LaTeX News

Issue 34, November 2021 (LaTeX release 2021-11-15)

Contents

Introduction

This release of LaTeX does not contain any major new modules, but is focused around consolidation and improvements of the functionality introduced in previous releases. In addition, various smaller enhancements and bug fixes have been added to the kernel and the core packages.

Hook business

Since the introduction of the hook management system in the 2020 release of LaTeX [4] package developers have started to make more and more use of this new functionality. One result of this increased activity has been a number of queries which show that some of the documentation was not precise enough and that some clarifications were needed; these deficiencies have now been addressed in the documentation. The increased usage has also revealed a small number of errors that we thought should be corrected now, while the adoption rate is still relatively small; the following problems have therefore been addressed in this release.

Provide `\ActivateGenericHook`

The hook management system offers a number of generic hooks, i.e., hooks whose names contain a variable component such as the name of an environment. Predeclaring such hooks is not feasible, so these hooks use a different mechanism: they are implicitly available, springing into life the moment a package, or the document preamble, adds any code to one by using `\AddToHook`. The kernel offers such hooks for environments (`env/...`) and commands (`cmd/...`), and also for files, packages and classes (`file/...`, `include/...`, `package/...`, `class/...`).

It is also possible to offer such generic hooks in packages if, for example, hooks are needed that depend on the current language and therefore need the language name as part of the hook name (but you probably don't know beforehand all the necessary names).

If you want to offer such generic hooks, you can now do this by using `\UseHook` or `\UseOneTimeHook`

LaTeX News, and the LaTeX software, are brought to you by the LaTeX Project Team; Copyright 2021, all rights reserved.

in your (package) code, but *without declaring the hook* with `\NewHook`. However, without further work, a call to `\UseHook` with an undeclared hook name will do nothing; so, as an additional setup step, it is necessary to explicitly activate the generic hook by using `\ActivateGenericHook`.[1]

Assuming that you don't know all the different hook names up front, it will remain the task of the users of your package to activate the hook themselves before adding code to it. For example, Babel offers hooks such as `babel/`⟨*language*⟩`/afterextras` that enable a user to add language specific declarations to these "extras". One can then write

```
\ActivateGenericHook
       {babel/ngerman/afterextras}
\AddToHook{babel/ngerman/afterextras}
       {\color{blue}}
```

after which all German words would be colored blue in the text.

Note that a generic hook produced in this way is always a normal hook.

Standardized names for the generic hooks

The initial set of generic hooks provided by the kernel had two patterns of names: ones like `env/`⟨*name*⟩`/after`, with the variable, ⟨*name*⟩, part in the middle position; and ones like `file/after/`⟨*name*⟩, with the variable part in the third position. The coexistence of these two types caused confusion because the user had to remember in which position the variable part was supposed to go; and it also made the code more complicated and slower.

The file-related hooks have therefore been renamed so that the variable part of the name is in the middle, as with all other hooks. The changes are listed here:

Old name	New name
`file/before/`⟨*name*⟩	→ `file/`⟨*name*⟩`/before`
`file/after/`⟨*name*⟩	→ `file/`⟨*name*⟩`/after`
`package/before/`⟨*name*⟩	→ `package/`⟨*name*⟩`/before`
`package/after/`⟨*name*⟩	→ `package/`⟨*name*⟩`/after`
`class/before/`⟨*name*⟩	→ `class/`⟨*name*⟩`/before`
`class/after/`⟨*name*⟩	→ `class/`⟨*name*⟩`/after`
`include/before/`⟨*name*⟩	→ `include/`⟨*name*⟩`/before`
`include/end/`⟨*name*⟩	→ `include/`⟨*name*⟩`/end`
`include/after/`⟨*name*⟩	→ `include/`⟨*name*⟩`/after`

Since this is a breaking change, the old names will still work for a while so that users and package authors have enough time to adjust; but a warning will be issued when the old names are used. Eventually the deprecated names will be turned into errors and then removed completely. *(github issue 648)*

[1] Note that in the previous release we offered `\ProvideHook` as a means to achieve this effect, but the name was badly chosen so we decided to deprecate it and now offer `\ActivateGenericHook` instead.

