LaTeX News

Issue 41, June 2025 — DRAFT version for upcoming release (LaTeX release 2025-06-01)

Contents

Introduction 1

Replacement for the legacy mark mechanism 1

Configurable output routine 1

News from Tagged PDF project 2
- Setting up math tagging . . . 2
- Promoting PDF 2.0 . . . 2
- Fixing the spacing after display math . . . 3

New or improved commands 3
- Socket and plug conditionals . . . 3
- Accessing the current counter . . . 3
- Collecting environment bodies verbatim . . . 3

Code improvements 3
- Refinement of `\MakeTitlecase` . . . 3
- Tab character as a special . . . 3
- Refinement of `v` specification category codes . . 4
- Logging text command and symbol declarations 4
- Improvement of the NFSS font series management . . . 4
- Supporting the `ssc` and `sw` shapes . . . 4
- Improving the handling of `\label`, `\index`, and `\glossary` . . . 4

Bug fixes 4
- Fix the use of `localmathalpabets` . . . 4
- docstrip: Error if `.ins` file is problematical . . . 4
- Prevent `cmd` hook from defining an undefined command . . . 4
- Process global options once per package . . . 5
- Make `\label`, `\index`, and `\glossary` truely invisible in running headers . . . 5
- Correct the float placement algorithm . . . 5
- Correct `\CheckEncodingSubset` . . . 5

Documentation 5
- Clarifying space handling of `\textcolor` . . . 5

Changes to packages in the graphics category 5
- More accessibility keys in graphicx . . . 5

Changes to packages in the tools category 5
- multicol: Full support for extended marks . . . 5
- array: Improve preamble setup code for `p` and friends . . . 5
- varioref: How to make `\reftex...` empty . . . 5

Introduction

to write

Replacement for the legacy mark mechanism

LaTeX's legacy mechanism only supported two classes (left and right marks) and setting the left mark (with `\markboth`) always altered the state of the right mark as well, i.e., they were far from independent. For generating running headers with "chapter titles" on the left and "section titles" on the right they work reasonably well but without much flexibility, e.g., `\leftmark` always generated the first "left"-mark on the page, while `\rightmark` always generated the last "right"-mark.

A few releases ago [5] we therefore introduced a new mark mechanism for LaTeX that supports arbitrary many truly independent mark classes and also offers querying the state at the top of the page, something that wasn't available in LaTeX at all.

Up to now, both mechanisms coexisted with completely separate implementations. With this release we have retired the legacy code and instead implement its public interfaces by using the new concepts, i.e., `\markboth`, `\markright`, `\leftmark`, and `\rightmark` remain supported but internally use `\InsertMark`, etc. Existing document classes or documents using the interfaces will therefore continue to work without any modifications but use a single underlying implementation and new documents can benefit from the additional flexibility, e.g., by displaying not only the last right-mark (`\leftmark` or `\LastMark{2e-right}`) but also the first right-mark (`\FirstMark{2e-right}`) or the top right-mark (`\TopMark{2e-right}`), etc.

See [9] for details on the extended functionality.

Configurable output routine

For nearly 40 years LaTeX's output routine (the mechanism to paginate the document and attach footnotes, floats and headers & footers) was a largely hardwired algorithm with a limited number of configuration possibilities. Packages that attempted to alter one or the other aspect of the process had to overwrite the internals

with the usual problems: incompatibilities and out of date code whenever something was changed in LaTeX.

To improve this situation and to support the production of accessible PDF documents we started to refactor the output routine and added a number of hooks and sockets, so that packages that want to adjust the output routine can do so safely without the dangers associated with that in the past.

For packages, we implemented the following hooks:

build/page/before, build/page/after These two hooks enable packages to prepend or append code to the page processing in the output routine. They are implemented as mirrored hooks.

Technically, they are executed at the start and the end of the internal LaTeX 2_ε `\@outputpage` command, respectively. A number of packages alter that command to place code in exactly these two places—they can now simply add their code to the hooks instead.

build/page/reset Packages that set up special conventions for text in the main galley (such as catcode changes, etc.) can use this hook to undo these changes within the output routine, so that they aren't applied to unrelated material, e.g., the text for running header or footers.

build/column/before, build/column/after These two hooks enable packages to prepend or append code to the column processing in the output routine. They are implemented as mirrored hooks.

