LaTeX3 News, Issues 1–12

Contents

Issue 1, February 2009 **3**
- Welcome to LaTeX3 . . . 3
- What currently exists . . . 3
- What's happening now . . . 3
- What's happening soon . . . 3
- What's happening later . . . 3

Issue 2, June 2009 **4**
- TeX Live and the expl3 code . . . 4
- Planned updates . . . 4
- New members . . . 4
- Some specifics . . . 4
- The next six months . . . 4

Issue 3, January 2010 **6**
- Happy New Year . . . 6
- Recent developments . . . 6
- Upcoming plans . . . 6
  - Packages to tackle . . . 7

Issue 4, July 2010 **8**
- expl3 in practice . . . 8
- New xpackages . . . 8
- Developments with expl3 . . . 8
- TUG 2010 reflections . . . 8

Issue 5, January 2011 **9**
- Happy new year . . . 9
- The LPPL is now OSI-approved . . . 9
- Reflections on 2010 . . . 9
- Current progress . . . 9
- Plans for 2011 . . . 9

Issue 6, June 2011 **10**
- The LaTeX3 Team expands . . . 10
- The 'Big Bang' . . . 10
- LaTeX3 on GitHub . . . 10
- Next steps . . . 11

Issue 7, February 2012 **12**
- After the 'Big Bang' . . . 12
- Deforming boxes . . . 12
- A TeX-based regex engine . . . 12
- xparse improves . . . 12
- The galley . . . 12
- Relationships between document items . . . 12

Issue 8, July 2012 **13**
- Extended floating point support . . . 13
- Regular expressions in TeX . . . 13
- Separating internal and external code . . . 13
  - Naming convention for internals . . . 14
- Continual revolution—the 'small bang' . . . 14

Issue 9, March 2014 **16**
- Hiatus? . . . 16
- expl3 in the community . . . 16
- Logo for the LaTeX3 Programming Language . . . 17
- Recent activity . . . 17
- Work in progress . . . 17
  - Uppercasing and lowercasing . . . 17
  - Space-skipping in xparse . . . 18
- . . . and for 2014 onwards . . . 18
- What can you do for The LaTeX Project? . . . 19
  - Programming Layer . . . 19
  - Design Layer . . . 19
  - Document Interface Layer . . . 20
  - In Summary . . . 20
  - And something else 20

Issue 10, November 2016 **21**
- l3build: Testing LaTeX packages . . . 21
- Automating expl3 testing . . . 21
- Refining expl3 . . . 21
- Replacing `\lowercase` and `\uppercase` . . . 21
- Extending xparse . . . 22
- A new `\parshape` model . . . 22
- Globally optimized pagination of documents . . . 22
- Looking forward . . . 22

Issue 11, February 2018 **23**
- Move of sources from Subversion to Git . . . 23
  - Version identifiers . . . 23
- expl3 updates and extensions . . . 23
  - l3sort moves to the kernel . . . 23
  - Boolean functions . . . 23
  - Revision of l3file . . . 23
  - Detection of `\cs_generate_variant:Nn` errors . . . 24
  - Accessing random data . . . 24
  - More powerful debugging . . . 24
  - Mark-up changes in l3doc . . . 24
- l3build updates . . . 24

Issue 12, January 2020 **25**
Introduction 25
New features in expl3 25
A new argument specifier: e-type 25
New functions 25
String conversion moves to expl3 26
Case changing of text 26
Notable fixes and changes 26
File name parsing 26
Message formatting 26
Key inheritance 26
Floating point juxtaposition 26
Changing box dimensions 26
More functions moved to stable 26
Deprecations 26
Internal improvements 26
Cross-module functions 26
The backend 27
Better support for (u)pTEX 27
Options 27
Engine requirements 27
Documentation 27
News 27
ChangeLog 27
Changes in xparse 27
New experimental modules 27
l3build changes 27

LaTeX3 News

Issue 1, February 2009

Welcome to LaTeX3

Momentum is again starting to build behind The LaTeX Project. For the last few releases of TeX Live, the experimental programming foundation for LaTeX3 has been available under the name expl3. Despite large warnings that the code would probably change in the future, we wanted to show that there was progress being made, no matter how slowly. Since then, some people have looked at the code, provided feedback, and — most importantly — actually tried using it. Although it is yet early days, we believe that the ideas behind the code are sound and there are only 'cosmetic improvements' that need to be made before expl3 is ready for the LaTeX package author masses.

What currently exists

The current LaTeX3 code consists of two main branches: the expl3 modules that define the underlying programming environment, and the 'xpackages', which are a suite of packages that are written with the expl3 programming interface and provide some higher-level functionality for what will one day become LaTeX3 proper. Both expl3 and parts of the xpackages are designed to be used *on top* of LaTeX 2$_\varepsilon$, so new packages can take advantage of the new features while still allowing to be used alongside many of the vast number of LaTeX 2$_\varepsilon$ packages on CTAN.

What's happening now

In preparation for a minor overhaul of the expl3 code, we are writing a comprehensive test suite for each module. These tests allow us to make implementation changes and then test if the code still works as before. They are also highlighting any minor shortcomings or omissions in the code. As the tests are being written, our assumptions about what should be called what and the underlying naming conventions for the functions and datatypes are being questioned, challenged, and noted for further rumination.

At the time of writing, we are approximately half-way through writing the test suite. Once this task is complete, which we plan for the first half of 2009, we will be ready to make changes without worrying about breaking anything.

What's happening soon

So what do we want to change? The current expl3 codebase has portions that date to the pre-LaTeX 2$_\varepsilon$ days, while other modules have been more recently conceived. It is quite apparent when reading through the sources that some unification and tidying up would improve the simplicity and consistency of the code. In many cases, such changes will mean nothing more than a tweak or a rename.

Beyond these minor changes, we are also re-thinking the exact notation behind the way functions are defined. There are currently a handful of different types of arguments that functions may be passed (from an untouched single token to a complete expansion of a token list) and we're not entirely happy with how the original choices have evolved now that the system has grown somewhat. We have received good feedback from several people on ways that we could improve the argument syntax, and as part of the upcoming changes to the expl3 packages we hope to address the problems that we currently perceive in the present syntax.

What's happening later

After the changes discussed above are finished, we will begin freezing the core interface of the expl3 modules, and we hope that more package authors will be interested in using the new ideas to write their own code. While the core functions will then remain unchanged, more features and new modules will be added as LaTeX3 starts to grow.

Some new and/or experimental packages will be changing to use the expl3 programming interface, including breqn, mathtools, empheq, fontspec, and unicode-math. (Which is one reason for the lack of progress in these latter two in recent times.) There will also be a version of the siunitx package written in expl3, in parallel to the current LaTeX 2$_\varepsilon$ version. These developments will provide improvements to everyday LaTeX users who haven't even heard of The LaTeX Project.

Looking towards the long term, LaTeX3 as a document preparation system needs to be written almost from scratch. A high-level user syntax needs to be designed and scores of packages will be used as inspiration for the 'out-of-the-box' default document templates. LaTeX 2$_\varepsilon$ has stood up to the test of time — some fifteen years and still going strong — and it is now time to write a successor that will survive another score.

LaTeX3 News

Issue 2, June 2009

TeX Live and the expl3 code

TeX Live 2009 is almost upon us, and the LaTeX3 team have been readying a new release of the experimental LaTeX3 code for this. Very dramatic changes have occurred since the last public release of the code in TeX Live 2008; no backwards compatibility has been maintained (as warned in the beginning of the documentation) but we believe the changes made are all much for the better. Almost every single part of expl3 has been scrutinized, resulting in a far more coherent code base.

The expl3 code is now considered to be much more stable than it was before; a comprehensive test suite has been written that helps to ensure that we don't make any mistakes as we change things in the future. In the process of writing the test suite, many minor bugs were fixed; we recommend such test suites for all similar developmental projects! Some small underlying changes are still expected in the expl3 code, but major, disruptive, changes aren't planned.

Planned updates

Until now, the last update to CTAN of the expl3 bundle was for TeX Live 2008. Now that work on the code is happening on a semi-steady basis, we plan to keep updates rolling out to CTAN more frequently. This will allow anyone who wishes to experiment with the new code to use the TeX Live or MiKTeX updaters to install a recent version without having to 'check out' the SVN repository and install the packages manually.

New members

We didn't say anything about it in the last status update, but Joseph Wright and Will Robertson are now members of the LaTeX Team. They have been working fairly exclusively on the expl3 code.

