LaTeX News

Issue 39, June 2024 — DRAFT version for upcoming release (LaTeX release 2024-06-01)

Contents

Introduction

to write

News from the "LaTeX Tagged PDF" project

In the previous LaTeX News [6] we announced a first prototype support for tagged tabulars. Some of the necessary support code has now been moved from `latex-lab` to the corresponding packages (using sockets and plugs) and to the LaTeX kernel (for those parts that are also necessary for other aspects of tagging).

The kernel code specific to tagging is implemented in the file `lttagging.dtx`. For now it contains `\UseTaggingSocket`, a special invocation command for sockets that are specific to tagging. This enables us to also provide `\SuspendTagging` and `\ResumeTagging`, i.e., a very efficient way to temporarily disable the whole tagging process. This is, for example, necessary, if some code is doing trial typesetting. In that case the trials should not generate tagging structures—only the finally chosen version should. Thus, tabularx, for example, stops the tagging while doing its trials to figure out the correct column widths to use, and then renables tagging when the table is finally typeset.

Over time, `lttagging.dtx` will hold more general tagging code as appropriate. For now it is only documented as part of `source2e.pdf` but long term we will provide a separate guide for tagging, which will then also include the information currently found in various other places, e.g., `tagpdf.pdf`.

We also added support for a few missing commands described in Leslie Lamport's *LaTeX Manual* [**?**]: If `phase-III` is used the `\marginpar` command will be properly tagged (depending on the PDF version) as an `Aside` or a `Note` structure. In the standard classes `\maketitle` will be tagged if the additional testphase module `title` is used.

The `math` module has been extended and now includes options to attach MathML files to the structures. First tests with a PDF reader and screen reader that support associated files looks very promising. Examples of PDF files tagged with the new method can be found at `https://github.com/latex3/tagging-project/discussions/56`.

At last various small bugs and problems reported at `https://github.com/latex3/tagging-project` has been fixed. Such a feedback is very valuable, so we hope to see you there and thank you for any contribution, whether it is an issue or a post on a discussion thread.

Enhancements to the new mark mechanism

In June 2022 we introduced a new mark mechanism in LaTeX [1, p. 76] that allows keeping track of multiple independent marks. It also properly supports top marks, something that wasn't reliably possible with LaTeX before.

There was, however, one limitation: to retrieve the marks from the page data it was necessary to `\vsplit` that data artificially so that TeX would produce split marks that the mechanism could then use. Unfortunately, TeX gets very upset if it finds infinite negative glue (e.g., from `\vss`) within this data. This is not totally surprising because such glue would allow splitting off any amount of material as such glue would hide the size of it. TeX therefore responds with an error message if it find such glue while doing a `\vsplit` operation (and it does so even if a later glue item cancels the infinite glue).

To account for this, the code in 2022 attempted to detect this situation beforehand and if so did not do any splitting but, of course, it would then also not extract any mark information.

In this release the approach has been changed and we always do a `\vsplit` operation and thus always get the right mark data extracted. While it is not possible to avoid upsetting TeX in case we have infinite negative glue present, it is possible to hide this (more or less) from the user.[1] With the new code TeX will neither stop nor show anything on the terminal. What we can't do, though, is to avoid that an error is written to the log file, but to make it clear that this error is harmless and should be ignored we have arranged the code so that the error message, if it is issued, takes the following format:

```
! Infinite glue shrinkage found in box being split.
<argument> Infinite shrink error above ignored !
l. ...  }
```

Not perfect (especially the somewhat unmotivated `<argument>`), but you can only do so much if error messages and their texts are hard-wired in the engine.

So why all this? There are two reasons: we do not lose marks in edge cases any longer and perhaps more importantly we are now also reliably able to extract marks from arbitrarily boxed data, something that wasn't possible at all before. This is, for example, necessary to support extended marks in `multicols` environments or extract them from floats, marginpars, etc.

Details about the implementation can be found in `texdoc ltmarks-code` or in the shorter `texdoc ltmarks-doc` (which only describes the general concepts and the command interfaces).

[1]A note to l3build users that make use of its testing capabilities: the new mechanism temporarily changes `\interactionmode` and, for implementation reasons in TeX, that results in extra newlines in the `.log` file, so instead of seeing `[1] [2]` you will see each on separate lines. This means that test files might show difference of that nature, once the code is active, and must therefore be regenerated as necessary.

