# LaTeX News

Issue 37, June 2023 (LaTeX release 2023-06-01)

## Contents

## New functionality offered as part of the "LaTeX Tagged PDF" project

We have now enabled new automatic tagging functionality for additional LaTeX elements, among them most display environments, standard sectioning commands, content, figure and table listings, floats and graphics and bibliographies. This can be activated through

`\DocumentMetadata{testphase=phase-III}`

At this point in time tagging support is only available for a restricted set of documents, i.e., those that use one of the basic document classes (`article`, `report`, and `book`) and only use commands and environments described in Lamport's LaTeX manual.

Using other document classes or adding additional packages in the preamble may work (or may partially work) but at this stage it is not very likely, at least not for packages or classes that excessively alter internals of LaTeX.

Also note that there are still several environments and commands described in the LaTeX manual that do not have tagging support yet, notably tabulars, `tabbing`, the various math environment and a few others. They will get this support as part of `phase-III`, but some of them will be delayed until after the June release.

A prototype for math tagging (including support for the `amsmath` environments) is already available, but it is mainly intended for experimentation and feedback and the resulting tagging is by no means the way we envision it to be eventually. If you would like to try it out use the following line:

`\DocumentMetadata{testphase={phase-III,math}}`

Note that the math tagging code at this point in time will clash with packages that redefine the `$` character (which then may lead to strange errors) and that packages that use math mode for non-mathematical constructs may result in surprising output as far as tagging is concerned. Feedback on which packages fail with the code in one or another way would be appreciated.

The `latex-lab` bundle contains various (still untagged) documentation files about the new code that can be accessed with `texdoc -l latex-lab`.

Feedback is welcome! Please use `https://github.com/latex3/latex2e/discussions/1010`.

## New or improved commands

### Extending hooks to take arguments

Hooks have always been containers for code whose outcome was entirely dependent on the contents of the hook alone. If any type of contextual information had to be passed to the hook, it had to be done by setting some variable before the hook so that the code in the hook could use that. But this is somewhat hard to keep track of, clumsy to implement, and it required the programmer to have some kind of "hook before the hook" to do that setup.

To make things a bit easier, lthooks was enhanced to support hooks with arguments. Hooks can now be declared and used with arguments, then the code added to these hooks can reference the hook's arguments using #1, #2, etc., so now hooks can behave more like macros than like *token lists* (using expl3 terminology). Regular argument-less hooks continue to work exactly like they did before: this extension is completely compatible with older documents and packages.

To declare a hook with arguments, use

`\NewHookWithArguments` {*hook*} {*num-args*}

then, similarly, to use the code in the hook, supposing a hook declared with 2 arguments, write

`\UseHookWithArguments` {*hook*} {2} {*arg1*} {*arg2*}

Or, if you want to add some code to a hook that takes arguments, write

`\AddToHookWithArguments` {*hook*} [*label*] {*code*}

exactly like you would for regular hooks, except that the ⟨*code*⟩ can use the arguments by referencing #1, #2, etc. In this case, if you want to add an actual parameter token (#) to the ⟨*code*⟩, you have to double it, as usual.

Additionally, if you want to add "regular" code to a hook with arguments, you can still use `\AddToHook` — in that case # tokens are *not* doubled. This means that a package author can decide to add arguments to an existing hook without worrying about compatibility: `\AddToHook` will do the right thing and will not mistakenly reference the newly added arguments.

The commands `\NewReversedHookWithArguments`, `\NewMirroredHookPairWithArguments`, `\AddToHookNextWithArguments`, `\UseOneTimeHookWithArguments`, and the expl3 counterparts of the commands discussed in this section were also added. The complete documentation can be found in the lthooks documentation [2].

*Generic cmd hooks with arguments:* Along with the possibility of passing arguments to a regular hook as discussed above, generic `cmd` hooks can now access the arguments of the command they are patched into, using the interface described in the previous section.