It's worth repeating that LaTeX 2ε is essentially frozen in order to prevent any backwards compatibility problems. As desirable as it is to benefit from the new features offered by new engines XeTeX and LuaTeX, we cannot risk the stability of production servers running older versions of LaTeX 2ε which will inevitably end up processing documents written into the future.

LaTeX3 will not be inheriting the same restraints, so stay tuned.

Some specifics

Morten Høgholm will be presenting the recent changes in much more detail at TUG 2009. Here are some quick specifics for those interested. New code written and broad changes made to the expl3 modules:

More logical function names Many function names that were hold-outs from the TeX naming system have been changed to fit into the more logical scheme of expl3; e.g., `\def:Npn` and `\let:NN` are now `\cs_set:Npn` and `\cs_set_eq:NN`.

Defining functions and conditionals Much thought was put into new ways to define functions and conditionals with a minimum of code. See `\cs_set:Nn` and `\prg_set_conditional:Nnn`.

Smart comparisons Comparisons can be made much more easily now, with familiar notation such as `\intexpr_compare_p:n{ #1+3 != \l_tmpa_int }`.

Data from variables A new function argument specifier `V` has been added for extracting information from variables of different types, without needing to know the underlying variable structure. Some other tidy-ups on the argument specifiers offered, partially as a result of the addition of this new one.

l3msg New module to deal with communication between LaTeX3 code and the user (info messages, warnings, and errors), including message filtering partially inspired by the silence package.

The next six months

Having overhauled the expl3 code, we now plan to perform an analogous process with the foundations of the xpackages. These are the higher-level packages that will provide the basic needs such as control of the page layout and rich document-level interaction with the user. As the groundwork for this layer of the document processing matures, we will be able to start building more packages for a LaTeX3 kernel; these packages will also be usable on top of LaTeX 2ε and serve as broadly customisable templates for future document design.

As gaps in the functionality offered by expl3 are found (in some cases, we know that they exist already), the programming layer will be extended to support our needs. In other cases, wrappers around TeX functions that can be more usefully handled at a higher level will be written.

In terms of what we're planning to work on next, three xpackages will take the focus of our attention.

xbase 'xbase' is actually two packages: xparse and template. These contain code for, respectively, defining new document commands (such that a user would use; e.g., `\section`, `\makebox`, ...) and for handling keyval lists for user input and document specification. xparse was presented at TUG 1999[1] and Lars Hellström wrote some notes on template in 2000[2]. Functionality coverage for these packages is good but concepts need a good "airing". There are various approaches taken for keyval input, some more recent than the template code, so there are some alternatives to evaluate.

galley2 Sophisticated handling for constructing paragraphs and other document elements. Morten spoke on this at TUG 2008[3]. Design needs to be revisited after some stress testing.

xor This is the LaTeX3 output routine for splitting the galley into page and sub-page sized chunks. Ideas and code need work to move to "production ready" status. Early developments with this package were published by Frank in 2000[4].

Expect to hear again from us at Christmas. If you'd like to discuss any of these ideas, please join us on the LATEX-L mailing list[5].

[1] http://www.latex-project.org/papers/tug99.pdf
[2] http://www.latex-project.org/papers/template-notes.pdf
[3] http://river-valley.tv/the-galley-module/
[4] http://www.latex-project.org/papers/xo-pfloat.pdf
[5] http://www.latex-project.org/code.html

LaTeX3 News

Issue 3, January 2010

Happy New Year

Welcome to the holiday season edition of 'news of our activities' for the LaTeX3 team.

Recent developments

The last six months has seen two significant releases in the LaTeX3 code. In the CTAN repository for the xpackages,[1] you'll find two items of interest:

- A revised version of xparse; and
- The new package xtemplate, a re-implementation of template with a new syntax.

Special thanks to Joseph Wright who handled the implementations above almost single-handedly (with lots of input and feedback from other members of the team and members of the LATEX-L mailing list).
These two packages are designed for the LaTeX package author who wishes to define document commands and designer interfaces in a high-level manner.

xparse This package allows complex document commands to be constructed with all sorts of optional arguments and flags. Think of how `\newcommand` allows you to create a command with a single optional argument and xparse is a generalisation of that idea.

xtemplate This package requires more explanation. Xtemplate is designed to separate the logical information in a document from its visual representation. 'Templates' are constructed to fulfil individual typesetting requirements for each set of arguments; to change the look of a certain part of a document, instantiations of templates can be swapped out for another without (a) having to change the markup of the source document, or (b) having to edit some internal LaTeX macro. LaTeX 2_ε packages, such as geometry or titlesec, already provide parameterized interfaces to specific document elements. For example, one may use titlesec to change the layout of a `\section`: one modifies its layout parameters via `\titleformat` and `\titlespacing`. In a way, such packages define a template for a specific document element and some manipulation commands to instantiate it. However, the moment the intended layout is not achievable with one package you are on your own: either you have to resort to low-level programming or find some other high-level package which, of course, comes with its own set of conventions and manipulation commands.
The xtemplate package can be thought of a generalization of such ideas. It provides a uniform interface for defining and managing templates for any kind of document element and most importantly provides a uniform interface for instantiating the layout.
Thus the designer activity of defining or modifying a document class is limited to selecting the document elements that should be provided by the class (e.g., `\chapter`, `\section` `\footnote`, lists, ...), selecting appropriate "named" templates for each of them, and instantiating these templates by specifying values for their layout parameters. If a desired layout can't be achieved with a given template a different template for the same document element can be selected.
Programming is only necessary if no suitable template for the intended layout is available. It is then that a LaTeX programmer has to build a new template that supports the layout requirements. Once this task is complete, the template may be added to the selection of templates that designers and users may choose from to define or adjust document layouts seamlessly.
This is a slight gloss over the complexities of the package itself, which you can read about in the documentation. We've tried to document xtemplate clearly but we'd love feedback on whether the ideas make sense to you.
As an addendum to the introduction of xtemplate, the older template package will be retired in the near future. To our knowledge there is only a single package on CTAN that uses template, namely xfrac, and members of the LaTeX team are in the process of switching this package over to xtemplate. If you have any private code that uses template, please let us know!

Upcoming plans

Having announced the updated xparse and the new xtemplate, the next stage of development will revolve around using these two systems in the other components of the xpackages, feeding back our experience in practise with the original ideas behind the designs and evaluating if the packages are meeting our expectations.

[1] http://mirror.ctan.org/tex-archive/macros/latex/contrib/xpackages/

Packages to tackle

xhead The first work will be to create a new xpackage (probably called xhead), for typesetting section headings and other document divisions. Section headings are one of the more complex areas to work with, so the work should stress xtemplate enough to know if its current design is sufficient for most needs. Nothing has been released yet, but we'll announce further developments on the LATEX-L mailing list[2] in the mean time.

galley We also need to give galley the same treatment as xparse and xtemplate have already had. That is, we have an older implementation (in fact two) that needs some work before we're ready to release it to CTAN. The galley package is used to place material into the vertical list while typesetting but before page breaks occur. Since it works at such a low level, it is important to solidify this package before writing higher level design templates.
An issue we have to face is that to achieve best results, galley cannot be used in concert with LaTeX 2_ε code. This could limit its usefulness, and we may decide that it's better to scale back the features we're attempting, to allow better interoperability for existing packages and documents. More work remains before we can decide between these options.

[2] For details, see http://www.latex-project.org/code.html

LaTeX3 News

Issue 4, July 2010

Now that we're back from the TeX Users Group conference in San Francisco, it's time to discuss what's been going on over the last six months. Due to some extra travel plans after the conference, this issue is slightly late in coming out.

expl3 in practice

Joseph Wright and Will Robertson have both released significant new versions of their packages, resp., siunitx and fontspec. These have been re-written in the LaTeX3 programming language expl3, which we have discussed here previously. Using expl3 for production code has been very successful, both in demonstrating that the concepts are sound and highlighting areas that still need some attention.

In the case of fontspec, expl3 programming is being used to target LaTeX running on either XeTeX and LuaTeX. In the latter case, the package is a mixture of Lua code and expl3 code; Will presented the unicode-math package at TUG 2010, which is developed in the same style.

New xpackages

Frank Mittelbach has started to work on a new experimental LaTeX3 package xhead that provides templates for one of the most complex areas of document design: section headings and document divisions. This is the beginning of an ambitious idea to map out the requirements for typesetting most documents currently processed with LaTeX.