Some file hooks made one-time

Classes, packages and included files can only be loaded once in a LaTeX document. For this reason, the hooks that are specific to loading such files have been made one-time hooks. Beside being more efficient, this supports the following important use case

```
\AddToHook{package/varioref/after}
 {... apply when the package gets loaded,
  or apply now (if it is already loaded) ...}
```

without the need to first test whether the package is already loaded. *(github issue 626)*

Clearing extra hook code for the next invocation

There are a few use cases where it is helpful if one can cancel an earlier use of `\AddToHookNext`: for example, when a page is discarded with `\DiscardShipoutBox` because only some pages of the document are printed. For such situations the new command `\ClearHookNext` is now provided. *(github issue 565)*

Cleaning up after `\UseOneTimeHook`

Some hooks are meant to be used only once in a document, and any further attempt to add code to one of these will cause the code to be executed immediately instead of being added to the hook. The initial implementation of this concept was very simple and didn't anticipate that packages may try to execute a one-time hook several times, resulting in the hook code being executed repeatedly. Thus the implementation was fine for simple cases (such as the `begindocument` hook) but it causes trouble if the one-time hook was intended, for example, as an initialization hook that is used just once (when a command is first called) but is then ignored in further calls.

This deficiency has been addressed, and now a one-time hook will only be executed once, with its code being removed after use to free up some memory. *(github issue 565)*

`\RemoveFromHook` with a missing code label

In the first version of `\RemoveFromHook`, when the code label to be removed didn't exist in the hook a "removal order" would be queued; and then, the next time something tried to add that label to the hook, this `\AddToHook` action would be cancelled by the removal order, so that no code would be added that one time. This was so that, in principle, package loading order wouldn't matter. However, this implementation didn't work as intended because, while two `\AddToHook` actions with a given label would be removed by a single `\RemoveFromHook`, one `\RemoveFromHook` could not cancel two `\AddToHook` actions for that label; this caused confusion and also led to further problems.

The implementation has now been changed, so that `\RemoveFromHook` removes only code labels that already

exist in a hook: it will display a warning if there is no such code label.

Note that, whereas when working with a single package you should use `\RemoveFromHook` to remove a code label, when working with more than one package, the `voids` relation should preferably be used. This is best because this relation is non-destructive (meaning that it can be reverted later by using another relation), and it is also truly independent of package loading order.

(github issue 625)

Patching commands with parameter tokens

In the last release, LaTeX's hook mechanism was extended to add support for hooking into commands using generic `cmd` hooks (see [5]). That version of the extension had a bug: the patching of some commands that contained a parameter token (normally `#`) in their definition would fail with a low-level TeX error. This has now been fixed so that patching now works for those commands as well. *(github issue 697)*

New or improved commands

`\NewCommandCopy` and `\ShowCommand` extended

Since the 2020-10-01 release (see [4]), LaTeX has provided `\NewCommandCopy` to copy robust commands, and `\ShowCommand` to show their definitions on the terminal. In that same release, the xparse package was integrated into the kernel (as ltcmd) to offer `\NewDocumentCommand`, etc. However, the extended support for `\NewCommandCopy` and `\ShowCommand` was not implemented in ltcmd. The present LaTeX release implements this support, so now commands defined with `\NewDocumentCommand` and friends can also be copied, and their definitions can be easily shown on the terminal without the need for "`\csname` gymnastics".

(github issue 569)

Undo math alphabet allocations if necessary

TeX, or more exactly the 8-bit versions of TeX, such as pdfTeX, have a hard limit of 16 on the number of different math font groups (`\fam` or `\mathgroup`) that can be used in a single formula. For each symbol font declared (by a package or in the preamble) an extra math group is allocated, and the same happens for each math alphabet, (such as `\mathbf`) once it gets used anywhere in the document. Up to now, these math alphabet allocations were permanent, even if they were used only once; the result was that in complex documents you could easily run out of available math font groups. The only remedy for this was to define your own math version, which is a complicated and cumbersome process.

This situation has now been improved by the introduction of a new counter `localmathalphabets`: this counter governs how many of the math group slots are assigned locally when a new math alphabet (and a new math group) is needed. Once the current formula is finished, every such further (local) allocation is undone, giving you a fighting chance of being able to use different new math alphabets in the next formula.

The default value of `localmathalphabets` is 2, but if you need more local alphabets because of the complexity of your document, you can set this to a higher value such as 4 or 5. Setting it even higher is possible, but this would seldom be useful because many group slots will be taken up by symbol fonts and such slots are always permanently allocated, whether used or not.

