LATEX News

Issue 35, June 2022

Contents

Introduction

The 2022 June release of LATEX is again focussing on improvements made for our multi-year project to automatically offer tagged PDF output [1]. These are the new document metadata interface, the new mark mechanism for LATEX, a standard key/value approach for options, and the introduction of the **latex-lab** area for temporary code that can be optionally loaded by a document (when \DocumentMetadata is used with certain test keys). These additions are described in the first sections. Related to this effort there are updates to hyperref and tagpdf, both of which have their own distributions.

   As usual, we also added a number of smaller improvements and bug fixes in various components of core LATEX. Perhaps the most interesting ones (for some users) are direct support for floating point arithmetic (via \fpeval; see below) and the ability to properly color parts of math formulas without introducing spacing problems. For this we now offer the command \mathcolor; see the description near the end of the newsletter. There is also a new major release of the doc package that supports a more fine-grained classification of code elements and properly supports hyperref.

Document metadata interface

Until recently there was no dedicated location to declare settings that affect a document as a whole. Settings had to be placed somewhere in the preamble or as class options or sometimes even as package options. For some such settings this may cause issues, e.g., setting the PDF version is only possible as long as the PDF output file has not yet been opened which can be caused by loading one or the other package.

For the "LATEX Tagged PDF project" [1] further metadata about the whole document (and its processing) need to be specified and again this data should be all placed in a single well-defined place.

For this reason we introduce the new command `\DocumentMetadata` to unify all such settings in one place. The command expects a key/value list that describes all document metadata for the current document. It is only allowed to be used at the very beginning of the document, i.e., the declaration has to be placed *before* `\documentclass` and will issue an error if found later.

At this point in time we provide only the bare command in the format; the actual processing of the key/value is defined externally and the necessary code will be loaded if the command is used. This scheme is chosen for two reasons: by adding the command in the kernel it is available to everybody without the need to load a special package using `\RequirePackage`. The actual processing, though, is external so that we can easily extend the code (e.g., offering additional keys or changing the internal processing) while the above-mentioned project is progressing. Both together allows users to immediately benefit from intermediate results produced as part of the project, as well as offering the LATEX Project Team the flexibility to enable such intermediate results (for test purposes or even production use) in-between and independently of regular LATEX releases. Over time, tested and approved functionality can then seamlessly move into the kernel at a later stage without any alterations to documents already using it. At the same time, not using the new consolidated interface means that existing documents are in no way affected by the work that is carried out and is in a wider alpha or beta test phase.

Documentation about the new command and already existing keys are in `l3meta.pdf` and `documentmetadata-support.pdf` and also in the documentation of the `pdfmanagement-testphase` package.

Package and class authors can test if a user has used `\DocumentMetadata` with `\IfDocumentMetadataTF`.

The latex-lab bundle

We added a new latex-laboratory bundle in which we place new code that is going to be available only through a `\DocumentMetadata` declaration and that is—most importantly—work under development and subject to change without further notice. This means that commands and interfaces provided there may get altered or removed again after some public testing. The code can be accessed through the `\DocumentMetadata` key `testphase`. Currently supported values are `phase-I` and `phase-II` that enable code of the tagged PDF project (phase I is frozen, and phase II is the phase we

are currently working on). With

> `\DocumentMetadata{testphase=phase-II}`

you currently enable tagging for paragraphs and footnotes; more document elements will follow soon.

Eventually, code will move (once considered stable) from the testphase into the LATEX kernel itself. Tagging will continue to require a `\DocumentMetadata` declaration, but you will then be able to drop the `testphase` key setting.

A new mark mechanism for LATEX

The mark mechanism is TEX's way to pass information to the page-building process, which happens asynchronously, in order to communicate relevant data for running headers and footers to the latter, e.g., what is the first section on the page or the last subsection, etc. However, marks may also be used for other purposes. The new kernel module provides a generalized mechanism for marks of independent classes.