For example, if you were to add some code to the `\title` command using hooks, you could access the actual title given in the argument. Thus, to write the title of the document in the terminal you could use:

```
\AddToHookWithArguments{cmd/title/before}
  {\typeout{Document title: #1}}
```

As with regular hooks, code added to a `cmd` hook using `\AddToHook` will not be able to access the command's arguments. This means that, as with regular hooks, this change is completely backwards compatible, so previous usages of `cmd` hooks will work exactly as they did before.

### Providing copy and show functions for environments

To copy a command definition we introduced `\NewCommandCopy` in 2022. This even allows you to copy commands that consist of several internal components, such as robust commands or those with a complex signature. To do the same with environments, e.g., to define the environment `myitemize` to be equivalent to `itemize`, you can now write

```
\NewEnvironmentCopy{myitemize}{itemize}
```

There are also `\Renew...` and `\Declare...`, which may be useful depending on the circumstances.

In addition, we offer a `\ShowEnvironment` command, which displays the `\begin` and `\end` code of the environment passed as an argument. E.g., `\ShowEnvironment{center}` results in the following output:

```
> \begin{center}=environment:
> ->\trivlist \centering \item \relax .
<recently read> }
l. ...\ShowEnvironment{center}
> \end{center}:
> ->\endtrivlist .
<recently read> }
l. ...\ShowEnvironment{center}
```

*(github issue 963)*

### \IfFileAtLeastTF

The 2020-10-01 LaTeX release introduced the CamelCase tests `\IfClassAtLeastTF` and `\IfPackageAtLeastTF` for checking class and package dates. We have now added `\IfFileAtLeastTF` to allow the same to happen for generic files which contain a `\ProvidesFile` line.

*(github issue 1015)*

### \DeclareLowercaseMapping, \DeclareTitlecaseMapping and \DeclareUppercaseMapping

The move from a case-changing approach using `\lccode` and `\uccode` data to one where information is stored by

a kernel-managed structure left a gap in the ability of the user to *tune* the case changing outcomes. This has now been addressed by the addition of three commands

- \DeclareLowercaseMapping
- \DeclareTitlecaseMapping
- \DeclareUppercaseMapping

which can be used to customise the outcome for codepoints. This can be applied generally or to a specific locale (see also the next section). A small number of pre-defined customisations have been set up in the kernel where the outcomes for pdfTeX should be different for those from Unicode engines. For example

\DeclareUppercaseMapping{"01F0}{\v{J}}

allows ǰ to be produced in 8-bit engines: without this customisation, an error would occur as there is no pre-composed ǰ in Unicode. More detail is given in usrguide. *(github issue 1033)*

### \BCPdata

Improvements in the Unicode handling for case changing have highlighted that the kernel has not to-date been locale-aware. The packages babel and polyglossia provide comprehensive locale support, but did not have an agreed unified interface to pass that information back to other code. Following discussion with the maintainers of those two bundles, the kernel now defines \BCPdata as a stub (so it is always defined), and babel and polyglossia will redefine it to provide the locale data. An agreed set of keywords mean that \BCPdata can be queried in a structured way by both the kernel and any other "consumer" packages. *(github issue 1035)*

### Improve \samepage

The \samepage declaration sets various parameters to 10000 to prevent undesired page breaks. The \predisplaypenalty parameter has already by default a value of 10000, and to save space in the past it was therefore not explicitly set. However, there are a few classes that change the parameter and as result the user might experience a page break in front of a display formula within the scope of \samepage when using such classes. This has now been corrected and \predisplaypenalty is also explicitly set to 10000. *(github issue 1022)*

### Groups in \MakeUppercase

Prior to 2022, \MakeUppercase and \MakeLowercase used a brace group around their argument so providing a scope for any declarations within the argument. This grouping has been restored (also for \MakeTitlecase), although the underlying L3 text case commands do not use grouping. *(github issue 1021)*

### Extension of the \label command

Previously, in standard LaTeX, the \label command wrote a \newlabel declaration into the .aux file and stored two values in the second argument of this \newlabel command: \@currentlabel, which normally contains the state of the current counter and \thepage for the current page number.