Technically, they are executed at the start and the end of the internal LaTeX 2_ε `\@makecol` command, respectively. A number of packages alter `\@makecol` to place code in exactly these two places—they can now simply add their code to the hooks instead.

We also added a number of sockets for configuring the algorithm and to support tagging. One socket that is of interest for class files but also for user in the document preamble is `build/column/outputbox`. It defines how the column text, the column floats (top and bottom) and the footnotes are combined, i.e. their order and spacing. To change the layout all one has to do is to assign a different predefined plug to the socket with

`\AssignSocketPlug{build/column/outputbox}`
`{`⟨*plug-name*⟩`}`

The predeclared plugs are the following:

space-footnotes-floats After the galley text there is a vertical `\vfill` followed by the footnotes, followed by the bottom floats, if any.

footnotes-space-floats As before but the `\vfill` is between footnotes and floats.

floats-space-footnotes Floats are directly after the text, then a space and then footnotes at the bottom.

space-floats-footnotes Both floats and footnotes are pushed to the bottom with footnotes coming last.[1]

floats-footnotes All excess space is distributed across the existing glue on the page, e.g., within the text galley, the separation between blocks, etc. The order is text, floats, footnotes.

footnotes-floats As the previous one but floats and footnotes are swapped. This is the LaTeX default for newer documents, i.e., this plug is assigned to the socket when `\DocumentMetadata` is used.

footnotes-floats-legacy As the previous one but LaTeX's bottom skip bug is not corrected, i.e., in ragged bottom designs where footnotes are supposed to be directly attached to the text, they suddenly appear at the bottom of the page when the page is ended with `\newpage` or `\clearpage`. While this is clearly a bug, it was the case since the days of LaTeX 2.09; thus for compatibility we continue to support this behavior.

There are more configuration possibilities, mainly for class developers; more documentation on those can be found in [10, §54 ltoutput.dtx].

News from Tagged PDF project

The `testphase` key now takes also the value `latest`. This will load all modules that we recommend so that it is not necessary to specify individual modules. The list of loaded modules will be adjusted as needed when the project progresses. For reference, it is also written to the log.

to write

Setting up math tagging

With the LuaTeX engine there are now various options for accessible math described in full details in latex-lab-math.pdf. To simplify the setup a new key `math/setup` can be used that accepts a comma list with the values `mathml-SE` (add MathML structure element), `mathml-AF` (attach MathML associated file) or `tex-AF` (attach the TeX source).

Promoting PDF 2.0

PDF 2.0 is a requirement for accessible PDF containing math as the MathML name spaces isn't supported by earlier PDF versions. LaTeX will therefore set PDF 2.0 as default version if `\DocumentMetadata` is used. A different PDF version can be set with the `pdfversion` key.

[1] There are two more permutations, but neither of them has ever been requested so they aren't set up by default — doing that in a class would be trivial though.

Fixing the spacing after display math

When LaTeX produces an accessible (tagged) PDF it has to add structure data into the PDF to mark (i.e., tag) individual elements. If the pdfTeX engine is used this has to be done with the help of `\pdfliteral`s which are whatsit nodes like `\special` or `\write`. This means that they should be added only in places, where these extra nodes are not affecting the spacing—TeX can't, for example, look backwards past such a whatsit node so consecutive spaces that are normally collapsed into one, suddenly appear both, if such a node separates them.

The situation is especially complicated with math displays, because there TeX adds penalties and spaces with low-level procedures, that are not directly accessible from the macro level, and the tagging structures have to appear somewhere in the middle of that to ensure that the formula and the PDF structures are not separated by page break. Because of this it is necessary to use some fairly complex methods (essentially disable TeX's mechanisms and reprogram them on the macro level) to get the structure data in the right places.

Our first attempt in doing that was slightly faulty and resulted in some cases in an extra space (an additional `\parskip` space when there shouldn't be one). This has now corrected and the gymnastics to achieve this are rather an "interesting" study in obfuscated TeX coding.