The TEX engines offer a low-level mark mechanism to communicate information about the content of the current page to the asynchronous operating output routine. It works by placing `\mark` commands into the source document.

This mechanism works well for simple formats (such as plain TEX) whose output routines are only called to generate pages. It fails, however, in LATEX (and other more complex formats), because here the output routine is sometimes called without producing a page, e.g., when encountering a float and placing it into one of the float regions. When that happens TEX's `\topmark` no longer reflects the situation at the top of the next page when that page is finally boxed.

Furthermore, TEX only offered a single mark while LATEX wanted to keep track of more than one piece of information. For that reason, LATEX implemented its own mark mechanism where the marks always contained two parts with their own interfaces: `\markboth` and `\markright` to set marks and `\leftmark` and `\rightmark` to retrieve them.

Unfortunately, this extended mechanism, while supporting scenarios such as chapter/section marks, was far from general. The mark situation at the top of a page (i.e., `\topmark`) remained unusable and the two marks offered were not really independent of each other because `\markboth` (as the name indicates) was always setting both.

The new mechanism now available in LATEX starting with the 2022 release overcomes both issues:

- It provides arbitrary many, fully independent named marks, that can be allocated and from that point onwards used.

- It offers access for each such mark to retrieve its top, first, and bottom value separately.

- Furthermore, the mechanism is augmented to give access to marks in different "regions", which may be other than full pages.

The legacy interfaces, e.g., `\markboth`, are kept. Thus classes and packages making use of them continue to work seamlessly. To make use of the extended possibility a new set of commands for the declaration of mark classes, setting their values and querying their state (in the output routine) is now available in addition. You find the documentation for the new interfaces together with examples and further notes on the mechanism in the file `ltmarks-doc.pdf`. Just call `texdoc ltmarks-doc` to display it on your computer.

A key/value approach to option handling

The classical LaTeX 2_ε method for handling options, using `\ProcessOptions`, treats each entry in the list as a string. Many package authors have sought to extend this handling by treating each entry as a key–value pair (keyval) instead. To date, this has required the use of additional packages, for example kvoptions.

The LaTeX team have for some time offered the package l3keys2e to allow keyvals defined using the L3 programming layer module l3keys to act as package options. This ability has now been integrated directly into the kernel. As part of this integration, the syntax for processing keyval options has been refined, such that

`\ProcessKeyOptions`

will now automatically pick up the package name as the key *family*, unless explicitly given as an optional argument:

`\ProcessKeyOptions[family]`

To support creating key options for this mechanism, the new command `\DeclareKeys` has been added. This works using the same general approach as l3keys or pgfkeys: each key has one or more *properties* which define its behavior.

Options for packages which use this new approach will not be checked for clashes by the kernel. Instead, each time a `\usepackage` or `\RequirePackage` line is encountered, the list of options given will be passed to `\ProcessKeyOptions`. Options which can only be given the first time a package is loaded can be marked using the property `.usage = load`, and will result in a warning if used in a subsequent package loading line.

Package options defined in this way can also be set within a package using the new command `\SetKeys`, which again takes an optional argument to specify the *family*, plus a mandatory one for the options themselves.

New or improved commands

Floating point and integer calculations

The L3 programming layer offers expandable commands for calculating floating point and integer values, but so far these functions have only been available to programmers, because they require `\ExplSyntaxOn` to be in force. To make them easily available at the document level, the small package xfp defined `\fpeval` and `\inteval`.

An example of use could be the following:

```
\LaTeX{} can now compute:
\[ \frac{\sin (3.5)}{2} + 2\cdot 10^{-3}
    = \fpeval{sin(3.5)/2 + 2e-3}        \]
```

which produces the following output:

LaTeX can now compute:

$$\frac{\sin(3.5)}{2} + 2 \cdot 10^{-3} = -0.1733916138448099$$

These two commands have now been moved into the kernel and in addition we also provide `\dimeval` and `\skipeval`. The details of their syntax are described in `usrguide3.pdf`. The command `\fpeval` offers a rich syntax allowing for extensive calculations, whereas the other three commands are essentially thin wrappers for `\numexpr`, `\dimexpr`, and `\glueexpr`—therefore inheriting some syntax peculiarities and limitations in expressiveness.

```
\newcommand\calculateheight[1]{%
  \setlength\textheight{\dimeval{\topskip
      + \baselineskip * \inteval{#1-1}}}}
```

The above, for example, calculates the appropriate `\textheight` for a given number of text lines.

