LaTeX News

Issue 28, April 2018 (2018-04-01)

Contents

A new home for LaTeX 2ε sources

In the past the development version of the LaTeX 2ε source files has been managed in a Subversion source control system with read access for the public. This way it was possible to download in an emergency the latest version even before it was released to CTAN and made its way into the various distributions.

We have recently changed this setup and now manage the sources using Git and placed the master sources on GitHub at

`https://github.com/latex3/latex2e`

where we already store the sources for expl3 and other work. As before, direct write access is restricted to LaTeX Project Team members, but everything is publicly accessible including the ability to download, clone (using Git) or checkout (using SVN). More details are given in [1].

Bug reports for core LaTeX 2ε

For more than two decades we used GNATS, an open source bug tracking system developed by the FSF. While that has served us well in the past it started to show its age more and more. So as part of this move we also decided to finally retire the old LaTeX bug database and replace it with the standard "Issue Tracker" available at Github.

The requirements and the workflow for reporting a bug in the core LaTeX software is documented at

`https://www.latex-project.org/bugs/`

and with further details also discussed in [1].

UTF-8: the new default input encoding

The first TeX implementations only supported reading 7-bit ASCII files—any accented or otherwise "special" character had to be entered using commands, if it could be represented at all. For example to obtain an "ä" one would enter `\"a`, and to typeset a "ß" the command `\ss`. Furthermore fonts at that time had 128 glyphs inside, holding the ASCII characters, some accents to build composite glyphs from a letter and an accent, and a few special symbols such as parentheses, etc.

With 8-bit TeX engines such as pdfTeX this situation changed somewhat: it was now possible to process 8-bit files, i.e., files that could encode 256 different characters. However, 256 is still a fairly small number and with this limitation it is only possible to encode a few languages and for other languages one would need to change the encoding (i.e., interpret the character positions 0–255 in a different way). The first code points 0–127 were essentially normed (corresponding to ASCII) while the second half 128–255 would vary by holding different accented characters to support a certain set of languages.

Each computer used one of these encodings when storing or interpreting files and as long as two computers used the same encoding it was (easily) possible to exchange files between them and have them interpreted and processed correctly.

But different computers may have used different encodings and given that a computer file is simply a sequence of bytes with no indication for which encoding is intended, chaos could easily happen and has happened. For example, the German word "Größe" (height) entered on a German keyboard could show up as "GrŤàe" on a different computer using a different encoding by default.

So in summmary the situation wasn't at all good and it was clear in the early nineties that LaTeX 2ε (that was being developed to provide a LaTeX version usable across the world) had to provide a solution to this issue.

The LaTeX 2ε answer was the introduction of the inputenc package [2] through which it is possible to

provide support for multiple encodings. It also allows to correctly process a file written in one encoding on a computer using a different encoding and even supports documents where the encoding changes midway.

Since the first release of LaTeX 2$_\varepsilon$ in 1994, LaTeX documents that used any characters outside ASCII in the source (i.e. any characters in the range of 128–255) were supposed to load inputenc and specify in which file encoding they were written and stored. If the inputenc package was not loaded then LaTeX used a "raw" encoding which essentially took each byte from the input file and typeset the glyph that happened to be in that position in the current font—something that sometimes produces the right result but often enough will not.

In 1992 Ken Thompson and Rob Pike developed the UTF-8 encoding scheme which enables the encoding of all Unicode characters within 8-bit sequences. Over time this encoding has gradually taken over the world, replacing the legacy 8-bit encodings used before. These days all major computer operating systems use UTF-8 to store their files and it requires some effort to explicitly store files in one of the legacy encodings.

As a result, whenever LaTeX users want to use any accented characters from their keyboard (instead of resorting to `\"a` and the like) they always have to use

```
\usepackage[utf8]{inputenc}
```

in the preamble of their documents as otherwise LaTeX will produce gibberish.

The new default

With this release, the default encoding for LaTeX files has been changed from the "fall through raw" encoding to UTF-8 if used with classic TeX or pdfTeX. The implementation is essentially the same as the existing UTF-8 support from `\usepackage[utf8]{inputenc}`.

The LuaTeX and XeTeX engines always supported the UTF-8 encoding as their native (and only) input encoding, so with these engines inputenc was always a no-op.

This means that with new documents one can assume UTF-8 input and it is no longer required to always specify `\usepackage[utf8]{inputenc}`. But if this line is present it will not hurt either.

Compatibility

For most existing documents this change will be transparent:

- documents using only ASCII in the input file and accessing accented characters via commands;
- documents that specified the encoding of their file via an option to the inputenc package and then used 8-bit characters in that encoding;
- documents that already had been stored in UTF-8 (whether or not specifying this via inputenc).

Only documents that have been stored in a legacy encoding and used accented letters from the keyboard *without* loading inputenc (relying on the similarities between the input used and the T1 font encoding) are affected.