One of the challenges here is providing a "natural" design language for describing the two-dimensional spatial relationships of objects participating in the design, e.g., the placement of a heading number in relation to the heading title, a possible sub-title, etc. In answer to this challenge Frank developed the xcoffin package, which he presented at TUG 2010. It is designed as a high-level interface for placing and aligning boxes on a page, allowing a 'designer's approach' for indicating the positional relationship between boxes. (A 'coffin' is a box with handles.) As an example, it is possible to represent ideas such as 'align the lower-left corner of box A with the upper-right corner of box B after rotating it ninety degrees', without having to calculate the intermediate positions.

We expect a future version of xcoffin (after some further work on its interface layer and its internal implementation) to play a major role in all packages providing layout templates for higher-level document objects, such as table of contents designs, floats, etc.

Finally, Joseph Wright has begun work with the current 'galley' packages, producing the new, minimal, xgalley based on xfm-galley as a testbed for what we need and what will work.

Developments with expl3

Meanwhile, Joseph's *also* been writing a new floating-point calculation module, called l3fp, for expl3. This module allows manipulation and calculation of numbers with a much larger range than TeX allows naturally. The l3fp module has already been utilised in the xcoffin code for calculatations such as coordinate rotations and intersection points of vectors.

The modules l3io and l3file have been revised, rethinking the way that read and write streams are dealt with. TeX has a hard limit of sixteen input and output streams open at any one time, and the new implementation for expl3 provides more flexibility in how they are allocated; there's now much less chance of running into a '`No room for a new \read`' (or `\write`) error.

Sometimes we discuss ideas for expl3 that *don't* end up making it into the final code. One example of this is the concept of having 'local registers' for integers, boxes, and so on, that do not survive outside of the group they are defined in (in contrast to Plain TeX and LaTeX, where allocators such as `\newcount` and `\newbox` are always global). Despite the scope for some small benefit, we decided that the extra complexity that the additional functions required, in both syntax and documentation, was not justified.

TUG 2010 reflections

Our interpretation of the broad themes discussed at the conference are that TeX-based systems are still thriving and there are some big problems to solve with robust solutions to transform LaTeX source, including mathematics, into a form such as HTML. While there are big pushes for standardising various aspects of the LaTeX syntax, we also believe that it is LaTeX's very flexibility—its inherently non-standardised markup—that has allowed it to survive for so many years. There is a delicate trade-off here between moving forward into more standards-based territory while also retaining the extensibility of the third-party package system.

LaTeX3 News

Issue 5, January 2011

Happy new year

Seasons greetings for 2011! As the previous news issue was released late, this season's issue will be shorter than usual.

The LPPL is now OSI-approved

We are happy to report that earlier this year the LaTeX Project Public License (LPPL) has been approved by the OSI as an open source licence.[1] Frank Mittelbach will be publishing further details on this news in a retrospective on the LPPL in an upcoming TUGboat article.

Reflections on 2010

We are pleased to see the continued development and discussion in the TeX world. The LaTeX ecosystem continues to see new developments and a selection of notable news from the second half of last year include:

June The TUG 2010 conference was held very successfully in San Francisco; videos, slides, and papers from LaTeX3 Project members are available from our website.[2]

Aug. The TeX Stack Exchange[3] question & answer website was created and has since grown quickly. At time of writing, some 2800 people have asked 2600 questions with 5600 answers total, and 2200 users are currently visiting daily.

Sept. TeX Live 2010 was released: each year the shipping date is earlier; the production process is becoming more streamlined and we congratulate all involved for their hard work. One of the most notable new components of TeX Live 2010 includes the 'restricted shell escape' feature to allow, among other things, automatic EPS figure conversion for pdfLaTeX documents.

Oct. TLContrib[4] was opened by Taco Hoekwater as a way to update a TeX Live installation with material that is not distributable through `tlmgr` itself. Such material includes executables (e.g., new versions of LuaTeX), non-free code, or test versions of packages.

Nov. Philipp Lehman released the first stable version of biblatex. One of the most ambitious LaTeX packages in recent memory, biblatex is a highly flexible package for managing citation cross-referencing and bibliography typesetting. In 'beta' status for some years now, reaching this point is a great milestone.

Dec. LuaTeX 0.65. We are happy to see LuaTeX development steadily continuing. LaTeX users may use LuaTeX with the `lualatex` program. Like `xelatex`, this allows LaTeX documents to use multilingual OpenType fonts and Unicode text input.

Current progress

The expl3 programming modules continue to see revision and expansion; we have added a LuaTeX module, but expl3 continues to support all three of pdfLaTeX, XeLaTeX, and LuaLaTeX equally.

The l3fp module for performing floating-point arithmetic has been extended and improved. Floating point maths is important for some of the calculations required for complex box typesetting performed in the new 'coffins' code. The l3coffin module has been added based on the original xcoffins package introduced at TUG 2010 as reported in the last news issue; this code is now available from CTAN for testing and feedback.

We have consolidated the l3int and l3intexpr modules (which were separate for historical purposes); all integer/count-related functions are now contained within the 'int' code and have prefix `\int_`. Backwards compatibility is provided for, but eventually we will drop support for the older `\intexpr_` function names.

Plans for 2011

In the following year, we plan to use the current LaTeX3 infrastructure to continue work in building high-level code for designing LaTeX documents using the xtemplate package. Our first priority is to look at section headings and document divisions, as we see this area as one of the most difficult, design-wise, of the areas to address. From there we will broaden our scope to more document elements.

We will also do some low-level work on the 'galley', which is the code that LaTeX3 uses to build material for constructing pages, and we will continue to extend expl3 into a more complete system from which we can, one day, create a pure LaTeX3 format.

[1] http://www.opensource.org/licenses/lppl
[2] http://www.latex-project.org/papers/
[3] http://tex.stackexchange.com
[4] http://tlcontrib.metatex.org/

LaTeX3 News

Issue 6, June 2011

A key aim of releasing 'stable' LaTeX3 material to CTAN is to allow users to benefit from new ideas *now*, and also to raise the profile of usable LaTeX3 ideas. This is clearly being successful, with xparse being of particular utility to end users. This increase in interest has been particularly notable on the new TeX.SX Q&A site.

The LaTeX3 Team expands

Raising interest in LaTeX3 developments has inevitably led to feedback on cases where the code base has required attention. It has also attracted new programmers to using LaTeX3 ideas, some more than others! Bruno Le Floch has over the past few months made many useful contributions to LaTeX3, and we are very pleased that he has recently joined The LaTeX Project. Bruno has taken a particular interest in improving the performance and reliability of the expl3 language. This has already resulted in new implementations for the `prop` and `seq` data types. At the same time, he has identified and fixed several edge-case issues in core expl3 macros.

The 'Big Bang'

In parallel to Bruno's improvements, Joseph Wright initiated a series of 'Big Bang' improvements to LaTeX3. The aim of the Big Bang was to address a number of long-standing issues with the LaTeX3 code base. Development has taken place over many years, with the status of some of the resulting code being less than clear, even to members of The LaTeX Project! At the same time, different conventions had been applied to different parts of the code, which made reading some of the code rather 'interesting'. A key part of the Big Bang has been to address these issues, cleaning up the existing code and ensuring that the status of each part is clear.

The arrangement of LaTeX3 code is now the same in the development repository and on CTAN, and splits the code into three parts.

l3kernel The core of LaTeX3, code which is expected to be used in a LaTeX3 kernel in more or less the current form. Currently, this part is made up of the LaTeX3 programming layer, expl3.

l3packages LaTeX 2$_\varepsilon$ packages making use of LaTeX3 concepts and with stable interfaces. The xparse and xtemplate packages are the core of this area. While many of the *ideas* explored here may eventually appear in a LaTeX3 kernel, the interfaces here are tied to LaTeX 2$_\varepsilon$.

l3experimental LaTeX 2$_\varepsilon$ packages which explore more experimental LaTeX3 ideas, and which may see interface changes as development continues. Over time, we expect code to move from this area to either l3kernel or l3packages, as appropriate.

In addition to these release areas, the development code also features a l3trial section for exploring code ideas. Code in l3trial may be used to improve or replace other parts of LaTeX3, or may simply be dropped!

As well as these improvements to the *code* used in LaTeX3, much of the documentation for expl3 has been made more precise as part of the Big Bang. This means that `source3.pdf` is now rather longer than it was previously, but also should mean that many of the inaccuracies in earlier versions have been removed. Of course, we are very pleased to receive suggestions for further improvement.

LaTeX3 on GitHub

The core development repository for LaTeX3 is held in an SVN repository, which is publicly viewable *via* the Project website. However, this interface misses out on some of the 'bells and whistles' of newer code-hosting sites such as GitHub and BitBucket. We have therefore established a mirror of the master repository on GitHub[1]. This is kept in synchronisation with the main SVN repository by Will Robertson (or at least by his laptop!).