The packages hyperref and nameref then patched the \label command to store five values instead. In addition to the above they saved \@currentlabelname, which normally contains the current title text and can be retrieved with \nameref, and \@currentHref, which is the name of the destination needed to create an active link. The fifth argument was only used if external references were loaded with the xr-hyper package.

Starting with this release, the number of values stored in \newlabel has been unified. \label now writes a \newlabel command that always contains five values in the second argument (each in a brace group): \@currentlabel, \thepage, \@currentlabelname, \@currentHref, and \@kernel@reserved@label@data (which is reserved for the kernel).

Additionally, a hook with the name label has been added. It takes one argument: the label string. Code added to the hook can refer to this argument with #1. The hook is executed directly before the \label command writes to the .aux file but *after* the \@bsphack command has done its spacing magic, and it is located *inside* a group; thus, its code only affects the write operation.

## Code improvements

### Performance in checking file existence

The addition of hooks, etc., to file operations had a side effect of making multiple checks that the file existed. In larger documents using many files, these file system operations caused non-trivial performance impact. We now cache the existence of files, such that these repeated filesystem calls are avoided.

### doc: Handle \␣ correctly in the index

Due to some problems in the code it wasn't possible to prevent \␣ from showing up in the index—\DoNotIndex{\ }, etc. had no effect. This has now been corrected. *(github issue 943)*

### doc: Support the upquote package

The default quote and backquote characters in typewriter fonts are typographical quotes, e.g., the input

\verb/`prog 'my input''/

is rendered as `prog 'my input'` and not as `prog 'my input'` as preferred by many programmers. This can be adjusted, for example, with the upquote package, which results in the second output. However,

for historical reasons `doc` had its own definition of `\verb` and `verbatim` and as a consequence the two packages did not cooperate. This has now been fixed and loading `upquote` together with `doc` has the desired effect. *(github issue 953)*

### Default definition for `\do`

The command `\do` with its nice public name is in reality an internal command inherited from plain TeX for list processing. However, it only got a definition when `\begin{document}` was executed, with a result that a user definition in the preamble was unconditionally overwritten at this point. To properly alert the user that this command is not freely available we now make a definition in the format, so that `\newcommand` and friends produce a proper error message instead of allowing a definition that doesn't last. *(github issue 975)*

### New key for `filecontents`

The `filecontents` environment warns on the terminal if a file gets overwritten even if that is intentional, e.g., when you write a temporary file over and over again. To make the warning less noisy in this case we added a new `nowarn` key that redirects the overwriting warning to the transcript file. We think that some record of the action is still required to help with debugging, thus it is not completely silenced. The warning that nothing gets written, because the file already exists (and the `force` key was not used), is not altered and still shows up on the terminal. *(github issue 958)*

### A further hook for shipping out pages

Since October 2020 the shipout process offers a number of hooks to adjust what is happening before, during, and after the `\shipout`. For example, with the `shipout/before` hook, packages can reset code they have altered (e.g., `\catcode`s during verbatim-like processing) and with `shipout/background` and `shipout/foreground` material can be added to the pages. Details are given in [1].

However, still missing was a hook that allows a package writer to manipulate the completed page (with foreground and background attached) just before the actual shipout happens. For this we now provide the additional hook `shipout`. One use case (sometimes needed in print production) is to mirror the whole page via `\reflectbox` including all the extra data that may have been added into the fore- or background. *(github issue 920)*

### Displaying release information in the `.log`

LaTeX displays its release information at the very beginning of the LaTeX run on the terminal, and also writes it to the transcript file if that is already opened at this point. While this is normally true, it is not the case if the LaTeX run was started passing additional TeX code on the command line, e.g.,

```
pdflatex '\PassOptionsToClass{11pt}{article}
          \input{myarticle}'
```

In this case the release information is displayed when `\PassOptionsToClass` is processed but the transcript file is only opened when the output file name is known, i.e., after `\input` has been seen, and as a result the release information is only shown on the terminal.