New or improved commands

doc: provide \ProvideDocElement

Beside `\NewDocElement` and `\RenewDocElement` we now also offer a `\ProvideDocElement` declaration that does nothing unless the doc element could be declared with `\NewDocElement`. This can be useful if documentation files are processed both individually as well as combined.

doc: better support for upquote

In LaTeX News 37 [5] we wrote that support for the upquote package was added to the doc package, but back then this was only done for `\verb`, and the `verbatim` environments. However, the bulk of code in a typical `.dtx` file is within `macrocode` or `macrocode*` environments which were not affected by adding upquote. We have now updated those, such that upquote alters the quote characters in these environments as well.
(github issue 1230)

ifthen: guard against active characters in comparisons

The `\ifthenelse` command now ensures that `<`, `=` and `>` are safe in numeric tests, even if they have been made active (typically by babel language shorthands.)
(github issue 756)

Providing xtemplate in the format

In LaTeX News 32, we described the move of one long-term experimental idea into the kernel: the package xparse, which was integrated as ltcmd. With this edition, we move another long-term development idea to stable status: *templates*.

In this context, templates are a mechanism to abstract out various elements of a document (such as "sectioning") in such a way that different implementations can be interchanged, and design decisions are set up efficiently and controllably.

In contrast to ltcmd, which provides a mechanism that many document authors will exploit routinely, templates are a more specialised tool. We anticipate that they will be used by a small number of programmers, providing generic ideas that will then be used within document classes. Most document authors will therefore likely encounter templates directly only rarely. We anticipate though that they will be *using* templates provided by the team or others.

The template system requires three separate ideas

- Template *type*: the "thing" we are using templates for, such as "sectioning" or "enumerated-list"
- A template: a combination of code and keys that can be used to implement a type. Here for example we might have "standard-LaTeX-sectioning" as a template for "sectioning"
- One or more *instances*: a specific use case of a template where (some) keys are set to known values. We might for example see "LaTeX-section", "LaTeX-subsection", etc.

As part of the move from the experimental `xtemplate` to kernel integration, the team have revisited the commands provided. The stable set now comprises

- `\NewTemplateType`
- `\DeclareTemplateInterface`
- `\DeclareTemplateCode`
- `\DeclareTemplateCopy`
- `\EditTemplateDefault`
- `\UseTemplate`
- `\DeclareInstance`
- `\DeclareInstanceCopy`
- `\EditInstance`
- `\UseInstance`

To support existing package authors, we have released an updated version of `xtemplate` which will work smoothly with the new kernel-level code. The existing commands provided in `xtemplate` will continue to work, but we encourage programmers to move to the set above.

Code improvements

Loading packages at the top level

Classes and packages should only be loaded with `\documentclass`, `\usepackage`, or class interface commands such as `\LoadClass` or `\RequirePackageWithOptions` at the top level, not inside a group. Previously LaTeX did not check this, which would often lead to low level errors later on if package declarations were reverted as a group ended. LaTeX now checks the group level and an error is thrown if the class or package is loaded in a group. *(github issue 1185)*

Keep track of lost glyphs

A while ago we changed the LaTeX default value for `\tracinglostchars` from `1` to `2` so that missing glyphs generate at least a warning, but we forgot to make the same change to `\tracingnone`. Thus, when issuing that command LaTeX stopped generating warnings about missing glyphs. This has now been corrected. *(github issue 549)*

Improve fontenc error message

If the `fontenc` is asked to load a font encoding for which it doesn't find a suitable `.def` file it generates an error message indicating that the encoding name might be misspelled. That is, of course, one of the possible causes, but another one is that the installation is missing a necessary support package, e.g., that no support for Cyrillic fonts has been installed. The error message text has therefore been extended to explain the issue more generally. *(github issue 1102)*

Warn if counter names are problematic

In the past it was possible to declare, for example, `\newcounter{index}` with the side-effect that this defines `\theindex`, even though LaTeX has a `theindex` environment that then got clobbered by the declaration. This has now been changed: if `\the`⟨*counter*⟩ is already defined it is not altered, but instead a warning message is displayed. *(github issue 823)*

Extended information in `\listfiles`

The `\listfiles` command provides useful information when finding issues related to variation in package versions. However, this has to date relied on the information in the `\ProvidesPackage` line, or similar: that can be misleading if for example a file has been edited locally. We have now extended `\listfiles` to take an optional argument which will then include the MD5 hash of each file (and the size of each file) in the `.log`. Thus for example you can use

```
\listfiles[hashes,sizes]
```

to get both the file sizes and file hashes in the `.log` as well as the standard release information. *(github issue 945)*