(github issue 711)

CamelCase commands for changing arguments to csnames

It is sometimes helpful to "construct" a command name on the fly rather than providing it as a single `\...` token. For these kinds of tasks the LaTeX3 programming layer offers a general mechanism (in the form of `\exp_args:N...` and `\cs_generate_variant:Nn`). However, when declaring new document-level commands with `\NewDocumentCommand` or `\NewCommandCopy`, etc. the L3 programming layer may not be active, and even if it is, mixing CamelCase syntax with L3 programming syntax is not really a good approach. We have therefore added the commands `\UseName` and `\ExpandArgs` to assist in such situations, e.g.,

```
\NewDocumentCommand\newcopyedit{mO{red}}
  {\newcounter{todo#1}%
   \ExpandArgs{c}\NewDocumentCommand{#1}{s m}%
     {\stepcounter{todo#1}%
      \IfBooleanTF {##1}%
```

```
        {\todo[color=#2!10]%
            {\UseName{thetodo#1}: ##2}}%
        {\todo[inline,color=#2!10]%
            {\UseName{thetodo#1}: ##2}}%
      }%
  }
```

which provides a declaration mechanism for copyedit commands, so that `\newcopyedit{FMi}[blue]` then defines `\FMi` (and the necessary counter).

The command `\ExpandArgs` can be useful with the argument `cc` or `Nc` in combination with `\NewCommandCopy` if the old or new command name or both need constructing. Finally, there is `\UseName` which takes its argument and turns it into a command (i.e., a CamelCase version of `\@nameuse` (LaTeX 2ε) or `\use:c` (L3 programming layer)) which was also used in the example above.

(github issue 735)

Testing for (nearly) empty arguments

In addition to `\IfNoValueTF` to test if an optional argument was provided or not, there is now also `\IfBlankTF`, which tests if the argument is empty or contains only blanks. Based on the result it selects a true or false code branch. As usual, the variants `\IfBlankT` and `\IfBlankF` are also provided for use when only one branch leads to some action. Further details and examples are given in `usrguide3.pdf`.

This test can also be useful if you set up key/value options and want to test if a key was specified without giving a value, vs. specifying "*key = ,*".

Better allocator for Lua command ids

In LuaTeX we already had the `\newluafunction` macro which allocates a Lua function identifier which can be used to define commands with `\luadef`. But this always required two steps: `\newluafunction` defines the passed control sequence as an integer, which then has to be used to define the actual Lua command with `\luadef`. After that, the integer is no longer needed. This was inconsistent with other allocators. Therefore we added two new allocators `\newluacmd` and `\newexpandableluacmd` which directly define a control sequences invoking the allocated Lua function. The former defines a non-expandable Lua command, the latter an expandable one. Of course, the associated Lua function still has to be defined by assigning a function to the `lua.get_functions_table()` table. The required index is available in `\allocationnumber`.

An example could be

```
\newluacmd \greeting
\directlua {
lua.get_functions_table()
   [tex.count.allocationnumber]
  = function()
```

```
    local name = token.scan_argument()
    tex.sprint('Hello ', name, '!')
  end
}

\greeting{world}
```

(github issue 536)

Starred command version for `\ref`, `\Ref` and `\pageref`

For a long time hyperref has provided starred versions for the reference commands which do not create active links. This syntax extension required users and package authors to check if hyperref was loaded and adjust the coding accordingly or take the starred forms out if text was copied to a document without hyperref. The commands have now been aligned with the hyperref usage and always allow an optional star. The showkeys package has been updated to handle the starred versions too, both with hyperref or nameref and without. The commands are defined with `\NewDocumentCommand` and so no longer expand when written to auxiliary files. This reduces the number of compilations needed to resolve references in captions and sectioning commands. The package ifthen has been updated to ensure that `\pageref` can still be used inside tests like `\isodd`.