In LuaTeX the situation is much better because there the structures can be added later when the formula processing has already finished. *(tagging-project issue 762)*

New or improved commands

Socket and plug conditionals

It is sometimes necessary/helpful to know if a particular socket or plug exists (or is assigned to a certain socket) and based on that take different actions. With the current release we added conditionals, such as `\IfSocketExistsTF`, to support such scenarios. Corresponding L3 programming layer conditionals are also provided. *(github issue 1577)*

Accessing the current counter

Counter commands such as `\alph`, `\stepcounter`, may now have the argument `*` to denote the current counter (as used by `\label`). This is compatible with the package enumitem use of `\alph*` in item labels but is generally available. Not all commands accept `*`, for example `\counterwithin` and `\counterwithout` require counter names as before. *(github issue 1632)*

Collecting environment bodies verbatim

The mechanisms in ltcmd ("xparse") offer a powerful way to specify a range of types of document command and environment syntax. This includes the ability to collect the entire body of an environment, for cases where treating it as a standard argument is useful. It is also possible in ltcmd to define arguments which grab their content verbatim, another specialist argument form. To date, however, it was not possible to combine these two ideas.

In this release, a new specifier `c` is introduced, which collects the body of an environment in a verbatim-like way. Like the existing `+v` specification, each separate line is marked by the special `\obeyedline` marker, which as standard issues a normal paragraph. Thus, this new specifier is usable both for typesetting and collecting file contents (the letter `c` indicates "collect code"). Thus, we may use

```
\NewDocumentEnvironment
  {MyVerbatim}{!O{\ttfamily} c}
  {\begin{center}#1 #2\end{center}} {}
\begin{MyVerbatim}[\ttfamily\itshape]
  % Some code is shown here
  $y = mx + c$
\end{MyVerbatim}
```

to obtain

```
% Some code is shown here
$y = mx + c$
```

Code improvements

Refinement of `\MakeTitlecase`

We introduced `\MakeTitlecase` as a late addition to the June 2022 release, making use of the improved case code in expl3. Unlike upper and lowercasing, making text titlecased is more tricky to get right: this can apply either to the whole text or on a word-by-word basis.

A subtle issue was reported against the expl3 repository (`https://github.com/latex3/latex3/issues/1316`) which links to how we deal with the question of case changing "words" but shows up if you titlecase text stored in a command.

We have looked again at how to implement `\MakeTitlecase` to be as predictable as possible, and have made a change in this release. The command no longer tries to lowercase text before applying titlecasing, and gives the correct result for text stored in commands.

We have also added an additional key to the optional argument to `\MakeTitlecase` which allows the user to decide if this will apply only to the first word (the default) or to all words.

Tab character as a special

In LaTeX News 38, we described the extension of `\verb`, etc., to cover the tab character as equivalent to a space. We have now added tabs to the standard list of characters covered by `\dospecials`. This allows them to be used in for example a `v` specification document command without additional steps.

Refinement of v specification category codes
Work on verbatim argument handling has highlighted that storing all characters as "other" (category code 12) when using a `v` specification in ltcmd was problematic. We have now revised this to capture letters with their original cateogry code.

Logging text command and symbol declarations
For thirty years the documentation claimed that `\DeclareTextSymbol`, `\DeclareTextCommand`, and friends log their changes, but in contrast to their math counterparts they never did. This has now finally changed. *(github issue 1242)*

Improvement of the NFSS font series management
LaTeX's font selection mechanism (NFSS) supports 9 weight levels, from ultra-light (`ul`) to ultra-bold (`ub`), and also 9 width levels, from ultra-condensed (`uc`) to ultra-expanded (`ux`). With the February 2020 release, this mechanism was extended so that requests to set the weight or the width attributes of the series are combined in a sensible way [3]: E.g., if you typeset a paragraph in a condensed face using `\fontseries{c}\selectfont` and then use `\textbf` inside the paragraph, a bold condensed face is selected. The combination of the series values is done by consulting a simple lookup table whose entries are defined with `\DeclareFontSeriesChangeRule`.