To account for this scenario, we now repeat the release information also at the very end of the transcript file where we can be sure that it is open and ready to receive material. *(github issue 944)*

## Bug fixes

### Incompatibility between doc and unicode-math

The `unicode-math` package alters the catcode of | but does not adjust its value for use in `doc`, with the result that "or" modules, i.e., $\langle A|B \rangle$ are displayed in a strange way. This is now fixed with some firstaid code that will eventually be moved into `unicode-math`. *(github issue 820)*

### A fix for `\hspace`

The change to `\hspace`, done in 2020 to make it `calc`-aware, had the unfortunate side effect that starting a paragraph with `\hspace` would result in the execution of `\everypar` inside a group (i.e., any local changes would immediately be revoked, breaking, for example, `wrapfig` in that special situation). This got fixed with the 2022-11 PL1 hotfix, so was already corrected in the previous release, but is only now documented in the newsletter. *(github issue 967)*

### Ensure that `\cs` is defined in ltxdoc

The class `ltxdoc` defined the command `\cs` to typeset a command name with a backslash in front. This definition was moved to the `doc` package itself. This meant that it was suddenly missing when reverting to the old `doc` package implementation via the class option `doc2`. This has now been corrected. *(github issue 981)*

### Improve spacing at top of `minipages`

A list and several other document elements add some vertical space in front of them. However this should not happen at the beginning of a box (such as a `minipage`) and normally it doesn't, because TeX automatically drops such space at the start of a vertical list. However, if there is some invisible material, such as a `\color` command, a `hyperref` anchor, a `\write` or other such items, then the list is no longer empty and TeX no longer drops the vertical space.

With the new paragraph handling introduced in 2021 it is now finally possible to detect and avoid this

problem and apply appropriate counter measures so that from now on the spacing will always be correct.

*(github issue 989)*

### A fix for `\NewCommandCopy` and `\ShowCommand`

When copying and showing definitions of (non-expandable) document commands (a.k.a. commands defined by `\NewDocumentCommand` and friends) containing empty or only `m`-type arguments, these commands were wrongly recognized as expandable ones. This is fixed in the present LaTeX release. *(github issue 1009)*

### Corrections for switching math version

Some internal code improvements improve support for switching math version when nested within an outer math expression. This will improve `\boldsymbol` and `\bm` and similar commands. *(github issue 1028)*

### Allow par as a filename

`\input{par}` or `\includegraphics{par}` could give spurious errors. This has been fixed by making an internal command `\long`. *(github issue 942)*

### Correct setting of `\endlinechar` in +v arguments

In the particular case of a document command with a `+v`-type argument used inside `\ExplSyntaxOn/Off`, newlines would be misinterpreted as spaces because the value of `\endlinechar` was set too late. This has been fixed, and now newlines are correctly translated to "`the character ^^M`". *(github issue 876)*

### Correct handling of hooks with only 'next' code

When `\AddToHookNext` was used on a not-yet-declared hook, that hook would be incorrectly identified as empty by `\ShowHook`. Also, if that hook was later declared, that 'next' code would not be executed. This has been fixed by correctly initializing the hook structure when `\AddToHookNext` is used. *(github issue 1052)*

### Ignoring space after $$

Space is normally ignored after a closing `$$`, but internal LaTeX font handling code could interfere if `\eqno` was used. `\eqno` and `\leqno` have been redefined to add `\ignorespaces` after the math group. *(github issue 1059)*

## Documentation improvements

### Updates to the guides

When LaTeX $2_\varepsilon$ was released, the team provided documentation for both document authors and package/class developers in the two files `usrguide` and `clsguide`. Over time, the team have augmented these documents as new methods have been added to the kernel. However, they retained their structure as assuming familiarity with LaTeX 2.09. This meant that for new users, there

was material which is no longer relevant, and less clarity than desirable regarding the approaches that are recommended today.

The two files have now been (partially) re-written, with the versions available previously now frozen as `usrguide-historic` and `clsguide-historic`. More material has been carried forward in the class/package guide than in the user guide, but both are worth a re-read by experienced LaTeX users.