Preparation for supporting PDF in backends

At the current point in time, basic support for PDF in backends is not part of LaTeX core; it is provided by an external package like hyperref. At some time in the future that work will be placed into the kernel but for now it is separate and has to be explicitly loaded in the document. To enable class and package authors to support PDF-specific tasks like the creation of link targets without having to test first if hyperref has been loaded, dummy versions of the commands `\MakeLinkTarget`, `\LinkTargetOn`, `\LinkTargetOff` and `\NextLinkTarget` are provided.

Code improvements

`\protected` UTF-8 character definitions

The characters defined via `utf8.def` are now defined as `\protected` macros. This makes them safe to use in expansion contexts where the classic `\protect` mechanism is not enabled, notably L3 programming layer `e` and `x` arguments.

Related to this change `\MakeUppercase` and `\MakeLowercase` have been updated to use the Unicode-aware case changing functions `\text_lowercase:n` in place of the TeX primitive `\lowercase`. The `\NoCaseChange` command from the textcase package has also been added.

Note: for technical reasons these low-level character handling changes will not be rolled back if the format

version is rolled back using the latexrelease package rollback mechanism. *(github issue 780)*

A small update to \obeylines and \obeyspaces

The plain TEX versions of \obeylines and \obeyspaces make ^^M and ␣ active and force them to execute \par and \space, respectively. Don Knuth makes a remark in the TEXbook that one can then use a trick such as

```
\let\par=\cr \obeylines \halign{...
```

However, redefining \par like this may lead to all kinds of problems in LATEX. We have therefore changed the commands to use an indirection: the active characters now execute \obeyedline and \obeyedspace, which in turn do what the hardwired solution did before.

> This • means • that • it • is • now • possible • to • achieve • special • effects • in • a • safe • way. • This • paragraph, • for • example, • was • produced • by • making • \obeyedspace • generate • {␣\textbullet␣} • and • enabling • \obeyspaces • within • a • quote • environment.

Thus, if you are keen to use the plain TEX trick, you need to say \let\obeyedlines=\cr now. *(github issue 367)*

doc upgraded to version 3

After roughly three decades the doc package received a cautious uplift, as already announced at the 2019 TUG conference—changes to doc are obviously always done in a leisurely manner.

Given that most documentation is nowadays viewed on screen, hyperref support is added and by default enabled (suppress it with option nohyperref or alternatively with hyperref=false) so the internal cross-references are properly resolved including those from the index back into the document.

Furthermore, doc now has a general mechanism to define additional "doc" elements besides the two Macro and Env it has known in the past. This enables better documentation because you can now clearly mark different types of objects instead of simply calling them all "macros". If desired, they can be collected together under a heading in the index so that you have a section just with your document interface commands, or with all parameters, or . . .

The code borrows ideas from Didier Verna's dox package (although the document level interface is different) and it makes use of Heiko Oberdiek's hypdoc package, which at some point in the future will be completely integrated, given that its whole purpose it to patch doc's internal commands to make them hyperref-aware.

All changes are expected to be upward compatible, but if you run into issues with older documentation using doc a simple and quick solution is to load the package as follows: \usepackage{doc}[=v2]

doc can now show dates in change log

Up to now the change log was always sorted by version numbers (ignoring the date that was given in the \changes command). It can now be sorted by both version and date if you specify the option reportchangedates on package level and in that case the changes are displayed with

$$\langle version \rangle - \langle date \rangle$$

as the heading (instead of just $\langle version \rangle$), when using \PrintChanges. *(github issue 531)*

ltxdoc gets options nocfg and doc2

The LATEX sources are formatted with the ltxdoc class, which supports loading a local config file ltxdoc.cfg. In the past the LATEX sources used such a file but it was not distributed. As a result reprocessing the LATEX sources elsewhere showed formatting changes. We now distribute this file which means that it is loaded by default. With the option nocfg this can be prevented.