Until now, this lookup table was missing some entries, especially with regard to rarely used width values. In such cases, the series values were not combined as expected. This has been fixed (thanks to Maurice Hansen) by adding numerous `\DeclareFontSeriesChangeRule` entries so that the full range of weights (from `ul` to `ub`) and widths (from `uc` to `ux`) is now supported when combining font series values. *(github issue 1396)*

Supporting the ssc and sw shapes
The `ssc` shape (spaced small capitals) is supported in LaTeX through the commands `\sscshape` and `\textssc`. However, until this release there where no font shape change rules defined for this admittely seldom available shape, so that

```
\sscshape\itshape
```

changed unconditionally to `it` (italics) rather than to `sscit` (spaced small italic capitals). Thanks to Michael Ummels, the missing declarations have now been added so that shape changes in font families that support spaced small capitals work properly.

At the same time we took the opportunity to improve the fallbacks for the `sw` (swash) shapes, which are accessible through the commands `\swshape` or `\textsw`. If an `sw` combination is not available, the rules now try to replace `sw` with `it` rather than falling back to `n`. *(github issue 1581)*

Improving the handling of `\label`, `\index`, and `\glossary`
In standard LaTeX, the three commands `\label`, `\index`, and `\glossary` take exactly one mandatory argument, e.g., `\index{`⟨*entry*⟩`}`. In some extension packages, for example, in index or cleveref, they are augmented to accept an optional argument and in case of `\index` also a star form. These extensions conflict with LaTeX's way of disabling the commands within the table of contents or within the running header, because there, they were redefined to expect just a mandatory argument and then do nothing. We have now changed that behavior so that the redefinitions in these places accept an extended syntax. *(github issue 311)*

Bug fixes

Fix the use of `localmathalpabets`
In 2021 we introduced a method to overcome the problem that classic TeX engines (but not the Unicode engines) have only a limited number of math alphabets available that got easily fill up simply by loading math font packages, even if their symbols got used only occasionally. The idea was to avoid allocating all math alphabets globally, but instead allow a number of them (defined by counter `localmathalpabets`) to vary from one formula to the next. This way different formulas can make use of different alphabets and chances are much higher that the processing a complex the document succeeds. See [4] for details.

Unfortunately, the approach taken failed in some cases of nested formulas with the result that the wrong symbol glyphs were used. This has now been corrected. *(github issues 1101 1028)*

docstrip: Error if `.ins` file is problematical
If a file to generate had the same name as a preamble declared with `\declarepreamble` the preamble definition was overwritten because the macro used to store it was reused for denoting the output stream. The same problem happened with postambles declared with `\declarepostamble`. This is now detected and an error message is issued. To circumvent the issue in that case, simply use a different macro name for the preamble or postamble. *(github issue 1150)*

Prevent `cmd` hook from defining an undefined command
Using `\AddToHook{cmd/FOO/...}` when the command `\FOO` was undefined resulted in the command becoming `\relax`. Thus, if used, it no longer raised an "Undefined control sequence" error but silently did nothing. This behavior has been corrected and if the command `\FOO` isn't defined later, e.g., in a package, it now raises an error if it is used in the document. *(github issue 1591)*

Process global options once per package

In 2022, we introduced key–value (keyval) option processing in the kernel [5]. This also added the idea that keys could have scope: load-only, preamble-only and general use. However, we overlooked that an option given globally (in the optional argument to `\documentclass`) would be repeatedly processed and could therefore lead to spurious warnings. This has now been corrected: global options are seen exactly once per package by the keyval-based option handling system. *(github issue 1619)*

Make `\label`, `\index`, and `\glossary` truely invisible in running headers

LaTeX has had a bug since its initial implementation, in that it correctly ignored any `\label`, `\index`, or `\glossary` appearing in a mark, but neglected to handle the spaces around the command. As a result one could end up with two spaces in the running header when only one should be present. This was detected as part of working on issue 311 and has now been corrected. *(github issue 1638)*

Correct the float placement algorithm

When floats are added to the current or next page, LaTeX makes several tests to find an area that can receive the float. One of these tests calculates how much space is already used on the page and how much additional space is needed to place the float in a particular area. This means that it looks not only at the height of the float but also at the values from `\intextsep` (for `h` floats) or `\textfloatsep` and `\floatsep` (for `t` and `b` floats). The resulting space requirement is then stored in an internal variable and compared to the space still available on the page.