These documents will now generate an error that they contain invalid UTF-8 sequences. However, such documents may be easily processed by adding the new command `\UseRawInputEncoding` as the first line of the file. This will re-instate the previous "raw" encoding default.

`\UseRawInputEncoding` may also be used on the command line to process existing files without requiring the file to be edited

```
pdflatex '\UseRawInputEncoding \input'  file
```

will process the file using the previous default encoding.

Possible alternatives are reencoding the file to UTF-8 using a tool (such as recode or iconv or an editor) or adding the line

```
\usepackage[⟨encoding⟩]{inputenc}
```

to the preamble specifying the ⟨*encoding*⟩ that fits the file encoding. In many cases this will be `latin1` or `cp1252`. For other encoding names and their meaning see the inputenc documentation.

As usual, this change may also be reverted via the more general latexrelease package mechanism, by speciying a release date earlier than this release.

BOM: byte order mark handling

When using Unicode the first bytes of a file may be a, so called, BOM character (byte order mark) to indicate the byte order used in the file. While this is not required with UTF-8 encoded files (where the byte order is known) it is nevertheless allowed by the standard and some editors add that byte sequence to the beginning of a file. In the past such files would have generated a "Missing begin document" error or displayed strange characters when loaded at a later stage.

With the addition of UTF-8 support to the kernel it is now possible to identify and ignore such BOMs characters even before `\documentclass` so that these issues will no longer be showing up.

A general rollback concept for packages and classes

In 2015 a rollback concept for the LaTeX kernel was introduced. Providing this feature allowed us to make corrections to the software (which more or less didn't happen for nearly two decades) while continuing to maintain backward compatibility to the highest degree.

In this release we have now extended this concept to the world of packages and classes which was not covered initially. As the classes and the extension packages have different requirements compared to the kernel, the approach is different (and simplified). This should make it easy for package developers to apply it to their packages and authors to use when necessary.

The documentation of this new feature is given in an article submitted to TUGboat and also available from our website [3].

Integration of remreset and chngcntr packages into the kernel

With the optional argument to `\newcounter` LaTeX offers to automatically reset counters when some counter is stepped, e.g., stepping a `chapter` counter resets the `section` counter (and recursively all other heading counters). However, what was until now missing was a way to undo such a link between counters or to link two counters after they have been defined.

This can be now be done with `\counterwithin` and `\counterwithout`, respectively. In the past one had to load the chngcntr package for this. For the programming level we also added `\@removefromreset` as the counterpart of the already existing `\@addtoreset` command. Up to now this was offered by the remreset package.

Testing for undefined commands

LaTeX packages often use a test `\@ifundefined` to test if a command is defined. Unfortunately this had the side effect of *defining* the command to `\relax` in the case that it had no definition. The new release uses a modified definition (using extra testing possibilities available in ε-TeX). The new definition is more natural, however code that was relying on the side effect of the command being tested being defined if it was previously undefined may have to add `\let\`⟨*command*⟩`\relax`.

Changes to packages in the tools category

LaTeX table columns with fixed widths

Frank published a short paper in TUGboat [4] on producing tables that have columns with fixed widths. The outlined approach using column specifiers "`w`" and "`W`" has now been integrated into the array package.

Obscure overprinting with multicol fixed

A rather peculiar bug was reported on StackExchange for multicol. If the column/page breaking was fully controlled by the user (through `\columnbreak`) instead of letting the environment do its job and if then more `\columnbreak` commands showed up on the last page then the balancing algorithm was thrown off track. As a result some parts of the columns did overprint each other.

The fix required a redesign of the output routines used by multicol and while it "should" be transparent in other cases (and all tests in the regession test suite came out fine) there is the off-chance that code that hooked into internals of multicol needs adjustment.

Changes to packages in the amsmath category

With this release of LaTeX a few minor issues with amsmath have been corrected.

Updated user's guide

Furthermore, `amsldoc.pdf`, the AMS user's guide for the amsmath package [5], has been updated from version 2.0 to 2.1 to incorporate changes and corrections made between 2016 and 2018.

References

[1] Frank Mittelbach: *New rules for reporting bugs in the LaTeX core software*. In: TUGboat, 39#1, 2018. `https://www.latex-project.org/publications/`

[2] Frank Mittelbach: *LaTeX 2ε Encoding Interface — Purpose, concepts, and Open Problems*. Talk given in Brno June 1995. `https://www.latex-project.org/publications/`

[3] Frank Mittelbach: *A rollback concept for packages and classes*. Submitted to TUGboat. `https://www.latex-project.org/publications/`

[4] Frank Mittelbach: *LaTeX table columns with fixed widths*. In: TUGboat, 38#2, 2017. `https://www.latex-project.org/publications/`

[5] American Mathematical Society and The LaTeX Project: *User's Guide for the amsmath package* (Version 2.1). April 2018. Available from `https://www.ctan.org` and distributed as part of every LaTeX distribution.