LaTeX News

Issue 34, November 2021 — Draft Version (with many unfinished blocks)

Contents

Introduction

write

???

write

Hook business

After 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 the new functionality. This resulted in a number of queries showing that some of the documentation was not precise enough and that one or the other clarification was needed. This has now been addressed in the documentation. The extended usage also showed a small number of deficiencies that we thought should be corrected now while the adoption rate is still relatively small. These are addressed in this release and documented below.

Providing `\ActivateGenericHook`

The hook management system offers a number of generic hooks, i.e., hooks whose names contain a variable component, for example the name of an environment. Predeclaring such hooks as not really feasible which is why these hooks use a different mechanism: they are implicitly available end spring into life the moment a package or the user in the preamble adds code to them using `\AddToHook`. The kernel offers such hooks for environments `env/...`, commands `cmd/...`, and files, package or classes, `file/...`, `include/...`, `package/...`, and `class/...`.

It is possible to offer generic hooks in packages, e.g., if you have hooks that depend on the current language and therefore need the language name as part of the hook name and you don't know all possible names beforehand.

If you want to offer your generic hooks you do this by using `\UseHook` or `\UseOneTimeHook` in your (package) code, but *without declaring the hook* with `\NewHook`. Without any further work a call to `\UseHook` with an undeclared hook name does nothing. So as an additional setup step, it is necessary to explicitly activate the generic hook with `\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/afterextras/`⟨*language*⟩ enabling the user to add language specific declarations there. They can then write

```
\ActivateGenericHook
          {babel/afterextras/ngerman}
\AddToHook{babel/afterextras/ngerman}
          {\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.

Clear extra hook code for next invocation

There are a few use cases where it would be 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 provided. *(github issue 565)*

Clean up after `\UseOneTimeHook`

Some hooks are meant to be used only once in a document, and any further attempt to add code to them causes 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 usages (e.g., the `begindocument` hook), but caused trouble if the one-time hook was intended, for example, as an initialization hook that is used once when a command is first called, but then ignored in further calls.

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

[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 because that is what it is meant for.

Class, package, and include hook improvements

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

