_inline:Nn #2
   { 
     \seq_if_in:NnF \l_@@_remove_seq {##1}
     { \seq_put_right:Nn \l_@@_remove_seq {##1} }
   }#1 #2 \l_@@_remove_seq
 }
\tl\_count:n is ‘public’.
\@@\_count:n \equiv \__\_tl\_count:n is ‘internal’.
\__\_tl\_ to \@@\_ doesn’t save many letters here...
...but consider \__\_fontspec_!
‘Quick’ sorting in \l3sort

\clist_set:Nn \l_foo_clist { 3, 01, -2, 5, +1 }
\clist_sort:Nn \l_foo_clist
{ \int_compare:nNnTF { #1 } > { #2 }
{ \texttt{\textbackslash sort\_reversed:} }
{ \texttt{\textbackslash sort\_ordered:} }
}

Produces: ‘-2,+1,01,3,5’

\TeX\ by Topic has an example of a lexicographic comparison.
Expandable floating point

— \( \text{T\TeX} \) uses integer arithmetic for everything even dimension calculation in multiples of 1sp.
— Some have written maths modules for fixed point maths.
— Joseph wrote floating point maths.
— Bruno made it expandable!
Example

This code:

\usepackage{xparse, siunitx}
\ExplSyntaxOn
\NewDocumentCommand { \calcnum } { m } { \num { \fp_to_scientific:n {#1} } }
\ExplSyntaxOff
\calcnum { round ( 200 \pi \times \sin ( 2.3 ^ 5 ) , 2 ) }

Produces: \textit{6.2784} \times 10^2
Coffins example
Regular expressions in l3regex

JWZ:

Some people, when confronted with a problem, think “I know, I’ll use regular expressions.”
Now they have two problems.

But regular expressions are useful!
Advanced pattern matching.

\tl_set:Nn \l_my_tl { That~cat. }
\regex_replace_once:nnN { at } { is } \l_my_tl

→ ‘This cat.’

\tl_set:Nn \l_my_tl { This~cat~your~cat }\n\regex_replace_all:nNn { \w+ } { \0 , } \l_my_tl

→ ‘This, cat, your, cat,’
support tokens

- `'\c{begin} \cB. (\c[^BE].*) \cE.'` matches: `\begin{<anything without {}}>`
- `[a-oq-z\cC.]` matches any lowercase latin letter except p, as well as control sequences.

Available functions:

- Match (TF)
- Count
- Extract
- Split
- Replace
Poor man’s grep (for Windows users)

\ior_new:N \l_grep_stream

\cs_new:Npn \expl_grep:nn #1 #2
\{\ior_open:Nn \l_grep_stream {#2}\ior_str_map_inline:Nn \l_grep_stream
\{\regex_match:nnT {#1} {##1} { \texttt{##1}\ }\}\ior_close:N \l_grep_stream\}\}

\expl_grep:nn {\usepackage} {\jobname}
Poor man’s grep (for Windows users)

This is the output:

\usepackage{expl3, l3regex, l3sort}
\usepackage{calc, graphicx}
\usepackage{metalogo, fancyvrb}
\usepackage{fontspec, siunitx}
\usepackage{biblatex}
\usepackage{xparse, siunitx}
\expl_grep:nn {\usepackage} {\jobname}
\expl_grep:nn {\usepackage} {\jobname}
Conclusion

The Hitch-Hiker’s Guide to LaTeX3
by Andrew Stacey (TeX.sx Community Blog)

As with the original Hitch-Hiker’s Guide, this blog post won’t actually be all that useful to someone wanting to truly explore LaTeX3. It’s more of a “What I did on my holidays” kind of guide. I’ve recently had my first go at doing some coding with LaTeX3 and I thought it might be interesting to record my experiences.

...

Will I use LaTeX3 again? Absolutely. I wouldn’t choose it for a non-TeX situation, but if it’s something to be done within TeX then LaTeX3 is definitely high up on the list of choices. Do I expect an easy ride? Not at all. But at the end, I expect a sense of accomplishment not quite like coding in any other language.
Conclusion

- \LaTeX{}3 shouldn’t be thought as monolithic
- Programming layer is solid and being used by others
- Document interface layer for \LaTeX{} 2ε available
- Current team focus is on the typesetting foundation layer
  - font selection
  - output routines
  - page layout