# LaTeX3 Project Team

## A Modern Regression Test Suite for TeX Programming

Frank Mittelbach, Joseph Wright, Will Robertson

2014-07-28, TUG 2014 Portland, Oregon

# Outline

History

The Needs

Approach

A Time Line

The New Needs

Today's Issues

The New System

Live Demo

# How it began

— Don's approach when developing TEX

# How it began

— Don's approach when developing TEX
- Literate Programming:
  - Tangle and Weave
- Trip test for TEX
  - get into a devilish mindset

# How it began

- Don's approach when developing TEX
  - Literate Programming:
    - Tangle and Weave
  - Trip test for TEX
    - get into a devilish mindset

- My takeaway from that
  - Literate Programming:
    - `doc.sty` and and later `docstrip.tex`

# How it began

— Don's approach when developing TEX
- Literate Programming:
  - Tangle and Weave
- Trip test for TEX
  - get into a devilish mindset

— My takeaway from that
- Literate Programming:
  - `doc.sty` and and later `docstrip.tex`
- Ideas for regression tests for LATEX
  - ensure LATEX $2_\varepsilon$ maintains (most) of the typesetting functionality of LATEX 2.09 correctly
  - add tests for each bug fix
  - add tests for each interface (changed or unchanged)

# Excursion on doc and docstrip

- Requirements
  - It should be easily available
  - It should work on any platform TEX works

# Excursion on doc and docstrip

- Requirements
  - It should be easily available
  - It should work on any platform TEX works

- Initial ideas (doc):
  - Use a format that works both directly (as a LATEX package)
  - But could also be automatically formatted (with a suitable setup)

# Excursion on doc and docstrip

— Requirements
  - It should be easily available
  - It should work on any platform TEX works

— Initial ideas (doc):
  - Use a format that works both directly (as a LATEX package)
  - But could also be automatically formatted (with a suitable setup)

— Extensions (docstrip):
  - Strip out documentation lines to speed up loading
  - Provide features for generating several files from one source
  - Provide features for reorganizing code, adding licenses, etc.
  - Provide installation support into different directories

# How it continued (**Validating LaTeX 2.09**)

Writing test files for regression testing: checking bug fixes and improvements to verify that they don't have undesirable side effects; making sure that bug fixes really correct the problem they were intended to correct; testing interaction with various document styles, style options, and environments. We would like three kinds of validation files:

1. General documents.
2. Exhaustive tests of special environments/modules such as tables, displayed equations, theorems, floating figures, pictures, etc.
3. Bug files containing tests of all bugs that are supposed to be fixed (as well as those that are not fixed, with comments about their status).

A procedure for processing validation files has been devised; details will be furnished to anyone interested in this task.

Estimated time required: 2 to 3 weeks, could be divided up.

# Outline

History

The Needs

Approach

A Time Line

The New Needs

Today's Issues

The New System

Live Demo

# What was needed (back then in the ‘90s)?

— Verification

— Assembling a complex distribution

— Installation independence

— Full automation

# What was needed (back then in the ‘90s)?

- Verification
  - of coding (interfaces, functionality)
  - of documentation
- Assembling a complex distribution
- Installation independence
- Full automation

# What was needed (back then in the ‘90s)?

- Verification
  - of coding (interfaces, functionality)
  - of documentation
- Assembling a complex distribution
  - unpacking sources files and generating production files
  - typesetting and verifying documentation
  - adding license information
- Installation independence
- Full automation

# What was needed (back then in the ‘90s)?

- Verification
  - of coding (interfaces, functionality)
  - of documentation
- Assembling a complex distribution
  - unpacking sources files and generating production files
  - typesetting and verifying documentation
  - adding license information
- Installation independence
  - several developers, different OSes, different installations
- Full automation

# What was needed (back then in the ‘90s)?

- Verification
  - of coding (interfaces, functionality)
  - of documentation
- Assembling a complex distribution
  - unpacking sources files and generating production files
  - typesetting and verifying documentation
  - adding license information
- Installation independence
  - several developers, different OSes, different installations
- Full automation
  - as few manual steps as possible