Optimize creation of simple document commands

Creating document commands using `\NewDocumentCommand`, etc., provides a very flexible way of grabbing arguments. When the document command only takes simple mandatory arguments, this has to-date added an overhead that could be avoided. We have now refined the internal code path such that "simple" document commands avoid almost any overhead at point-of-use, making the results essentially as efficient as using `\newcommand` for low-level TeX constructs. Note that as `\NewDocumentCommand` makes engine-robust commands, the direct equivalent to `\newcommand` is `\NewExpandableDocumentCommand`. *(github issue 1189)*

Handling of end-of-lines in `\NewDocumentCommand +v` arguments

The `+v` argument type provided by `\NewDocumentCommand`, etc., allows grabbing of multiple lines of text in a verbatim-like argument. Almost always, the result of this grabbing will be used in a typesetting context. Previously, the end-of-line characters were stored literally as category code 12 (“other”) `^^M` tokens. However, these are difficult to work with in general. We have now revised this behavior, such that end-of-line characters are converted to the `\obeyedline` command when parsed by `+v`-type arguments. This may require adjustment in the source of some documents, but the enhanced ability of users and programmers to exploit the `+v`-type argument means we believe it is necessary.

Declaring appropriate sub-encodings for TS1 symbol fonts

In 2020 we incorporated support for the `TS1` symbol encoding directly into the kernel and in this way removed the need to load the `textcomp` package [2] to make commands such as `\texteuro` available.

There is, however, a big problem with this `TS1` symbol encoding: only very few fonts can provide every glyph that is supposed to be part of `TS1`. This means that changing font families might result in certain symbols becoming unavailable. This can be a major disaster if, for example, the `\texteuro` (€) or the `\textohm` (Ω) are no longer printed in your document, just because you altered your text font family.

To mitigate this problem, we also introduced in 2020 the declaration `\DeclareEncodingSubset`. This declaration is supposed to be used in font definition files for the `TS1` encoding to specify which subset (we have defined 10 common ones) a specific font implements. If such a declaration is used then missing symbols are automatically taking from a fallback font. While this is not perfect, it is the best you can do other than painstrickenly checking that your document only uses glyphs that the font supports and if necessary switch to a different font or avoid the missing symbols. See also the discussion in [3].

To jumpstart the process we also added declarations to the LaTeX kernel for most of the fonts found in TeXLive at the time—with the assumption that such declarations would over time be superseded by declarations in the `.fd` files. Unfortunately, this hasn’t happened yet (or not often) and so many of the initial declarations went stale: several fonts got new glyphs added to them (so their sub-encoding should have been changed but didn’t); others (mainly due to license issues) changed the family name and thus our declarations became useless and the renamed fonts (now without a declaration) ended up in the default sub-encoding which offers only few glyphs; yet others such as CharisSIL (which triggered the GitHub issue) were simply not around at the time.

We have therefore, again attempted to provide the (currently) correct declarations, but it is obvious that this is not a workable process. As we do not maintain the fonts we do not have the information that something has changed, and to regularly check the ever growing font support bundles is simply not possible. It is therefore very important that maintainers of font packages do not only provide `.fd` files but also add such a declaration to every `TS1...fd` font definition file that they distribute.

To simplify this process, we now provide a simple LaTeX file (`checkencodingsubset.tex`) for determining the correct (safe) sub-encoding. If run, it asks for a font family and then outputs its findings, for example, for `AlgolRevived-TLF` you will get:

```
-----------------------------------------
Testing font family AlgolRevived-TLF
(currently TS1-sub-encoding 9)
-----------------------------------------
Some glyphs are missing from sub-encoding 8:
   ==> \textcelsius (137) is missing
   ==> \texttwosuperior (178) is missing
   ==> \textthreesuperior (179) is missing
   ==> \textonesuperior (185) is missing
Some glyphs are missing from sub-encoding 7:
   ==> \texteuro (191) is missing
All glyphs between sub-encoding 6 and 7 exist
All glyphs between sub-encoding 5 and 6 exist
All glyphs between sub-encoding 4 and 5 exist
Some glyphs are missing from sub-encoding 3:
   ==> \textwon (142) is missing
All glyphs between sub-encoding 2 and 3 exist
Some glyphs are missing from sub-encoding 1:
   ==> \textmho (77) is missing
   ==> \textpertenthousand (152) is missing
All glyphs between sub-encoding 0 and 1 exist
All glyphs in core exist
-----------------------------------------
TS1 encoding subset for AlgolRevived-TLF (ok)
Use sub-encoding 9
-----------------------------------------
```

This output is meant for human consumption, e.g., you see which glyphs are missing and why a certain sub-encoding is suggested, but it is not that hard to use it in a script and extract the suggested sub-encoding by grepping for the line starting with `Use sub-encoding`.