The GitHub mirror offers several useful features for people who wish to follow the LaTeX3 code changes. GitHub offers facilities such as highlighted differences and notification of changes. It also makes it possible for non-Team members to submit patches for LaTeX3 as 'pull requests' on GitHub.

As well as offering a convenient interface to the LaTeX3 code, the GitHub site also includes an issue database[2]. Given the very active nature of LaTeX3 development, and the transitory nature of many of the issues, this provides a better approach to tracking issues than the main LaTeX bug database[3]. Developers and users are

[1] http://github.com/latex3/svn-mirror
[2] http://github.com/latex3/svn-mirror/issues
[3] http://www.latex-project.org/bugs.html

therefore asked to report any issues with LaTeX3 code *via* the GitHub database, rather than on the main Project homepage. Discussion on the LaTeX-L mailing list is also encouraged.

Next steps

The 'Big Bang' involves making a number of changes to expl3 function names, and is likely to break at least some third-party code. As a result, the updates will not appear on the TeX Live 2011 DVD release, but will instead be added to TeX Live once regular updates restart (probably August).

Bruno is working on a significant overhaul of the l3fp floating-point unit for LaTeX3. He has developed an approach which allows expandable parsing of floating-point expressions, which will eventually allow syntax such as

```
\fp_parse:n { 3 * 4 ( ln(5) + 1 ) }
```

This will result in some changes in the interface for floating-point numbers, but we feel that the long-term benefit is worth a small amount of recoding in other areas.

Joseph has completed documentation of the xgalley module, and this is currently being discussed. Joseph is hoping to move on to implement other more visible ideas based on the xtemplate concept over the next few months.

LaTeX3 News

Issue 7, February 2012

After the 'Big Bang'

The last LaTeX3 News gave details of the 'Big Bang', in which the team have revised the layout and coverage of the LaTeX3 codebase. This process has made the status of different modules clearer, so that both the team themselves and everyone else know what is going on.

The 'Big Bang' changes were not shipped to CTAN until after the TeX Live 2011 freeze, as we did not want to end up with a DVD containing badly broken code. The update went to CTAN soon after TeX Live 2011 shipped, and has now propagated around the world. The new package naming (l3kernel, l3packages and l3experimental) has caused some surprises for a small number of users, but there have not been any major issues with the changes at the code level.

The 'Big Bang' has attracted attention from programmers outside of the LaTeX3 team, with useful feedback arriving on the `LaTeX-L` list and TeX.sx, in particular. One area that this has highlighted is the need to document carefully when changes to the 'stable' parts of the LaTeX3 codebase occur. All changes to l3kernel now come with an explicit date for the change in the documentation, which means that programmers can check exactly when the features they want were introduced.

Another key part of supporting LaTeX3 use beyond the team is making it easy to check on the version of LaTeX3 installed. To support that, the file date of the main expl3 package is now set each time there is a release of the LaTeX3 material to CTAN. This means that the LaTeX 2_ε `\@ifpackagelater` test can be used reliably to detect if the installed version of LaTeX3 is going to supply the functions that a programmer is using.

Deforming boxes

Additions to both the LaTeX3 stable material and more experimental modules continue. Joseph Wright has been working on adding 'native' drivers for LaTeX3 to support box transformations. These allow box rotation, clipping and scaling with the drivers `dvips`, `xdvipdfmx` and direct PDF output.

The development of clipping support for the `xdvipdfmx` driver has also allowed us to suggest improvements to the LaTeX 2_ε graphics drivers, enabling clipping with the XeTeX engine.

A TeX-based regex engine

Bruno Le Floch has been improving the efficiency and robustness of a number of LaTeX3 functions. Most notably, he has created a purely TeX-based regular expression (regex) system for LaTeX3. This is currently experimental, but is already proving useful and will hopefully stabilise over the coming months.

Bruno's regex system works with all of the supported engines (pdfTeX, XeTeX and LuaTeX). He has implemented the core ideas of standard regex systems, along with some TeX-specifics to allow matching and replacing the content of token lists by category code.

xparse improves

The xparse module has been overhauled, making the internal code more efficient and adding additional argument types. This has also allowed us to deal with a number of internal bugs, meaning that argument grabbing is now more reliable.

The argument grabbers themselves have been reworked so that in the event of an error, the user will normally get a meaningful message from TeX rather than one pointing to xparse internal function names. This should help in tracking down erroneous input in real documents.

The galley

As detailed in the last issue, work on the galley module has been continuing. Discussion of Joseph's reimplementation of the galley concepts highlighted some important areas to work on, with the nature of the template concept being particularly significant.

More work is still needed to finalise the galley concepts, but it is clear that some of this will require feedback from other areas. Joseph therefore hopes to finish work on the current round of galley improvements by the end of February, and to return to them once some other areas have been addressed.

Relationships between document items

The TUG2011 meeting took place in October in India. Frank Mittelbach spoke there about ideas for describing the design relationship between document elements. These ideas allow a document designer to specify the design of a document element based on its context within a document, and progress in this area will likely lead to an extension in the xtemplate system.

LaTeX3 News

Issue 8, July 2012

Extended floating point support

Bruno Le Floch has been re-writing the floating point module to function in an 'expandable' manner. This allows floating point calculations to be computed far more flexibly and efficiently than before. The expandable nature of the new code allows its use inside operations such as writing to external files, for which previously any such calculations would have to be precalculated before any of the writing operations began. Bruno's work on the floating point module has been officially released into the main SVN repository for `l3kernel`; TeX Live 2012 will contain the 'old' code for stability while this year is spent testing the new code in production environments and ironing out any wrinkles. Here's a neat example as suggested in the documentation, which produces '$6.278\,400\,000\,000\,000 \times 10^2$':

```
\usepackage{xparse, siunitx}
\ExplSyntaxOn
\NewDocumentCommand { \calcnum } { m }
  { \num { \fp_to_scientific:n {#1} } }
\ExplSyntaxOff

\calcnum {
  round ( 200 pi * sin ( 2.3 ^ 5 ) , 2 )
}
```

This feature is invaluable for simple (and not-so-simple) calculations in document and package authoring, and has been lacking a robust solution for many years. While LuaLaTeX can perform similar tasks within its Lua environment, the floating point support is written using the `expl3` programming language only, and is thus available in pdfLaTeX and XeLaTeX as well.

Regular expressions in TeX

As if expandable floating point support wasn't enough, Bruno has also written a complete regular expression engine in `expl3` as well. Many reading this will be familiar with the quote attributed to Jamie Zawinski:

> *Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems.*

And as humorous as the saying is, it's still fair to say that regular expressions are a great tool for manipulating streams of text. We desperately hope that people will *not* start using the regex code to do things like parse XML documents; however, for general search–replace duties, there's never been anything like `l3regex` for the LaTeX world. As a trivial example, there are 0 instances of the word 'We' or 'we' in this document (including those two). This value is counted automatically in two lines of code.

And again, it is available for pdfLaTeX and XeLaTeX users as well as LuaLaTeX ones; it also bears noting that this provides an easy solution for applying regular expression processing to LaTeX documents and text data even on the Windows operating system that does not have native support for such things.

Separating internal and external code

LaTeX packages are written by a wide range of package authors and consist of code and commands for various purposes. Some of these commands will be intended for use by document authors (such as `\pbox` from the `pbox` package); others are intended for use by other package writers (such as `\@ifmtarg` from the `ifmtarg` package). However, a large portion of them are internal, i.e., are intended to be used only within the package or within the LaTeX kernel and should not be used elsewhere. For example, `\@float` is the LaTeX kernel interface for floats to be used in class files, but the actual work is done by a command called `\@xfloat` which should not be used directly. Unfortunately the LaTeX 2_ε language makes no clear distinction between the two, so it is tempting for programmers to directly call the latter to save some "unnecessary" parsing activity.

The downside of this is that the "internal" commands suddenly act as interfaces and a reimplementation or fix in any of them would then break add-on packages as they rely on internal behavior. Over the course of the last twenty years probably 80% of such "internal" commands have found their way into one or another package. The consequences of this is that nowadays it is next to impossible to change anything in the LaTeX 2_ε kernel (even if it is clearly just an internal command) without breaking something.

In LaTeX3 we hope to improve this situation drastically by clearly separating public interfaces (that extension packages can use and rely on) and private functions and variables (that should not appear outside of their module). There is (nearly) no way to enforce this without severe computing overhead, so we implement it only through a naming convention, and some support mechanisms. However, we think that this naming convention is easy to understand and to follow, so that we are

confident that this will be adopted and provides the desired results.