(github issue 676)

New default value for `\tracinglostchars`

In 2021 all TeX engines were enhanced so that `\tracinglostchars` supported the value `3` to turn missing characters into errors and not just warnings. This engine change made us realize that LaTeX should set a better default value for this parameter (previously, the warning was written only to the transcript file). Using the now available value of `3` as the default would be ideal, but for compatibility reasons we have only increased it to `2` in the kernel. However, we recommend setting `\tracinglostchars=3`, in either a package or the preamble of your documents: this is because having missing glyphs in the output is definitely an error and should therefore be flagged as such (to ensure that it gets proper attention). Further reasons, related especially to Unicode engines, for making this recommended change are explained later in this newsletter (in connection with the misuse of text accents in math mode).

`\PackageNote` and `\ClassNote` added

LaTeX offers these three commands: `\PackageError` to signal errors that stop the processing; `\PackageWarning` to generate a warning message on the terminal but continue with the processing; and `\PackageInfo` to provide some information that is only written to the `.log` file but not sent to the terminal. What has not existed up to now is a way to provide information on the terminal that identifies itself as coming from a specific package but which does not claim to be a warning. (Packages that wanted to write to the terminal used `\PackageWarning` even though the information was not in fact a warning.)

We have therefore now added `\PackageNote` (and the closely related `\PackageNoteNoLine`); these identify themselves as "informational", but they still go to the terminal and not only to the `.log` file. Similar commands exist for classes and so there too we have new commands: `\ClassNote` and `\ClassNoteNoLine`.

(github issue 613)

New \ShowFloat command
The package fltrace offers a (fairly low-level but very detailed) way to trace LaTeX's float mechanism. This can help in understanding why a certain float is placed into a certain region, or why it shows up unexpectedly on a later page. LaTeX stores floats in registers named `\bx@A`, `\bx@B`, etc., and these names show up in the tracing information.

To display the contents of a float register, you can now say `\ShowFloat{`*identifier*`}` where *identifier* is the uppercase letter (or letters) after `bx@` in the register name shown in the tracing. If additional registers have been allocated (with `\extrafloats`), the *identifier* can also be a number. The command is generally available, whether or not you have loaded fltrace, because it is also useful when interpreting the tracing output of the fewerfloatpages package.

New argument for \counterwithin/without
The commands `\counterwithout` and `\counterwithin` each now has an additional optional argument, similar to that of the command `\numberwithin` from amsmath, for which these are now the preferred replacements. This optional argument specifies the format of the counter, such as `\roman`; the default value is `\arabic`. Alternatively, you can use a starred form, in which case the format of the counter is not altered at all.

Tests for package and class loading
To test whether a package has been loaded you can now use `\IfPackageLoadedTF {`⟨*package*⟩`} {`⟨*true*⟩`} {`⟨*false*⟩`}` and, based on the result, execute different code. It is also possible to check whether the package was loaded with certain options. This is done with `\IfPackageLoadedWithOptionsTF`. It takes four arguments: `{`⟨*package*⟩`}{`⟨*option-list*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`. It uses the ⟨*false*⟩ code if one or more options in the ⟨*option-list*⟩ were not specified when loading the package, or if the package has never been loaded. Both commands can be used anywhere in the document, i.e., they are not restricted to the preamble.[2]

For classes, similar commands, with `Package` replaced by `Class` in the name, are provided. *(github issue 621)*

Better handling for a misuse of \include
The command `\include` has by now been used quite often, but erroneously, to input a variety of files in the preamble of the document (before `\begin{document}`). Therefore LaTeX now warns about such bad use of `\include`. As a recovery action it will nevertheless input the specified file if it exists (this is as before). Note, however, that this is now done without any adjustments to the `.aux` file settings and without running the `\include` file hooks (only the generic file hooks from `\InputIfFileExists` are run). *(github issue 645)*

[2] This is now also true for the corresponding internal commands, e.g., `\@ifpackageloaded`, that had this restriction in the past.

Code improvements

Use OpenType version of Latin Modern Upright Italic font
When a Latin Modern font is used with the TU encoding under XeTeX or LuaTeX and fontshape `ui` is requested, LaTeX now uses the OpenType version of the font instead of substituting the (T1-encoded) Type 1 version.