Of course, this check will only work if the missing glyphs are really missing: some fonts placed “tofu”[2] into such slots and in this case it looks to TeX as if the glyph is provided. For example, for the old Palatino fonts (family `ppl`) it would report

```
-----------------------------------------
TS1 encoding subset for ppl (bad)
Use sub-encoding 0 (not 5)
-----------------------------------------
```

i.e., all glyphs are provided, while in reality more than twenty are missing and sub-encoding 5 as declared in the kernel is correct. *(github issue 1257)*

[2]Little squares to indicate a missing symbol.

Documentation improvements

Further updates to the guides

We reported about the updated versions of `usrguide` and `clsguide` in LaTeX News 37 [5]. We have now revised `fntguide` as well to reflect the changes and macros added to the kernel over the last years of development. Note that the file name hasn't changed and there is no `fntguide-historic`.

Bug fixes

Fix inconsistent expansion on package option list

LaTeX applies one-step expansion to raw option list of packages and classes so constructions like

```
\def\myoptions{opt1,opt2}
\usepackage[\myoptions]{foo}
```

are supported. But when a package declares its options with the new key/value approach [4] and was loaded a second time, its raw option list was not expanded and an error might be raised. This has now been corrected. *(github issue 1298)*

Changes to packages in the amsmath category

amsmath: correct equation tag placement

If there is not enough space to place an equation tag on the same line as the equation amsmath calculates a suitable offset and then places the tag above (or below) the equation. In the case of the `gather` environment this offset was not reset at the end, with the result that it applied to a following environment as well resulting in incorrect spacing in certain situations. This has now been corrected. *(github issue 1289)*

Changes to packages in the tools category

array, longtable, tabularx: support tagging

The three packages are now extended to enable producing tagged tabulars upon request. This is done by adding a number of sockets (see [6]) that, by default, do nothing, but are equipped with appropriate plugs if tagged PDF is requested.

In the previous LaTeX release this was handled in `latex-lab` patching the packages when tagging was requested.

verbatim: `\verb` showed visible spaces

A recent change in the kernel was not reflected in the verbatim package with the result that `\verb` showed visible spaces (␣) after the package was loaded. This has already been corrected in a hotfix for release 2023-11. *(github issue 1160)*

verbatim: Support tabs in `\verbatiminput*`

Mimicking the kernel update (November, 2023) that allowed `\verb*` to mark tabs as spaces, the verbatim package has been updated such that `\verbatiminput*` marks tabs as spaces as well. *(github issue 1245)*

multicol: `\columnbreak` interferes with mark mechanism

The multicol package has to keep track of marks (from `\markright` or `\markboth`) as part of its output routine code and can't rely on LaTeX handling that automatically. It does so by artificially splitting page data with `\vsplit` to extract the mark data. With the introduction of `\columnbreak` that code failed sometimes, because it was not seeing any mark that followed such a forced column break.

This has now been corrected, but there is further work to do, because as of now multicol does not yet handle marks using the new mark mechanism—see the discussion at the beginning of the newsletter. *(github issue 1130)*

showkeys: Allow `\newline` in amsthm to work

Previously showkeys added an extra box layer which disabled the `\newline` of amsthm theorem styles. This extra box has now been avoided. *(github issue 1123)*

Changes to files in the cyrillic category

Correct definition of `\k`

Ages ago, the encoding specific definitions for various accent commands were changed to guard against altering some parameter values non-locally by mistake. For some reason the definition for `\k` in the Cyrillic encodings `T2A`, `T2B`, and `T2C` didn't get this treatment. This oversight has now been corrected. *(github issue 1148)*

References

[1] LaTeX Project Team. *LaTeX 2ε news 1–39*. June, 2024. `https://latex-project.org/news/latex2e-news/ltnews.pdf`

[2] LaTeX Project Team. *LaTeX 2ε news 31*. February, 2020. `https://latex-project.org/news/latex2e-news/ltnews31.pdf`

[3] LaTeX Project Team. *LaTeX 2ε news 33*. June 2021. `https://latex-project.org/news/latex2e-news/ltnews33.pdf`

[4] LaTeX Project Team. *LaTeX 2ε news 35*. June 2022. `https://latex-project.org/news/latex2e-news/ltnews35.pdf`

[5] LaTeX Project Team. *LaTeX 2ε news 37*. June 2023. `https://latex-project.org/news/latex2e-news/ltnews37.pdf`

[6] LaTeX Project Team. *LaTeX 2ε news 38*. November 2023.
`https://latex-project.org/news/latex2e-news/ltnews38.pdf`