Naming convention for internals

We've been throwing around some ideas for this for a number of years but nothing quite fit; the issue comes down to the fact that TEX does not have a 'name-spacing' mechanism so any internal command needs to have a specific prefix to avoid clashing with other packages' commands. The prefix we have finally decided on for expl3 code is a double underscore, such that functions like `\seq_count:N` are intended for external use and `\__seq_item:n` is an internal command that should not be used or relied upon by others.

All this is well and good, but it can be inconvenient to type long prefixes such as `\__seq_` before all command names, especially in a package for which nearly *all* package functions are internal.

We therefore also extended DocStrip slightly by adding a 'shorthand' for internal package prefixes. Commands and variables in `.dtx` code may now contain `@@` which is expanded to the function prefix when the `.sty` file is extracted. As an example, writing

```
%@@=seq
\cs_new:Npn \@@_item:n
  ...
```

is equivalent to

```
\cs_new:Npn \__seq_item:n
  ...
```

There are clear advantages to this syntax. Function names are shorter and therefore easier to type, and code can still be prototyped using the `@@` syntax (e.g., pasting code between a `.dtx` file and a regular `.tex` document). Most importantly, it is explicitly clear from the code source which commands are intended to be used externally and which should be avoided.

We hope that this syntax will prove popular; in our initial experiments we think it works very well. In fact we found a good number of smaller errors when being forced to think about what is internal and what is an external function.

Continual revolution—the 'small bang'

In addition to the major additions introduced above, Frank Mittelbach has been examining `expl3` with a fresh eye to resolve any outstanding issues in the consistency or logic of the names of functions.

We are very mindful of the fact that for people to find `expl3` a useful tool, it must have a stable interface. This said, there are still some musty corners that we can show where people simply haven't been using certain functions. In select cases, we're re-assessing whether all of the (sometimes esoteric) odds and ends that have been added to `expl3` really belong; in other cases, it's now clear that some naming or behaviour choices weren't correct the first time around.

To address this tarnish, we're carefully making some minor changes to parts of the `expl3` interface and we'd like to allay any fears that `expl3` isn't stable. The `expl3` language now offers a wide range of functions plus their variants, and we're talking about changing but a very small percentage of these, and not common ones at that. We don't want it to become a mess, so we think it's better to tidy things up as we go. Follow the `LaTeX-L` mailing list for such details as they arise.

dropped/replaced:
The LaTeX3 codebases ranges between these two extremes; the packages in `l3packages` are largely the former while the modules composing expl3 are largely the latter type. In both cases, the 'external' commands (whether for document author or package writer) are usually defined in terms of other internal package commands that should not be used by anyone else, but often when reading the internal package code it's not always clear which is which.
For LaTeX3 we are experimenting with an extension to the DocStrip mechanism to provide a clear distinction between internal and external package commands.

LaTeX3 News

Issue 9, March 2014

Contents

Hiatus?

Well, it's been a busy couple of years. Work has slowed on the LaTeX3 codebase as all active members of the team have been — shall we say — busily occupied with more pressing concerns in their day-to-day activities.

Nonetheless, Joseph and Bruno have continued to fine-tune the LaTeX3 kernel and add-on packages. Browsing through the commit history shows bug fixes and improvements to documentation, test files, and internal code across the entire breadth of the codebase.

Members of the team have presented at two TUG conferences since the last LaTeX3 news. (Has it really been so long?) In July 2012, Frank and Will travelled to Boston; Frank discussed the challenges faced in the past and continuing to the present day due to the limits of the various TeX engines; and, Frank and Will together covered a brief history and recent developments of the expl3 code.

In 2013, Joseph and Frank wrote a talk on complex layouts, and the "layers" ideas discussed in LaTeX3; Frank went to Tokyo in October to present the work. Slides of and recordings from these talks are available on the LaTeX3 website.

These conferences are good opportunities to introduce the expl3 language to a wider group of people; in many cases, explaining the rationale behind why expl3 looks a little strange at first helps to convince the audience that it's not so weird after all. In our experience, anyone that's been exposed to some of the more awkward expansion aspects of TeX programming appreciates how expl3 makes life much easier for us.

expl3 in the community

While things have been slightly quieter for the team, more and more people are adopting expl3 for their own use. A search on the TeX Stack Exchange website for either "`expl3`" or "`latex3`" at time of writing yield around one thousand results each.

In order to help standardise the prefixes used in expl3 modules, we have developed a registration procedure for package authors (which amounts to little more than notifying us that their package uses a specific prefix, which will often be the name of the package itself). Please contact us via the `latex-l` mailing list to register your module prefixes and package names; we ask that you avoid using package names that begin with `l3...` since expl3 packages use this internally. Some authors have started using the package prefix `lt3...` as a way of indicating their package builds on expl3 in some way but is not maintained by the LaTeX3 team.

In the prefix database at present, some thirty package prefixes are registered by fifteen separate individuals (unrelated to The LaTeX Project — the number of course grows if you include packages by members of the team). These packages cover a broad range of functionality:

acro Interface for creating (classes of) acronyms

hobby Hobby's algorithm in PGF/TiKZ for drawing optimally smooth curves.

chemmacros Typesetting in the field of chemistry.

classics Traditional-style citations for the classics.

conteq Continued (in)equalities in mathematics.

ctex A collection of macro packages and document classes for Chinese typesetting.

endiagram Draw potential energy curve diagrams.

enotez Support for end-notes.

exsheets Question sheets and exams with metadata.

lt3graph A graph data structure.

newlfm The venerable class for memos and letters.

fnpct Interaction between footnotes and punctuation.

GS1 Barcodes and so forth.

hobete Beamer theme for the Univ. of Hohenheim.

kantlipsum Generate sentences in Kant's style.

lualatex-math Extended support for mathematics in LuaLaTeX.

media9 Multimedia inclusion for Adobe Reader.

pkgloader Managing the options and loading order of other packages.

substances Lists of chemicals, etc., in a document.

withargs Ephemeral macro use.

xecjk Support for CJK documents in XeLaTeX.

xpatch, **regexpatch** Patch command definitions.

xpeek Commands that peek ahead in the input stream.

xpinjin Automatically add pinyin to Chinese characters

zhnumber Typeset Chinese representations of numbers

zxjatype Standards-conforming typesetting of Japanese for XeLaTeX.

Some of these packages are marked by their authors as experimental, but it is clear that these packages have been developed to solve specific needs for typesetting and document production.

The expl3 language has well and truly gained traction after many years of waiting patiently.

A logo for the LaTeX3 Programming Language

To show that expl3 is ready for general use Paulo Cereda drew up a nice logo for us, showing a hummingbird (agile and fast — but needs huge amounts of energy) picking at "l3". Big thanks to Paulo!

Recent activity

LaTeX3 work has only slowed, not ground to a halt. While changes have tended to be minor in recent times, there are a number of improvements worth discussing explicitly.

1. Bruno has extended the floating point code to cover additional functions such as inverse trigonometric functions. These additions round out the functionality well and make it viable for use in most cases needing floating point mathematics.
2. Joseph's refinement of the experimental galley code now allows separation of paragraph shapes from margins/cutouts. This still needs some testing!
3. For some time now expl3 has provided "native" drivers although they have not been selected by default in most cases. These have been revised to improve robustness, which makes them probably ready to enable by default. The improvements made to the drivers have also fed back to more "general" LaTeX code.

Work in progress

We're still actively discussing a variety of areas to tackle next. We are aware of various "odds and ends" in expl3 that still need sorting out. In particular, some experimental functions have been working quite well and it's time to assess moving them into the "stable" modules, in particular the l3str module for dealing with catcode-twelve token lists more commonly known in expl3 as *strings*.

Areas of active discussion including issues around uppercasing and lowercasing (and the esoteric ways that this can be achieved in TeX) and space skipping (or not) in commands and environments with optional arguments. These two issues are discussed next.

Uppercasing and lowercasing

The commands `\tl_to_lowercase:n` and `\tl_to_uppercase:n` have long been overdue for a good hard look. From a traditional TeX viewpoint, these commands are simply the primitive `\lowercase` and `\uppercase`, and in practice it's well known that there are various limitations and peculiarities associated with them. We know these commands are good, to one extent or another, for three things:

1. Uppercasing text for typesetting purposes such as all-uppercase titles.
2. Lowercasing text for normalisation in sorting and other applications such as filename comparisons.
3. Achieving special effects, in concert with manipulating `\uccode` and the like, such as defining commands that contain characters with different catcodes than usual.