```
\AddToHook{package/varioref/after}
  { ... apply my customizations if the package
      gets loaded (or was loaded already) ... }
```

without the need to first test if the package was already loaded before. *(github issue 623)*

Standardizing generic hook names

The initial set of generic hooks provided by the LaTeX 2_ε kernel had two patterns of hook 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 was the variable part supposed to go, and made the code more complicated and slower.

The file-related hooks have been renamed so that the variable part of their is in the middle, as with other hooks. The changes were:

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, and 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)*

Changed how `\RemoveFromHook` treats code that isn't in the hook

In the first version of `\RemoveFromHook`, in case the code label being removed didn't exist in the hook, a "removal order" would be queued, and the next time something tried to add that label to the hook, the `\AddToHook` would be cancelled by the removal order, and no code would be added that once. This was so that in principle package loading order wouldn't matter. However this implementation didn't work quite as intended, because while two `\AddToHook` to a given label would be removed by a single `\RemoveFromHook`, one `\RemoveFromHook`

could not cancel two \AddToHook to that label, and this asymmetry caused confusion and was a recipe for further problems.

The implementation was changed and now \RemoveFromHook only removes labels that already exist in a hook, and will display a warning if they don't. For usage across packages, for removing code in a hook, the voids relation should be used instead: this relation is non-destructive (meaning it can be later reverted with another one), and it is truly independent of package loading order, so it should be preferred.

(github issue 625)

???

(github issue 000)

New or improved commands

Added \PackageNote and \ClassNote

LaTeX offers \PackageError to signal errors that stop processing, \PackageWarning that generates a warning message on the terminal, but continues with the processing and also \PackageInfo to provide some information that is only written to the .log file. What hasn't existed up to now is a way to provide some information on the terminal that is identifying itself as coming from a specific package but which isn't claiming to be a warning. Thus, packages that wanted to write to the terminal used \PackageWarning even though the information wasn't really warning the user. For this we now have \PackageNote and \PackageNoteNoLine, that identify themselves as "informational", but still go to the terminal and not only to the transcript.

Similar commands exist for classes and there we have added the missing \ClassNote and \ClassNoteNoLine as well. *(github issue 613)*

New implementation for \counterwithin

New implementation for \counterwithout and \counterwithin with an additional optional arg so it becomes a drop-in replacement for amsmath \numberwithin.

write appropriate description

New default for \tracinglostchars

In 2021 the TeX engines got enhanced so that \tracinglostchars is now also supporting the value 3, turning missing characters into errors and not just warnings. This change made us realize that LaTeX should use a better default for this parameter (so far the warning was only written to the transcript file). Using the now available 3 would really be the best, but for compatibility reasons we only set it to 2 in the kernel. However, we recommend adding \tracinglostchars=3 to the preamble of documents, because missing glyphs in the output are an error and should therefore be properly looked at.

Provide tests for package and class loading

To test if a package was loaded you can now use \IfPackageLoadedTF {⟨*package*⟩} {⟨*true*⟩} {⟨*false*⟩} and based on the result execute different code. It is also possible to check if 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 at least one option in the ⟨*option-list*⟩ has not been used during loading or if the package hasn't been loaded at all. Both commands can be used anywhere in the document, i.e., they are not restricted to the preamble.[2]

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

New \ShowFloat command

The package fltrace offers a (fairly low-level but very detailed) way to trace LaTeX's float mechanism. This can help to understand 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 their contents 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.

Add ltcmd support for \NewCommandCopy and \ShowCommand

Since the 2020-10-01 release (see [4]), LaTeX provides \NewCommandCopy to copy robust commands, and \ShowCommand to show their definition in the terminal. In the same release, the xparse package was integrated in the kernel (now called ltcmd). However, the extended support for \NewCommandCopy and \ShowCommand was not implemented in ltcmd. The present LaTeX release ships with that support implemented, so now commands defined with xparse/ltcmd can be copied and their definition can be easily shown in the terminal without \csname gymnastics. *(github issue 569)*

???

(github issue 000)

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

Code improvements

Detect "\endfoo" when using \NewDocumentEnvironment

The `\newenvironment` command has always checked that both `\foo` and `\endfoo` do not exist before creating a `foo` environment. In contrast, `\NewDocumentEnvironment` has only to date checked for `\foo`; this reflects the fact that historically the code was designed around the target of an entirely new format. The behavior of `\NewDocumentEnvironment` now aligns with `\newenvironment`, except that it gives separate errors for the existence of `\foo` and `\endfoo`.

Improve "\begin ended by" error message

In the past it was possible to get an error message stating something like `\begin{foo} ended by \end{foo}`. This could happen when the environment name was partly hidden inside a macro, because the test was comparing the literal strings while the error message expanded the strings fully. This has now been changed to show a more sensible error message in this instance.

(github issue 587)

Additional Extended Latin characters predefined

Some additional characters such as ḱ (U+1E131) are now pre-defined and will work without needing `\DeclareUnicodeCharacter` declarations.

(github issue 593)

Use OpenType version of Latin Modern Upright Italic

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

Pick up all arguments to \contentsline

The `\contentsline` commands in the TOC file are always followed by four arguments, the last one being empty by default and only used by hyperref. The `\contentsline` command itself only used the first three arguments and relied on the fourth being empty (and thus doing no harm). But this assumption is not always correct, e.g., if you use hyperref and then remove it from the preamble.

So now we pick up all four arguments and save the last one away, so that it can be used by hyperref.

(github issue 633)

Fix dropping math lists in LuaTeX callbacks

The LuaTeX callbacks `pre_mlist_to_hlist_filter` and `post_mlist_to_hlist_filter` no longer create an error if the callback handler indicates to remove the entire math block. *(github issue 644)*

???

(github issue 000)

Bug fixes

Replicate argument processors for all embellishments

There was a bug in ltcmd (former 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, leading to unpredictable behavior. This bug has been fixed by applying the same argument processors to all the embellishments in a set, so 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 changes for \ij and \IJ

The ligatures "ij" and "IJ", as used in Dutch, are only available in most TeX fonts, if the commands `\ij` or `\IJ` are used or when you enter them as UTF-8 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 typesetting is done in `OT1` encoding).

(github issue 658)

Improve handling of legacy \bfdefault 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, i.e., with that declaration you can alter the default for individual meta font families, for example, by only altering the bold settings 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, it alters the setting for all meta families in one go. This cannot be done when the `\renewcommand` happens and was therefore delayed until the next time `\bfseries` or `\mdseries` was executed.

However, the problem with that approach was that any call to `\DeclareFontSeriesDefault` that happened in the mean time was overwritten—the two approaches didn't work well side by side. This is a problem because older font packages use the legacy method while newer ones use `\DeclareFontSeriesDefault`. This has now been corrected by changing `\DeclareFontSeriesDefault` to

do any necessary resetting prior to setting new defaults. *(github issue 663)*

Use of # in \textbf and similar commands

Previously you could not use the macro parameter character # in inline functions in the argument of `\texbf` and similar text font commands. An internal definition is now guarded with `\unexpanded` so that # does not generate an error. *(github issue 665)*

???

(github issue 000)

Changes to packages in the graphics category

Key for alt text

A new key `alt` has been added to `\includegraphics` allowing specification of an alternative text for accessibility. This is unused by default but may be used by extension packages and possible future use. *(github issue 651)*

Changes to packages in the tools category

varioref: Improve missing label handling

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, the data structure changes to five arguments, resulting in low-evel errors in some cases. The code has been changed to avoid these errors. *(https://tex.stackexchange.com/603948)*

array: Cancel \mathsurround for tabular

A `tabular` environment is internally typeset as an `array` environment with special settings and therefore in math mode. This math group should not get any `\mathsuround` added as it isn't a real formula because otherwise the spacing around the `tabular` changes. This bug has been there forever (which means not many people use `\mathsurround` or noticed the difference). Anyhow, this now got fixed. *(github issue 614)*

longtable: Improve behavior after a section heading

The `longtable` environment now sets the `\@nobreakfalse` flag so that spacing and indentation changes after a section heading are not triggered by a paragraph which follows a table which starts a section. Similarly a test for `\if@noskipsec` added so that a table follows a run-in heading rather than appearing before it. *(github issues 131 and 173)*

???

(github issue 000)

Changes to packages in the amsmath category

Improve compability with hyperref

When hyperref is used, incrementing a counter creates anchors and this can affect spacing. For a long time hyperref patched the `equation` environment to avoid some of the side effects. This patch has now been moved directly into amsmath. Additionally, a `\mathopen` has been added to avoid that the anchor affects a following unary symbol. *(github issue 652)*

???

(github issue 000)

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`