We also added a doc2 option to the class so that it is possible to run old documentation with doc version 2, if necessary.

LuaTEX callback improvements

The LuaTEX callbacks hpack_quality and vpack_quality are now exclusive and therefore only allow a single handler. The previous type list resulted in incorrect parameters when multiple handlers were set; therefore, this only makes an existing restriction more explicit.

Additionally the return value true for list callbacks is now handled internally and no longer passed on to the engine. This simplifies the handling of these callbacks and makes it easier to provide consistent interfaces for user-defined list callbacks.

Class proc supports twoside

The document class proc, which is a small variation on the article class, now supports the twoside option, displaying different data in the footer line on recto and verso pages. *(github issue 704)*

Croatian character support

The default inputenc support has been extended to support the 9 characters DŽ, Dž, dž, LJ, Lj, lj, NJ, Nj, nj, input as single UTF-8 code points in the range U+01C4 to U+01CC. *(github issue 723)*

Cleanup of the Unicode declaration interface

When declaring encoding specific commands for the Unicode (TU) encoding some declarations (e.g., \DeclareUnicodeComposite) do not have an explicit argument for the encoding name, but instead use the command \UnicodeEncodingName internally. There was one exception though: \DeclareUnicodeAccent

required an explicit encoding argument. This inconsistency has now been removed and the encoding name is always implicit. To avoid a breaking change for a few packages on CTAN, `\DeclareUnicodeAccent` still accepts three arguments if the second argument is `TU` or `\UnicodeEncodingName`. Once all packages have been updated this code branch will get removed.

At the same time we added `\DeclareUnicodeCommand` and `\DeclareUnicodeSymbol` for consistency. They also use `\UnicodeEncodingName` internally, instead of requiring an encoding argument as their general purpose counterparts do. *(github issue 253)*

New hook: `include/excluded`

A few releases ago we introduced a number of file hooks for different types of files; see [2] and in particular [4]. The hooks for `\include` files now have an addition: if such a file is not included (because `\includeonly` is used and its ⟨*name*⟩ is not listed in the argument) then the hooks `include/excluded` and `include/`⟨*name*⟩`/excluded` are executed in that order—of course, only if they contain code. This happens after LaTeX has loaded the `.aux` file for this include file, i.e., after LaTeX has updated its counters to pretend that the file was seen.

Input support for normalized angle brackets

Source files containing ⟨ or ⟩ directly written as Unicode codepoints U+2329 and U+232A no longer break when the source file gets normalized under Unicode normalization rules. *(github issue gh/714)*

Bug fixes

Using `\DeclareUnicodeCharacter` with C1 control points

An error in the UTF-8 handling for non-Unicode TeX has prevented `\DeclareUnicodeCharacter` being used with characters in the range hex 80 to 9F. This has been corrected in this release. *(github issue 730)*

Fix `\ShowCommand` when used with ltcmd

When `\ShowCommand` support was added for ltcmd in the previous release [3], a blunder in the code made it so that when `\ShowCommand` was used on a command defined with ltcmd, it only printed the meaning of the command in the terminal, but didn't stop for interaction as it does elsewhere (mimicking `\show`). The issue is now fixed. *(github issue 739)*

Make `\cite{}` produce a warning

When the `\cite` command can't resolve a citation label it issues a warning "Citation '⟨*label*⟩' on page ⟨*page*⟩ undefined". However, due to some implementation details a completely empty argument was always silently accepted. Given that there are probably people who write `\cite{}` with the intention to fill in the

correct label later it is rather unfortunate if that is not generating a warning that something in the document is still amiss. This has finally been corrected and a warning is now generated also in this case. *(github issue 790)*