We are working on providing a set of commands to achieve all three of these functions in a more direct and easy-to-use fashion, including support for Unicode in LuaLaTeX and XeLaTeX.

Space-skipping in xparse

We have also re-considered the behaviour of space-skipping in xparse. Consider the following examples:

```
\begin{dmath}          \begin{dmath}[label=foo]
[x y z] = [1 2 3]      x^2 + y^2 = z^2
\end{dmath}            \end{dmath}
```

In the first case, we are typesetting some mathematics that contains square brackets. In the second, we are assigning a label to the equation using an optional argument, which also uses brackets. The fact that both work correctly is due to behaviour that is specifically programmed into the workings of the `dmath` environment of breqn: spaces before an optional argument are explicitly forbidden. At present, this is also how commands and environments defined using xparse behave. But consider a pgfplots environment:

```
\begin{pgfplot}
  [
    % plot options
  ]
  \begin{axis}
    [
      % axis options
    ]
    ...
  \end{axis}
\end{pgfplot}
```

This would seem like quite a natural way to write such environments, but with the current state of xparse this syntax would be incorrect. One would have to write either of these instead:

```
\begin{pgfplot}%          \begin{pgfplot}[
  [                          % plot options
   % plot options           ]
  ]
```

Is this an acceptable compromise? We're not entirely sure here — we're in a corner because the humble `[` has ended up being part of both the syntax and semantics of a LaTeX document.

Despite the current design covering most regular use-cases, we have considered adding a further option to xparse to define the space-skipping behaviour as desired by a package author. But at this very moment we've rejected adding this additional complexity, because environments that change their parsing behaviour based on their intended use make a LaTeX-based language more difficult to predict; one could imagine such behaviour causing difficulties down the road for automatic syntax checkers and so forth. However, we don't make such decisions in a vacuum and we're always happy to continue to discuss such matters.

. . . and for 2014 onwards

There is one (understandable) misconception that shows up once in a while with people claiming that

$$\text{expl3} = \text{LaTeX3}.$$

However, the correct relation would be a subset,

$$\text{expl3} \subset \text{LaTeX3},$$

with expl3 forming the Core Language Layer on which there will eventually be several other layers on top that provide

- higher-level concepts for typesetting (Typesetting Foundation Layer),
- a designer interface for specifying document structures and layouts (Designer Layer),
- and finally a Document Representation Layer that implements document level syntax.

Of those four layers, the lowest one — expl3 — is available for use and with xparse we have an instance of the Document Representation Layer modeled largely after LaTeX 2_ε syntax (there could be others). Both can be successfully used within the current LaTeX 2_ε framework and as mentioned above this is increasingly happening.

The middle layers, however, where the rubber meets the road, are still at the level of prototypes and ideas (templates, ldb, galley, xor and all the good stuff) that need to be revised and further developed to arrive at a LaTeX3 environment that can stand on its own and that is to where we want to return in 2014.

An overview on this can be found in the answer to "What can *I* do to help The LaTeX Project?" on Stack Exchange,[1] which is reproduced below in slightly abridged form. This is of course not the first time that we have discussed such matters, and you can find similar material in other publications such as those at http://latex-project.org; e.g., the architecture talk given at the TUG 2011 conference.

[1] http://tex.stackexchange.com/questions/45838

What can you do for The LaTeX Project?

By Frank Mittelbach

My vision of LaTeX3 is really a system with multiple layers that provide interfaces for different kinds of roles. These layers are

- the underlying engine (some TeX variant)
- the programming layer (the core language, i.e., expl3)
- the typesetting foundation layer (providing higher-level concepts for typesetting)
- the typesetting element layer (templates for all types of document elements)
- the designer interface foundation layer
- the class designer layer (where instances of document elements with specific settings are defined)
- document representation layer (that provides the input syntax, i.e., how the author uses elements)

If you look at it from the perspective of user roles then there are at least three or four roles that you can clearly distinguish:

- The Programmer (template and functionality provider)
- The Document Type Designer (defines which elements are available; abstract syntax and semantics)
- The Designer (typography and layout)
- The Author (content)

As a consequence The LaTeX Project needs different kinds of help depending on what layer or role we are looking at.

The "Author" is using, say, list structures by specifying something like `\begin{itemize} \item` in his documents. Or perhaps by writing `<ul> ... </ul>` or whatever the UI representation offers to him.

The "Document Type Designer" defines what kind of abstract document elements are available, and what attributes or arguments those elements provide at the author level. E.g., he may specify that a certain class of documents provides the display lists "enumerate", "itemize" and "description".

The "Programmer" on the other hand implements templates (that offer customizations) for such document elements, e.g., for display lists. What kind of customization possibilities should be provided by the "Programmer" is the domain of the "Document Designer"; he drives what kind of flexibility he needs for the design. In most cases the "Document Designer" should be able to simply select templates (already written) from a template library and only focus on the design, i.e., instantiating the templates with values so that the desired layout for "itemize" lists, etc., is created.

In real life a single person may end up playing more than one role, but it is important to recognise that all of them come with different requirements with respect to interfaces and functionality.

Programming Layer

The programming layer consists of a core language layer (called expl3 (EXP erimental L aTeX 3) for historical reasons and now we are stuck with it `:-)`) and two more components: the "Typesetting Foundation Layer" that we are currently working on and the "Typesetting Element Layer" that is going to provide customizable objects for the design layer. While expl3 is in many parts already fairly complete and usable the other two are under construction.

Help is needed for the programming layer in

- helping by extending and completing the regression test suite for expl3
- helping with providing good or better documentation, including tutorials
- possibly helping in coding additional core functionality — but that requires, in contrast to the first two points, a good amount of commitment and experience with the core language as otherwise the danger is too high that the final results will end up being inconsistent

Once we are a bit further along with the "Typesetting Foundation Layer" we would need help in providing higher-level functionality, perhaps rewriting existing packages/code for elements making use of extended possibilities. Two steps down the road (once the "Designer Layer" is closer to being finalized) we would need help with developing templates for all kinds of elements.

In summary for this part, we need help from people interested in programming in TeX and expl3 and/or interested in providing documentation (but for this a thorough understanding of the programming concepts is necessary too).

Design Layer

The intention of the design layer is to provide interfaces that allow specifying layout and typography styles in a declarative way. On the implementation side there are a number of prototypes (most notably xtemplate and the recent reimplementation of ldb). These need to be unified into a common model which requires some more experimentation and probably also some further thoughts.

But the real importance of this layer is not the implementation of its interfaces but the conceptual view

of it: provisioning a rich declarative method (or methods) to describe design and layout. I.e., enabling a designer to think not in programs but in visual representations and relationships.

So here is the big area where people who do not feel they can or want to program TEX's bowels can help. What would be extremely helpful (and in fact not just for LATEX3) would be

- collecting and classifying a *huge* set of layouts and designs
  - designs for individual document elements (such as headings, TOCs, etc)
  - document designs that include relationships between document elements
- thinking about good, declarative ways to specify such designs
  - what needs to be specified
  - to what extent and with what flexibility

I believe that this is a huge task (but rewarding in itself) and already the first part of collecting existing design specifications will be a major undertaking and will need coordination and probably a lot of work. But it will be a huge asset towards testing any implementations and interfaces for this layer later on.

Document Interface Layer

If we get the separation done correctly, then this layer should effectively offer nothing more than a front end for parsing the document syntax and transforming it into an internal standardised form. This means that on this layer one should not see any (or not much) coding or computation.

It is envisioned that alternative document syntax models can be provided. At the moment we have a draft solution in xparse. This package offers a document syntax in the style of LATEX 2_ε, that is with *-forms, optional arguments in brackets, etc., but with a few more bells and whistles such as a more generalized concept of default values, support for additional delimiters for arguments, verbatim-style arguments, and so on. It is fairly conventional though. In addition when it was written the clear separation of layers wasn't well-defined and so the package also contains components for conditional programming that I no longer think should be there.

Bottom line on what is needed for this layer is to

- think about good syntax for providing document content from "the author" perspective
- think about good syntax for providing document content from an "application to typesetting" perspective, i.e., the syntax and structure for automated typesetting where the content is prepared by a system/application rather than by a human

These two areas most likely need strict structure (as automation works much better with structures that do not have a lot of alternative possibilities and shortcuts, etc.) and even when just looking at the human author a lot of open questions need answering. And these answers may or may not be to painfully stick with existing LATEX 2_ε conventions in all cases (or perhaps with any?).