If the test fails, the algorithm tries the next area. Unfortunately, it was reusing the value in that internal variable as the starting point for the next test without removing the added space for the float separation (`\intextsep`, `\floatsep`, or `\textfloatsep`). Thus the comparison was being made with the wrong value (i.e., too high); therefore the test may have incorrectly concluded that a float doesn't fit, even though in fact it did.

This has now been corrected. *(github issue 1645)*

Correct `\CheckEncodingSubset`

In [6] and again in [7] we suggested that font maintainers should place an appropriate `\DeclareEncodingSubset` declaration into each `ts1`⟨*family*⟩`.fd` file so that it is tied to the font definition and available if a font family is explicitly selected through `\fontfamily{`⟨*name*⟩`}` instead of using some font support package. Unfortunately, doing this could result in incorrectly selected glyphs when the font encoding subset setting was evaluated before the `.fd` file was loaded (because then subset 9 was assumed). This has now been corrected and `\CheckEncodingSubset` now first loads the `.fd` file, if necessary. *(github issue 1669)*

Documentation

Clarifying space handling of `\textcolor`

In contrast to other `\text`-commands like `\textbf` or `\textrm`, the command `\textcolor` gobbles spaces at the start of its argument, so `Hello\textcolor{red}{␣World}` will output HelloWorld. There are technical as well as compatibility reasons for this, so the behavior will not change. This has now been clarified in the documentation. *(github issue 1474)*

Changes to packages in the graphics category

More accessibility keys in graphicx

The `\includegraphics` command now accepts `actualtext` and `artifact` keys, which by default do nothing but are used by the tagging code to provide an ActualText string and a boolean flag that the graphic is an artifact. *(github issue 1552)*

Changes to packages in the tools category

multicol: Full support for extended marks

In 2022 we introduced a new mark mechanism for LaTeX [5]. However, the initial implementation only covered the standard output routine of LaTeX. As a result the extended marks were not available within columns produced with the multicol package (where they would be especially useful). This limitation has finally been lifted and the new mechanism is now fully supported. *(github issue 1421)*

array: Improve preamble setup code for p and friends

While the preamble of a `tabular` or `array` is being built the arguments to `p`, `m`, or `b` columns got expanded several times. This is normally harmless because that argument contains usually just a dimension. However, in a case like `p{\fpeval{15}pt}` this resulted in an error, because `\fpeval` was expanded a few times, but not often enough to result in a single number. This has now been corrected and the argument is not expanded at all to allow for such edge cases as well as the extension available with the calc package, such as `p{\widthof{AAAAAA}}` (the latter was possible before but needed to be taken into account while the correction was implemented). *(github issue 1585)*

varioref: How to make `\reftextfaceafter`, etc. empty

In the case that one wants to make a command such as `\reftextfaceafter` produce nothing, one has to get rid of the space that is automatically placed in front of

the command. This can be done by simply defining the command to remove it, e.g.,

```
\renewcommand\reftextfaceafter{\unskip}
```

The varioref package does not test if such strings are empty, because that would require a lot of tests each time `\vref` is used, and it would nearly always find that the text is not empty. However, as shown above, the solution for this uncommon case is simple, and it is now explicitly documented in the package documentation.

(github issue 1622)

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

[2] LaTeX Project Team. *LaTeX 2ε news 1–41*. June, 2025. https://latex-project.org/news/latex2e-news/ltnews.pdf

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

[4] LaTeX Project Team. *LaTeX 2ε news 34*. November 2021. https://latex-project.org/news/latex2e-news/ltnews34.pdf

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

[6] LaTeX Project Team. *LaTeX 2ε news 36*. November 2022. https://latex-project.org/news/latex2e-news/ltnews36.pdf

[7] LaTeX Project Team. *LaTeX 2ε news 39*. June 2024. https://latex-project.org/news/latex2e-news/ltnews39.pdf

[8] LaTeX Project Team. *LaTeX 2ε news 40*. November 2024. https://latex-project.org/news/latex2e-news/ltnews40.pdf

[9] Frank Mittelbach, LaTeX Project Team. *The `ltmarks.dtx` code*. June 2025. https://latex-project.org/help/documentation/ltmarks-doc.pdf

[10] LaTeX Project Team. *The LaTeX 2ε Sources*. June 2025. https://latex-project.org/help/documentation/source2e.pdf