# Outline

History

The Needs

Approach

A Time Line

The New Needs

Today's Issues

The New System

Live Demo

# What to test? & How to test?

— Typical problems with LaTeX code

- Many hidden dependencies
- Packages that hook into various layers of LaTeX
- Packages that overlay/replace macros

# What to test? & How to test?

— Typical problems with LaTeX code
- Many hidden dependencies
- Packages that hook into various layers of LaTeX
- Packages that overlay/replace macros

— Questions
- How do you verify correctness of typography (other than by looking at the `.dvi` or `.pdf`)?
- How do you verify correctness of interfaces?
- How do you avoid generating false positives?

# What to test? & How to test?

— Typical problems with LaTeX code
- Many hidden dependencies
- Packages that hook into various layers of LaTeX
- Packages that overlay/replace macros

— Questions
- How do you verify correctness of typography (other than by looking at the `.dvi` or `.pdf`)?
- How do you verify correctness of interfaces?
- How do you avoid generating false positives?

— Approach
- Use verified `.log` files for comparison
- Provide commands that add suitable data to the `.log` file
- Provide a mechanism to hide irrelevant details during comparison

# Output “relevant” data to the `.log`

- In general limit output to a suitable minimum
- Use `\typeout`, `\showthe`, etc. for “results”
- Avoid using `\tracingall` or other macro expansion tracing settings (like `\show\somecommand`) as this displays internal implementation details that we should not be concerned with (normally)
- A few `\tracing...` parameters may be useful, e.g., `\tracingparagraphs` or `\tracingpages`
- For typesetting verification try `\showlists`, `\showbox` or `\showoutput` but be careful that they do not generate too much output that is difficult to verify
- In some cases you may end up visually verifying the printed page and then freezing its symbolic representation via `\showoutput` or `\tracingoutput`

# .log file cleanup

- A TEX or LATEX .log file receives a lot of irrelevant data some of which may change from run to run (or from installation to installation)
- To reduce the "noise" we post-process each .log drop some lines and modify others
- The commands \START, \END, \OMIT and \TIMO are used in the source to define the areas in the .log used for comparison (data outside the regions is dropped)
- Further sanitizing
  - shortening file path info to avoid differences between installations
  - drop empty lines (different web2c implementations put different amounts in)
  - drop line numbers in "on line <num>"
  - …
- …but don't go too far

# Putting it all together

— `.lvt` are the test files; `.tlg` the expected test results

— A Makefile supports the various activity goals:

`check <name>` Without argument picks up all `.lvt` files, runs the tests, cleans the logs and compares them to the `tlg` files, otherwise runs only tests for `<name>`

`doc` Generates all documentation (`.dtx` etc.) and verifies that all of them compile successfully

`clean` Cleans source and temp directories from any intermediate files

`unpack` Unpacks sources files e.g., running `.ins` files

`install` Installs unpacked files into local TEX tree

`ctan` Runs all tests and generates a (set of) `.zip` files

`save <name> <engine>` Save the current test result for `<name>.lvt` as a new `.tlg` file (use `<name>.lvt-<engine>` if engine is given)

# Outline

History

The Needs

Approach

A Time Line

The New Needs

Today's Issues

The New System

Live Demo

# A Time Line

- ‘80s `trip` test for TEX
- 1992 `validate.tex` for LATEX
- 1993 Extensive test files written for verifying LATEX2.09 typesetting results are still valid with LATEX $2_\varepsilon$ (close to 300)
- 1994 Makefile system for building and testing the LATEX $2_\varepsilon$ distribution

# A Time Line

- '80s `trip` test for TEX
- 1992 `validate.tex` for LATEX
- 1993 Extensive test files written for verifying LATEX2.09 typesetting results are still valid with LATEX $2_\varepsilon$ (close to 300)
- 1994 Makefile system for building and testing the LATEX $2_\varepsilon$ distribution