None of this requires coding or expl3 experience. What it requires is familiarity with existing input concepts, a feel for where the pain points currently are and the willingness to think and discuss what alternatives and extensions could look like.

In Summary

Basically help is possible on any level and it doesn't need to involve programming. Thoughts are sprinkled throughout this article, but here are a few more highlights:

- help with developing/improving the core programming layer by
  - joining the effort to improve the test suite
  - help improving the existing (or not existing) documentation
  - joining the effort to produce core or auxiliary code modules
- help on the design layer by
  - collecting and classifying design tasks
  - thinking and suggesting ways to describe layout requirements in a declarative manner
- help on shaping the document interface layer

These concepts, as well as their implementation, are under discussion on the list `latex-l`.[2] The list has only a fairly low level of traffic right now as actual implementation and development tasks are typically discussed directly among the few active implementors. But this might change if more people join.

And something else . . .

The people on the LATEX3 team are also committed to keeping LATEX 2_ε stable and even while there isn't that much to do these days there remains the need to resolve bug reports (if they concern the 2e core), provide new distributions once in a while, etc. All this is work that takes effort or remains undone or incomplete. Thus here too, it helps the LATEX3 efforts if we get help to free up resources.

[2] Instructions for joining and browsing archives at: http://latex-project.org/code.html

LaTeX3 News

Issue 10, November 2016

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.

LaTeX3 News

Issue 11, February 2018

Contents

Move of sources from Subversion to Git **23**
Version identifiers 23

expl3 updates and extensions **23**
l3sort moves to the kernel 23
Boolean functions 23
Revision of l3file 23
Detection of `\cs_generate_variant:Nn` errors 24
Accessing random data 24
More powerful debugging 24
Mark-up changes in l3doc 24

l3build updates **24**

Move of sources from Subversion to Git

The LaTeX team have used a variety of version control systems over the life of the LaTeX3 sources. For a long time we maintained the LaTeX3 sources in Subversion (`svn`) but also provided a read-only clone of them on GitHub using SubGit from TMate Software [1] to synchronize the two repositories—a solution that worked very well.

We have now retired the Subversion repository and completely moved over to Git, with the master LaTeX3 repository hosted on GitHub: https://github.com/latex3/latex3. This new approach means we are (slowly) adopting some new approaches to development, for example branches and accepting pull requests.

Version identifiers

Following this change, we have removed Subversion `$Id` lines from the LaTeX3 sources. At present, we will be retaining `\GetIdInfo` as there are several possible use cases. The LaTeX3 sources now have only release date strings as identifiers. However, the team recommend that package authors include version information directly in `\ProvidesExplPackage` (or similar) lines.

expl3 updates and extensions

Work has continued on the codebase over the last year, with both small changes/fixes and more substantial changes taking place. The following sections summarise some of the more notable changes.

l3sort moves to the kernel

Sorting is an important ability, and for some time the team have provided a stand-alone l3sort to support this. The functionality has seen wide take up, and so has now been integrated directly into the kernel. This took place in parallel with some interface changes to "round out" the code.

Boolean functions

For some time, the team have been aware that boolean expressions can fail in certain circumstances, leading to low-level errors. This is linked to two features of the long-standing `\bool_if:n(TF)` function: expandable operation and short-circuit evaluation.

Addressing that has meant two changes: altering `\bool_if:n(TF)` to *always* evaluate each part of the expression, and introducing new short-circuit functions without the issue. The latter are `lazy` in expl3 terms:

- `\bool_lazy_all:n(TF)`
- `\bool_lazy_and:nn(TF)`
- `\bool_lazy_any:n(TF)`
- `\bool_lazy_or:nn(TF)`

These new, stable functions are now the recommended way of handling boolean evaluations. Package authors are encouraged to employ these new functions as appropriate.

Revision of l3file

Large parts of l3file have been revised to give a better separation of path/file/extension. This has resulted in the addition of a number of new support functions and variables.

At the same time, new experimental functions have been added to utilise a number of useful primitives in pdfTeX: `\file_get_mdfive_hash:nN`, `\file_get_size:nN` and `\file_get_timestamp:nN`. Currently, XeTeX does not support getting file size/timestamp information: this is available in other engines.

Paralleling these changes, we have added (experimental) support for shell escape to the l3sys module, most notably `\sys_shell_now:n`. A range of test booleans are also available to check whether shell escape is enabled.

Detection of `\cs_generate_variant:Nn` *errors*

The ability to generate variants is an important feature of expl3. At the same time, there are crucial aspects of this approach that can be misunderstood by users. In particular, the requirement that variants map correctly to an underlying `N`- or `n`-type base function is sometimes misunderstood.

To help detect and correct these cases, `\cs_generate_variant:Nn` now carries out error checking on its arguments, and raises a warning where it is mis-applied. At present, the team have avoided making this an error as it is likely to be seen by end users rather than directly by package developers. In time, we are likely to revisit this and tighten up further on this key requirement.

Accessing random data

To support randomised data selection, we have introduced a family of experimental functions which use underlying engine support for random values, and provide one entry at random from the data type.

At the same time, we have addressed some issues with uniformity stemming from the random number function used by pdfTeX and inherited by other engines. This means that expl3's FPU will generate *pseudo*-random values across the range of possible outputs.

More powerful debugging

A new set of debugging functions have been added to the kernel. These allow debug code to be enabled locally using the new option `enable-debug` along with functions `\debug_on:n` and `\debug_off:n`. Accompanying this change, we have improved the handling of global/local consistency in variable setting.

Mark-up changes in l3doc

Since the introduction of the `__` syntax to mark internal functions, the need for explicit markup of internal material in sources has been negated. As such, we have now dropped the requirement to mark internal material with `[aux]` when using l3doc. Instead, the status of functions and variables is auto-detected from the presence of `__`. For cases where non-standard names are used for internal code, the mark-up `[int]` is retained, *e.g.*

```
\begin{macro}[int]{\l@expl@enable@debug@bool}
```

l3build updates

Work on l3build has continued in parallel with expl3 work, in particular continuing to develop features to allow wider use of the tool.

Paralleling the move of the LaTeX3 codebase to Git, l3build now has its own separate Git repository: https://github.com/latex3/l3build. This will enable us to involve other developers in the Lua code required for the build system. At the same time, we have split the code into a number of small source files, again to ease development both for the team ourselves and for potential collaborators.

Another major change is that l3build can now retain the structure of source repositories when creating a CTAN archive. Whilst the team favor 'flat' source setups, other users prefer structured approaches. Most notably, this new l3build functionality means that it is now used to carry out beamer releases.

The other major new feature is a new approach to multiple test setups, which replaces the older `--testfiledir` option. In the new approach, separate configuration files are listed in the main `build.lua` script, and can be selected manually using a new `--config` switch. This new approach allows complex test setups to be run in a totally automated fashion, which is important for kernel testing.

Some changes to the normalisation routines have been carried out, some to deal with upcoming LuaTeX changes, others to address aspects which show up only in some tests. This has required `.tlg` updates in some cases: as far as possible, we strive to avoid requiring changes to the reference files.

References

[1] *SubGit*, TMate Software, https://subgit.com

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

LaTeX3 News

Issue 12, January 2020

Contents

Introduction 25

New features in expl3 25

A new argument specifier: e-type 25

New functions 25

String conversion moves to expl3 26

Case changing of text 26

Notable fixes and changes 26

File name parsing 26

Message formatting 26

Key inheritance 26

Floating point juxtaposition 26

Changing box dimensions 26

More functions moved to stable 26

Deprecations 26

Internal improvements 26

Cross-module functions 26

The backend 27

Better support for (u)pTeX 27

Options 27

Engine requirements 27

Documentation 27

News 27

ChangeLog 27

Changes in xparse 27

New experimental modules 27

l3build changes 27

Introduction

There has been quite a gap since the last *LaTeX3 News* (Issue 11, February 2018), and so there is quite a bit to cover here. Luckily, one of the things there *is* to cover is that we are using a more formalised approach for logging changes, so writing up what has happened is a bit easier. (By mistake LaTeX3 News 11 itself did not get *published* when written, but is now available: we have kept the information it contains separate as it is a good summary of the work that had happened in 2017.)

Work has continued apace across the LaTeX3 codebase in the last (nearly) two years. A lot of this is ultimately focussed on making the core of expl3 even more stable: *squeezing* out more experimental ideas, refining ones we have and making it a serious option for core LaTeX programming.

As a result of these activities, the LaTeX3 programming layer will be available as part of the kernel of LaTeX 2_ε from 2020-02-02 onwards, i.e., can be used without explicitly loading expl3. See *LaTeX News 31* [2] for more details on this.