Additional Extended Latin characters predefined
More characters, such as ḱ (U+1E31), are now predefined and do not need a `\DeclareUnicodeCharacter` declaration. *(github issue 593)*

Check \endfoo in \NewDocumentEnvironment
The `\newenvironment` command has always checked that neither `\foo` nor `\endfoo` exists before creating a `foo` environment. In contrast (for historical reasons) the more recently introduced command `\NewDocumentEnvironment` checked only for `\foo`. The behavior of `\NewDocumentEnvironment` now aligns with that of `\newenvironment`, except that it gives distinct errors concerning the existence of `\foo` and `\endfoo`.

Improve the error message `\begin ended by ...`
In the past it was possible to get an error message along the lines of "`\begin{foo} ended by \end{foo}`". This could happen when the environment name was partly hidden inside a macro. It happened because the test was comparing the literal strings, whereas in the error message these got fully expanded. This has now been changed to show a more sensible error message. *(github issue 587)*

Pick up all arguments to \contentsline
A `\contentsline` command in the `.toc` file is always followed by four arguments, the last one being empty except when using the hyperref package. The `\contentsline` command itself only used the first three arguments and it relied on the fourth being empty (and thus doing no harm). But this assumption is not always correct: e.g., if you at first decide to load hyperref but then later you remove this loading from the preamble. So now all four arguments are picked up, with the fourth being saved away so that it can be used by hyperref. *(github issue 633)*

Allow dropping a math list in LuaTeX callback
The LuaTeX callbacks `pre_mlist_to_hlist_filter` and `post_mlist_to_hlist_filter` no longer create an error when the callback handler indicates removal of the entire math list. *(github issue 644)*

Extended label handling in package code

Since 2020, as noted in LaTeX News 32 [4], LaTeX has recorded the name of the counter associated with the current label in the internal command `\@currentcounter`. This facility (originally from the zref package of Heiko Oberdiek) can be used to generate prefixes such as "Figure" before the reference text, as long as the counter is not counting different objects in a single sequence (e.g., lemmas and theorems). In the most common cases the current label is set by `\refstepcounter`, which automatically stores the counter name; but some constructs (alignments and footnotes) may need to store the current label directly and so for these it is useful to update additionally `\@currentcounter` so as to store this counter name.

In this release both the footnote command in the kernel and also some of the environments in the amsmath package have been updated in this way. We encourage the maintainers of any class or package files that define `\@currentlabel` to also set `\@currentcounter` at the same point. *(github issue 300, 687)*

Better message if text accent used in math mode

Using text accents like `\^` in math does not work (and TeX explicitly provides math accents such as `\hat` for accessing such symbols in math mode). Therefore LaTeX issued a warning when such a wrongly placed accent was encountered and this was often followed by a strange, and apparently unrelated, low-level error. This has now been changed so that the message from this error is at least about accents, which we hope is less puzzling.

Discussion of such warnings or errors reminds us to reinforce here a recommendation from earlier in this newsletter, as part of the item on the value of `\tracinglostchars`. Using TeX implementations from 2020 onwards, any warning that concerns missing characters can be converted to an error by setting `\tracinglostchars` to 3; we therefore now recommend changing this setting to 3, especially for Unicode engines where such missing characters are common (because no font supports the full Unicode range). *(github issue 643)*

Bug fixes

Replicate argument processors for all embellishments in command declarations

There was a bug in ltcmd (formerly xparse) that caused commands to misbehave if they were defined with embellishments and argument processors. In that case, only one (possibly void) argument processor would be added to the full set of embellishment arguments, resulting in too few processors in some cases and thus leading to unpredictable behavior. This bug has been fixed by applying the same argument processors to all the embellishments in a set, so that a declaration like:

```
\NewDocumentCommand\foo{>{\TrimSpaces}e{_^}}
                       {(#1)[#2]}
\foo^{ a }_{ b }
```

will now correctly apply `\TrimSpaces` to both arguments. *(github issue 639)*

Correct case changing of `\ij` and `\IJ`

The ligatures "ij" and "IJ", as used in Dutch, are available (for most TeX fonts) only when the commands `\ij` or `\IJ` are used, or when you enter them as the Unicode characters U+0133 or U+0132. However, when using `OT1` or `T1` encoded fonts in pdfTeX, the upper or lower casing with `\MakeUppercase` and `\MakeLowercase` would always fail regardless of the input method. This has now been corrected. At the same time we improved the hyphenation results for words containing this ligature (when using the `OT1` encoding). *(github issue 658)*

Legacy font series default changes

In the past, changes to the font series defaults were made by directly altering `\bfdefault` or `\mddefault`. Since 2020 there is now `\DeclareFontSeriesDefault` that allows more granular control: with this declaration you can alter the default for individual meta font families by, for example, changing the bold setting only for the sans serif family, without changing it for `\rmfamily` or `\ttfamily`. See [3] for more details.