- 1997 Again looking for volunteers to improve the regression tests for LATEX $2_\varepsilon$

  — not much luck unfortunately

# A Time Line

- '80s `trip` test for TEX
- 1992 `validate.tex` for LATEX
- 1993 Extensive test files written for verifying LATEX2.09 typesetting results are still valid with LATEX $2_\varepsilon$ (close to 300)
- 1994 Makefile system for building and testing the LATEX $2_\varepsilon$ distribution

- 1997 Again looking for volunteers to improve the regression tests for LATEX $2_\varepsilon$
  — not much luck unfortunately
- 2008 Replacing the Makefiles with Perl Cons — Unix only

# A Time Line

- '80s `trip` test for TEX
- 1992 `validate.tex` for LATEX
- 1993 Extensive test files written for verifying LATEX2.09 typesetting results are still valid with LATEX $2_\varepsilon$ (close to 300)
- 1994 Makefile system for building and testing the LATEX $2_\varepsilon$ distribution

- 1997 Again looking for volunteers to improve the regression tests for LATEX $2_\varepsilon$
  — not much luck unfortunately
- 2008 Replacing the Makefiles with Perl Cons — Unix only
- 2011 Add `.bat` files as alternative for Windows
  — not really a satisfying solution either

# A Time Line

- '80s `trip` test for TEX
- 1992 `validate.tex` for LATEX
- 1993 Extensive test files written for verifying LATEX2.09 typesetting results are still valid with LATEX $2_\varepsilon$ (close to 300)
- 1994 Makefile system for building and testing the LATEX $2_\varepsilon$ distribution

- 1997 Again looking for volunteers to improve the regression tests for LATEX $2_\varepsilon$
  — not much luck unfortunately
- 2008 Replacing the Makefiles with Perl Cons — Unix only
- 2011 Add `.bat` files as alternative for Windows
  — not really a satisfying solution either
- 2014 Develop new Lua-based system

# Outline

History

The Needs

Approach

A Time Line

The New Needs

Today's Issues

The New System

Live Demo

# The New Needs

— Support for multiple distributions

— Support for multiple Operating Systems

— Support for multiple “TEX-like” engines

# The New Needs

— Support for multiple distributions
  - core LaTeX 2$_\varepsilon$ and main packages
  - Babel (which had a different release cycle)
  - The evolving `expl3` language layer for LaTeX3
  - Third-party code

— Support for multiple Operating Systems

— Support for multiple "TeX-like" engines

# The New Needs

— Support for multiple distributions
- core LaTeX $2_\varepsilon$ and main packages
- Babel (which had a different release cycle)
- The evolving `expl3` language layer for LaTeX3
- Third-party code

— Support for multiple Operating Systems
- Linux / Unix
- Windows
- MacOS

— Support for multiple "TeX-like" engines

# The New Needs

- Support for multiple distributions
  - core LaTeX $2_\varepsilon$ and main packages
  - Babel (which had a different release cycle)
  - The evolving `expl3` language layer for LaTeX3
  - Third-party code

- Support for multiple Operating Systems
  - Linux / Unix
  - Windows
  - MacOS

- Support for multiple "TeX-like" engines
  - pdfTeX
  - XeTeX
  - LuaTeX

# Outline

History

The Needs

Approach

A Time Line

The New Needs

Today's Issues

The New System

Live Demo

# Today's Issues

- Flexibility
  - Different packages require different setups
  - Hardwiring structual decisions is a no-go

# Today's Issues

- Flexibility
  - Different packages require different setups
  - Hardwiring structual decisions is a no-go

- Engine output differences
  - Slight differences in log file data formatting often result in `.tlg` differences
  - Different capabilities result in different output (e.g., extra nodes in listings)
  - New engines have bugs that surface