New features in expl3

A new argument specifier: e-type

During 2018, the team worked with the TeX Live, XeTeX and (u)pTeX developers to add the `\expanded` primitive to pdfTeX, XeTeX and (u)pTeX. This primitive was originally suggested for pdfTeX v1.50 (never released), and was present in LuaTeX from the start of that project.

Adding `\expanded` lets us create a new argument specifier: `e`-type expansion. This is *almost* the same as `x`-type, but is itself expandable. (It also doesn't need doubled `#` tokens.) That's incredibly useful for creating function-like macros: you can ensure that *everything* is expanded in an argument before you go near it, with not an `\expandafter` in sight.

New functions

New programming tools have appeared in various places across expl3. The highlights are

- Shuffling of sequences to allow randomization
- Arrays of integers and floating point values; these have constant-time access
- Functions to return values after system shell usage
- Expandable access to file information, including file size, MD5 hash and modification date

For the latter, we have revised handling of file names considerably. There is now support for finding files in expansion contexts (by using the `\(pdf)filesize` primitive). Spaces and quotes in file names are now fully normalised, in a similar manner to the approach used by the latest LaTeX 2_ε kernel.

String conversion moves to expl3

In addition to entirely new functions, the team have moved the l3str-convert module from the l3experimental bundle into the expl3 core. This module is essential for dealing with the need to produce UTF-16 and UTF-32 strings in some contexts, and also offers built-in escape for url and PDF strings.

Case changing of text

Within expl3, the team have renamed and reworked the ideas from `\tl_upper_case:n` and so on, creating a new module l3text. This is a "final" home for functions to manipulate *text*; token lists that can reasonably be expected to expand to plain text plus limited markup, for example emphasis and labels/references. Moving these functions, we have also made a small number of changes in other modules to give consistent names to functions: see the change log for full details.

Over time, we anticipate that functions for other textual manipulation will be added to this module.

Notable fixes and changes

File name parsing

The functions for parsing file names have been entirely rewritten, partly as this is required for the expandable access to file information mentioned above. The new code correctly deals with spaces and quote marks in file names and splits the path/name/extension.

Message formatting

The format of messages in expl3 was originally quite text-heavy, the idea being that they would stand out in the `.log` file. However, this made them hard to find by a regular expression search, and was very different from the LaTeX 2_ε message approach. The formatting of expl3 messages has been aligned with that from the LaTeX 2_ε kernel, such that IDE scripts and similar will be able to find and extract them directly.

Key inheritance

A number of changes have been made to the inheritance code for keys, to allow inheritance to work "as expected" in (almost) all cases.

Floating point juxtaposition

Implicit multiplication by juxtaposition, such as `2pi`, is now handled separately from parenthetic values. Thus for example `1in/1cm` is treated as equal to `(1in)/(1cm)` and thus yields `2.54`, and `1/2(pi+pi)` is equal to `pi`.

Changing box dimensions

TeX's handling of boxes is subtly different from other registers, and this shows up in particular when you want to resize a box. To bring treatment of boxes, or rather the grouping behavior of boxes, into line with other registers, we have made some internal changes to how functions such as `\box_set_wd:N` are implemented. This will be transparent for "well-behaved" use cases of these functions.

More functions moved to stable

A large number of functions which were introduced as candidates have been evaluated and moved to stable status. The team hopes to move all functions in expl3 to stable status, or move them out of the core, over the coming months.

Deprecations

There have been two notable sets of deprecations over the past 18 months. First, we have rationalised all of the "raw" primitive names to the form `\tex_<name>:D`. This means that the older names, starting `\pdftex_...`, `\xetex_...`, etc., have been removed.

Secondly, the use of integer constants, which dates back to the earliest days of expl3, is today more likely to make the code harder to read than anything else. Speed improvements in engines mean that the tiny enhancements in reading such constants are no longer required. Thus for example `\c_two` is deprecated in favour of simply using `2`.

In parallel with this, a number of older `.sty` files have been removed. These older files provided legacy stubs for files which have now been integrated in the expl3 core. They have now had sufficient time to allow users to update their code.

Internal improvements

Cross-module functions

The team introduced the idea of internal module functions some time ago. Within the kernel, there are places where functions need to be used in multiple modules. To make the nature of the kernel interactions clearer, we have worked on several aspects

- Reducing as far as possible cross-module functions
- Making more generally-useful functions public, for example scan marks
- Creating an explicit cross-kernel naming convention for functions which are internal but are essential to use in multiple kernel modules

The backend

Creating graphics, working with color, setting up hyperlinks and so on require backend-specific code. Here, backends are for example `dvips`, `xdvipdfmx` and the direct PDF mode in pdfTeX and LuaTeX. These functions are needed across the LaTeX3 codebase and have to be updated separately from the expl3 core. To facilitate that, we have split those sources into a separate bundle, which can be updated as required.

At the same time, the code these files contain is very low-level and is best described as internal. We have re-structured how the entire set of functions are referred to such that they are now internal for the area they implement, for example image inclusion, box affine transformations, etc.

Better support for (u)pTeX

The developers behind (u)pTeX (Japanese TeX) have recently enhanced their English documentation (see https://github.com/texjporg/ptex-manual). Using this new information, we have been able to make internal adjustments to expl3 to better support these engines.

Options

A new option `undo-recent-deprecations` is now available for cases where a document (or package) requires some expl3 functions that have been formally removed after deprecation. This is to allow *temporary* work-arounds for documents to be compiled whilst code is begin updated.

The "classical" options for selecting backends (`dvips`, `pdftex`, etc.) are now recognised in addition to the native key–value versions. This should make it much easier to use the expl3 image and color support as it is brought up to fully-workable standards.

Engine requirements

The minimum engine versions needed to use expl3 have been incremented a little:

- pdfTeX v1.40
- XeTeX v0.99992
- LuaTeX v0.95
- ε-(u)pTeX mid-2012

The team have also worked with the XeTeX and (u)pTeX developers to standardise the set of post-ε-TeX utility primitives that are available: the so-called "pdfTeX utilities". These are now available in all supported engines, and in time will all be *required*. This primarily impacts XeTeX, which gained most of these primitives in the 2019 TeX Live cycle. (Examples are the random number primitives and expandable file data provision.) See *LaTeX News 31* [2] for more.

Documentation

News

The *LaTeX3 News* files were until recently only used to create PDF files on the team website [1]. We have now integrated those into the l3kernel (expl3 core) bundle. The news files cover all of LaTeX3 files, as the core files are always available.

ChangeLog

Since the start of 2018, the team have commenced a comprehensive change log for each of the bundles which make up the LaTeX3 code. These are simple Markdown text files, which means that they can be displayed formatted in web views.

Changes in xparse

A number of new features have been added to xparse. To allow handling of the fact that skipping spaces may be required only in some cases when searching for optional arguments, a new modifier `!` is available in argument specifiers. This causes xparse to *require* that an optional argument follows immediately with no intervening spaces.

There is a new argument type purely for environments: `b`-type for collecting a `\begin...\end` pair, i.e., collecting the body of an environment. This is similar in concept to the environ package, but is integrated directly into xparse.

Finally, it is now possible to refer to one argument as the default for another optional one, for example

```
\NewDocumentCommand{\caption}{O{#2} m} ...
```

New experimental modules

A number of new experimental modules have been added within the l3experimental bundle:

l3benchmark Performance-testing system using the timing function in modern TeX engines

l3cctab Category code tables for all engines, not just LuaTeX

l3color Color support, similar in interface to xcolor

l3draw Creation of drawings, inspired by pgf, but using the LaTeX3 FPU for calculations

l3pdf Support for PDF features such as compression, hyperlinks, etc.

l3sys-shell Shell escape functions for file manipulation

l3build changes

The l3build tool for testing and releasing TeX packages has seen a number of incremental improvements. It is now available directly as a script in TeX Live and MiKTeX, meaning you can call it simply as

`l3build` *target*

Accompanying this, we have added support for installing scripts and script `man` files.

There is a new `upload` target that can take a zip file and send it to CTAN: you just have to fill in release information for *this* upload at the prompts.

Testing using PDF files rather than logs has been heavily revised: this is vital for work on PDF tagging.

There is also better support for complex directory structures, including the ability to manually specify TDS location for all installed files. This is particularly targeted at packages with both generic and format-specific files to install.

References

[1] *LaTeX Project Website.* https://latex-project.org/

[2] *LaTeX 2ε release newsletters on the LaTeX Project Website.* https://latex-project.org/news/latex2e-news/