LaTeX3 News

Issue 10, November 2016 (LaTeX release 2016-11-01)

There has been something of a gap since the last LaTeX3 News, but this does not mean that work has not been going on. The Team have been working on a number of areas, many of which reflect wider take-up of expl3. There have also been a number of significant new developments in the LaTeX3 "sphere" in the last two years.

l3build: Testing LaTeX packages

Testing has been an important part of the work of the team since they assumed maintenance of LaTeX over twenty years ago. Various scripts have been used over that time by the team for testing, but these have until recently not been set up for wider use.

With the general availability of LuaTeX it is now possible to be sure that every TeX user has a powerful general scripting language available: Lua. The team have used this to create a new general testing system for TeX code, l3build. This *is* designed to be used beyond the team, so is now available in TeX Live and MiKTeX and is fully documented. Testing using l3build makes use of a normalised version of the `.log` file, so can test any aspect of TeX output (e.g., by using `\showbox`) or its algorithms (by displaying results in the `.log`).

Part of the remit for creating l3build was to enable the team to work truly cross-platform and to allow testing using multiple TeX engines (earlier systems were limited to a single engine, normally ε-TeX). The new testing system means we are in a much stronger position to support a variety of engines (see below). It has also enabled us to give useful feedback on development of the LuaTeX engine.

As well as the core capability in testing, l3build also provides a "one stop" script for creating release bundles. The script is sufficiently flexible that for many common LaTeX package structures, setting up for creating releases will require only a few lines of configuration.

In addition to the documentation distributed with l3build, the project website [1, publications in 2014] contains some articles, videos and conference presentations that explain how to use l3build to manage and test any type of (LaTeX) package.

Automating expl3 testing

As well as developing l3build for local use, the team have also set up integration testing for expl3 using the Travis-CI system. This means that *every* commit to the LaTeX3 code base now results in a full set of tests being run. This has allowed us to significantly reduce the number of occasions where expl3 needs attention before being released to CTAN.

Automated testing has also enabled us to check that expl3 updates do not break a number of key third-party packages which use the programming environment.

Refining expl3

Work continues to improve expl3 both in scope and robustness. Increased use of the programming environment means that code which has to-date been under-explored is being used, and this sometimes requires changes to the code.

The team have extended formal support in expl3 to cover the engines pTeX and upTeX, principally used by Japanese TeX users. This has been possible in part due to the l3build system discussed above. Engine-dependent variations between pdfTeX, XeTeX, LuaTeX and (u)pTeX are now well-understood and documented. As part of this process, the "low-level" part of expl3, which saves all primitives, now covers essentially all primitives found in all of these engines.

The code in expl3 is now entirely self-contained, loading no other third-party packages, and can also be loaded as a generic package with plain TeX, etc. These changes make it much easier to diagnose problems and make expl3 more useful. In particular it can be used as a programming language for generic packages, that then can run without modifications under different formats!

The team have made a range of small refinements to both internals and expl3 interfaces. Internal self-consistency has also been improved, for example removing almost all use of `nopar` functions. Performance enhancements to the l3keys part of expl3 are ongoing and should result in significantly faster key setting. As keyval methods are increasingly widely used in defining behaviours, this will have an impact on compile times for end users.

Replacing `\lowercase` and `\uppercase`

As discussed in the last LaTeX3 News, the team have for some time been keen to provide new interfaces

which do not directly expose (or in some cases even use) the TEX primitives \lowercase and \uppercase. We have now created a series of different interfaces that provide support for the different conceptual uses which may flow from the primitives:

- For case changing text, \tl_upper_case:n, \tl_lower_case:n, \tl_mixed_case:n and related language-aware functions. These are Unicode-capable and designed for working with text. They also allow for accents, expansion of stored text and leaving math mode unchanged. At present some of the interface decisions are not finalised so they are marked as experimental, but the team expect the core concept to be stable.
- For case changing programming strings, \str_upper_case:n, \str_lower_case:n and \str_fold_case:n. Again these are Unicode-aware, but in contrast to the functions for text are not context-dependent. They are intended for caseless comparisons, constructing command names on-the-fly and so forth.
- For creating arbitrary character tokens, \char_generate:nn. This is based on the \Ucharcat primitive introduced by XETEX, but with the ideas extended to other engines. This function can be used to create almost any reasonable token.
- For defining active characters, \char_set_active_eq:NN and related functions. The concept here is that active characters should be equivalent to some named function, so one does not directly define the active character.

Extending xparse

After discussions at TUG2015 and some experimentation, the team have added a new argument type, e ("embellishment"), to xparse. This allows arguments similar to TEX primitive sub- and superscripts to be accepted. Thus

```
\DeclareDocumentCommand\foo{e{^_}}
  {\showtokens{"#1"}}
\foo^{Hello} world
```

will show

```
"{Hello}{-NoValue-}".
```

At present, this argument type is experimental: there are a number of models which may make sense for this interface.

A new \parshape model

As part of development of l3galley, Joseph Wright has proposed a new model for splitting up the functions of the \parshape primitive into three logical elements:

- Margins between the edges of the galley and the paragraph (for example an indented block);
- Cut-out sections running over a fixed number of lines, to support "in place" figures and so forth;
- Running or single-paragraph shape.

There are additional elements to consider here, for example whether lines are the best way to model the length of shaping, how to handle headings, cut-outs at page breaks, etc.

Globally optimized pagination of documents

Throughout 2016 Frank Mittelbach has worked on methods and algorithms for globally optimizing the pagination of documents including those that contain floats. Early research results have been presented at BachoTEX 2016, TUG 2016 in Toronto and later in the year at DocEng'16, the ACM Symposium on Document Engineering in Vienna. A link to the ACM paper (that allows a download free of charge) can be found on the project website [1]. The site also holds the speaker notes from Toronto and will host a link to a video of the presentation once it becomes available.

The framework developed by Frank is based on the extended functionality provided by LuaTEX, in particular its callback functions that allow interacting with the typesetting process at various points. The algorithm that determines the optimal pagination of a given document is implemented in Lua and its results are then used to direct the formatting done by the TEX engine.

At the current point in time this a working prototype but not yet anywhere near a production-ready system. However, the work so far shows great potential and Frank is fairly confident that it will eventually become a generally usable solution.

Looking forward

The LuaTEX engine has recently reached version 1.0. This may presage a stable LuaTEX and is likely to result in wider use of this engine in production documents.If that happens we expect to implement some of the more complex functionality (such as complex pagination requirements and models) only for LuaTEX.

References

[1] Links to various publications by members of the LATEX Project Team. https://www.latex-project.org/publications.