# Today's Issues

- Flexibility
  - Different packages require different setups
  - Hardwiring structual decisions is a no-go
- Engine output differences
  - Slight differences in log file data formatting often result in `.tlg` differences
  - Different capabilities result in different output (e.g., extra nodes in listings)
  - New engines have bugs that surface
- Register numbers changing
  - `expl3` code additions use up additional registers invalidating existing `.tlg` files
  - Resolution: preallocate registers to allow adjusting for this without changes to the `.tlgs`

# Outline

History

The Needs

Approach

A Time Line

The New Needs

Today's Issues

The New System

Live Demo

# The New System

- Automation provided for
  - compilation
  - testing
  - generation of documentation
  - packaging for CTAN

# The New System

- Automation provided for
  - compilation
  - testing
  - generation of documentation
  - packaging for CTAN
- Support for
  - managing dependencies
  - executing all tests in full isolation

# The New System

- Automation provided for
  - compilation
  - testing
  - generation of documentation
  - packaging for CTAN
- Support for
  - managing dependencies
  - executing all tests in full isolation
- One setup script per module / bundle
  - available on any modern TEX installation
  - minimal content if conventions are followed
  - customization possible as needed

# The New System

- Automation provided for
  - compilation
  - testing
  - generation of documentation
  - packaging for CTAN
- Support for
  - managing dependencies
  - executing all tests in full isolation
- One setup script per module / bundle
  - available on any modern TEX installation
  - minimal content if conventions are followed
  - customization possible as needed
- Extensive documentation of capabilities

# Default directory layout

— Individual package (module)

```
mymodule/
    build.lua
    support/
    testfiles/
    source files (.dtx, .ins, etc)
```

— Bundle

```
mybundle/
    build.lua
    mymodule-1/
        build.lua
        support/
        testfiles/
        source files (.dtx, .ins, etc)
    mymodule-2/
      ...
```

# Sample build script (breqn)

```
#! /usr/bin/env texlua

 -- Build script for breqn

 module = "breqn"

 -- variable overwrites (if needed)

 unpackfiles = {"*.dtx"}
 excludefiles = {"*/breqn-abbr-test.pdf",
                 "*/eqbreaks.pdf"}
 unpackopts = "-interaction=batchmode"

 -- call standard script

 kpse.set_program_name ("kpsewhich")
 dofile (kpse.lookup ("l3build.lua"))
```

# Sample build scripts (bundle))

```
#! /usr/bin/env texlua

 -- Build script for mybundle

 bundle = "mybundle"

 kpse.set_program_name ("kpsewhich")
 dofile (kpse.lookup ("l3build.lua"))
```

---

```
#! /usr/bin/env texlua

 -- Build script for mymodule-1

 bundle = "mybundle"
 module = "mymodule-1"

 maindir = ".."

 kpse.set_program_name ("kpsewhich")
 dofile (kpse.lookup ("l3build.lua"))
```

# Configuration for more complex scenarios

```
-- Common settings for LaTeX3 development repo, used by l3build script

checkdeps   = checkdeps   or {maindir .. "/l3kernel",
                              maindir .. "/l3build"}
typesetdeps = typesetdeps or {maindir .. "/l3kernel"}
unpackdeps  = unpackdeps  or {maindir .. "/l3kernel"}

cmdchkfiles     = {"*.dtx"}
checksuppfiles  = {"etex.sty", "lualatexquotejobname.lua", "minimal.cls",
                   "regression-test.cfg"}
unpacksuppfiles = {"docstrip.tex"}

typesetcmds = "\\AtBeginDocument{\\DisableImplementation}"

... etc ...
```

Then used in `build.lua` like this:

```
dofile (maindir .. "/l3build/l3build-config.lua")
dofile (maindir .. "/l3build/l3build.lua")
```

# Outline

History

The Needs

Approach

A Time Line

The New Needs

Today's Issues

The New System

Live Demo

# Live Demo (comma lists)


- expl3 has a data type for manipuliating “comma lists”
- Offer that as a standalone interface for LaTeX $2_\varepsilon$
- Tasks:
  - write xclists.dtx and xclists.ins
  - add a simple build.lua
  - write some test files (.lvt)
  - use it for testing, documenting, distribution generation