For backwards compatibility, changing `\bfdefault` with `\renewcommand` remained possible; if used, this alters the setting for all meta families in one go. This alteration cannot be done when the `\renewcommand` happens and it was therefore delayed until the next time `\bfseries` or `\mdseries` was executed. However, the problem with that approach was that any call to `\DeclareFontSeriesDefault` in the meantime was overwritten; thus, these two approaches didn't work well in combination. There was a problem because older font packages use the legacy method while newer ones use `\DeclareFontSeriesDefault`.

This has now been resolved by changing `\DeclareFontSeriesDefault` to do any necessary resetting prior to setting the new defaults. *(github issue 663)*

Use of `#` in `\textbf` and similar commands

Previously you could not use the macro parameter character `#` in inline functions within the argument of `\textbf` or similar text font commands. An internal definition is now guarded with `\unexpanded` so that the use of `#` here no longer generates an error. *(github issue 665)*

Changes to packages in the amsmath category

Improved compatibility with hyperref

This change in amsmath fixes a spacing problem caused by the method used in hyperref to change the `equation`

environment. For simplicity, an explicit, low-level (hence possibly temporary) patch has been added to amsmath: this consists of an extra, empty (hence invisible) `\mathopen` atom (with no mathematical meaning) at the start of the environment's mathematical content.
(github issue 652)

Changes to packages in the graphics category

graphicx: New key, for alt text

A new key, `alt`, has been added to `\includegraphics` to support the addition of descriptive text that is important for accessibility. This key is unused by default; it can be deployed by extension packages and it will provide useful support for other future possibilities. *(github issue 651)*

Changes to packages in the tools category

array: No `\mathsurround` around a `tabular`

A `tabular` environment is typeset (internally) as an `array` environment with special settings, and it therefore uses (hidden) math mode. Since it is not in fact a math formula, no extra space from `\mathsurround` should be added (the spacing around the `tabular` should not get changed). Note that this bug has been present "forever", which shows that `\mathsurround` is never used, or at least its use is never noticed. At any rate, this bug has now finally been fixed. *(github issue 614)*

longtable: Improvements after a section heading

The `longtable` environment now sets the `\@nobreakfalse` flag to correct the typesetting when a table immediately follows a heading. Previously the spacing and indentation changes that are required immediately after a section heading were incorrectly triggered within the next paragraph (if any) following the table. A similar test for `\if@noskipsec` has been added, so that a table is correctly placed after a run-in heading rather than appearing before that heading.
(github issues 131 and 173)

multicol: Better column break control

From version 1.9 onwards `\columnbreak` accepts an optional argument (like `\pagebreak`) in which you can specify the desirability of breaking the column after the current line: supported values are `0` to `4`, with higher numbers indicating increased desirability. This version also adds `\newcolumn`, which forces a break but runs the column short (comparable to `\newpage` for pages).
(github issue 682)

varioref: Improved handling of missing labels

If an undefined label is referenced, varioref makes a default definition so that later processing finds the right structure (two brace groups inside `\r@`⟨*label*⟩) However, if nameref or hyperref is loaded, this data structure changes to having five arguments; this could cause low-level errors in some cases. The code has therefore now been changed to avoid these errors.
(https://tex.stackexchange.com/q/603948)

References

[1] Frank Mittelbach and Chris Rowley: *LaTeX Tagged PDF—A blueprint for a large project.* https://latex-project.org/publications/indexbyyear/2020/

[2] *LaTeX documentation on the LaTeX Project Website.* https://latex-project.org/help/documentation/

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

[4] LaTeX Project Team: *LaTeX 2ε news 32.* https://latex-project.org/news/latex2e-news/ltnews32.pdf

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