### Displaying the exact release dates for LaTeX

In some situations it is necessary to find out the exact release dates for older versions of the LaTeX format, for example, when you need to use different code in a package depending on the availability of a certain feature and you therefore want to use `\IfFormatAtLeastTF{⟨date⟩}` or the rather horrible construction `\@ifl@t@r\fmtversion{⟨date⟩}`, if you want to cater for formats that are older than 2020.

Or you know that your package is definitely not going to work with a format before a certain ⟨date⟩, in which case you could use `\NeedsTeXFormat{LaTeX2e}[⟨date⟩]` to ensure that users are alerted if their format is too old.

The big problem is knowing the exact ⟨date⟩ to put into such commands; in the past, that was not that easy to find. You could have looked in the file `changes.txt`, but that is hidden somewhere in your installation and if you try `texdoc -l changes.txt` you get more than thirty results and the right file is by no means the first.

Yukai Chou (**@muzimuzhi**) kindly provided a patch for this, so that we now have the exact dates for each LaTeX format listed in an easy to remember place: in `ltnews.pdf` and that file conveniently also contains all major features and changes to LaTeX over the years—one of which is most likely the reason you need the ⟨date⟩ in the first place.

The date is now given in parentheses in the newsletter title, thus this newsletter tells you that on 2023-06-01 the command `\NewEnvironmentCopy`, a new `shipout` hook, etc. was made available. And looking into `ltnews.pdf` you can now easily find out that the LaTeX3 programming layer was added on 2020-02-02 (because the date was so nice) and not on the first of the month. *(github issue 982)*

### Fresh from the press: "The LaTeX Companion, third edition" is now in print

The third edition of *The LaTeX Companion* is now available. This is the result of five years of careful work and we hope that it will provide our readers with all the information they need to successfully navigate the LaTeX ecosystem and efficiently produce beautiful documents.

Since the publication of the last edition (2004), a lot has happened in the LaTeX world and thus a complete rewrite was necessary. All chapters have been thoroughly

revised, and in many cases significantly extended, to describe new important functionality and features. More than 5,000 add-on packages have been analyzed in detail, out of which roughly 10% have been chosen for inclusion in *The LaTeX Companion.* All important aspects of these packages are described to provide the user once again with a satisfying one-stop–shop experience for the decade to come.

To cover what we thought worth describing today, the book nearly doubled in size. The print edition is therefore produced as a two-volume set and sold as a bundle. Both volumes come as hardcover with ribbons to easily mark pages in the book.

To give you an idea of what is covered in the third edition you can find some excerpts at

> https://www.latex-project.org/news/
> 2023/03/17/TLC3

The edition is also available as an eBook (Parts I and II combined) consisting of PDF and ePub format, without DRM. Finally, the publisher offers the combination of the printed books and the digital versions at a very attractive price not available anywhere else.

## Changes to packages in the tools category

### multicol: Better support for CJK languages

The default minimum depth of each column in a `multicols` corresponds to the depth of a "p" in the current font. This helps to get some uniformity if rules are used between the columns and makes sense for Latin-based languages. Until now it was hard-wired, but for CJK (Chinese/Japanese/Korean) languages it is better to use a zero depth, because there all characters have the same height and depth. And even with Latin-based languages one might want to use the depth of a `\strut` or that of a parenthesis. So we now offer a way to adjust this while maintaining backward compatibility: redefine `\multicolmindepthstring` to hold whatever you want to get measured for its depth (the width is not relevant).

*(github issue 698)*

### multicol: Fix handling of nested environments

If `multicols` environments have been nested into each other (the inner one boxed) it could fail if the boxed environment appeared near a page break. The problem was that the output routine was called while the `\hsize` was still altered to fit the column width of the inner `multicols` — thereby messing up the placement of columns of the page. This has now been fixed.

*(github issue 1002)*

## References

[1] Frank Mittelbach, LaTeX Project Team:
*The* `ltshipout` *documentation.*
Run `texdoc ltshipout-doc` to view.

[2] Frank Mittelbach, Phelype Oleinik,
LaTeX Project Team: *LaTeX's hook management.*
Run `texdoc lthooks-doc` to view.