Fix adding cmd hooks to simple macros

A bug in how LaTeX detected the type of a command caused a premature forced expansion of such commands, which, depending on their definition, could be harmless or could cause severe trouble. This has been fixed in the latest release. *(github issue 795)*
(https://tex.stackexchange.com/q/637565)

Warn if `shipout/lastpage` hook is executed too early

The hook `shipout/lastpage` is intended to place `\special`s into the last page shipped out. This is needed for some use cases, e.g., tagging. If that hook is nonempty and the user has added additional pages since the last run, then LaTeX executes this hook too early, but until now without giving any indication that the document needs rerunning. This has now been corrected and an appropriate warning is given. *(github issue 813)*

More consistent use of cramped math styles in LuaTeX

Using LuaTeX's `\Udelimiterover` to place a horizontally extensible glyph on top of a mathematical expression now causes the expression to be set in cramped style, as used in similar situations by traditional TeX math rendering. Similarly, cramped style is now used for expressions set under such a delimiter using `\Uunderdelimiter`, but is no longer used when setting an expression on top of such extensible glyphs using `\Uoverdelimiter`. This new behavior follows TeX's rule that cramped style is used whenever something else appears above the expression. Additionally the math style of these constructs can now be detected using `\mathstyle`.

The old behavior can be restored by adding

    \mathdefaultsmode=0

to a document.

Fixed bug when setting hook rules for one-time hooks

If a `\DeclareHookRule` command is set for a one-time hook, it has to come *before* the hook gets used, because otherwise it never applies—after all, the hook is used only once. There was a bug in the implementation in that the sorting mechanism was still applied if the `\DeclareHookRule` declaration appeared while the one-time hook was executed, causing the spurious typesetting of the code labels and the hook name. This bug is now fixed and an error is raised when a new sorting rule is added to an already-used one-time hook.

A possible scenario in which this new error is raised is the following: package AAA declares a hook rule for `begindocument` (i.e., `\AtBeginDocument`) to sort out the behavior between itself and some other package.

Package BBB wants to load package AAA but only if it hasn't been loaded in the preamble, so delays the loading to `begindocument`. In that case the hook rule declared by AAA can no longer be applied and you get the error. If that happens the solution is to load the package in `begindocument/before`, which is executed at the very end of the preamble but before `begindocument` is processed. *(github issue 818)*

Changes to packages in the amsmath category

amsopn: Do not reset `\operator@font`
The package amsopn used to define `\operator@font` but this command has been provided by the LaTeX format for at least 14 years. As a result the definition in amsopn is equivalent to a reset to the kernel definition, which is unnecessary and surprising if you alter the math setup (e.g., by loading a package) and at a later stage add amsmath, which then undoes part of your setup. For this reason the definition was taken out and amsmath/amsopn now relies on the format definition.

In the unlikely event that you want the resetting to happen, use

```
\makeatletter
 \def\operator@font{\mathgroup\symoperators}
\makeatother
```

after loading the package. *(github issue 734)*

amsmath: Error in `\shoveleft`
If `\shoveleft` started out with the words "plus" or "minus" it was misunderstood as part of a rubber length and led either to an error or was swallowed without trace. By adding a `\relax` this erroneous scanning into the argument of `\shoveleft` is now prevented. *(github issue 714)*

amsmath and amsopn: Robustify user commands
Most user-level commands have been made robust in the LaTeX kernel during the last years, but variant definitions in amsmath turned them back into fragile beings. We have now made most commands in amsmath and amsopn robust as well to match the kernel behavior. This also resolves a bug recently discovered in the mathtools package, which was due to `\big` not being robust after amsmath was loaded. *(github issue 123)*

Changes to packages in the graphics category

Color in formulas
While it is possible to color parts of a formula using `\color` commands the approach is fairly cumbersome. For example, to color a summation sign, but not its limits, you need four `\color` commands and some seemingly unnecessary sets of braces to get coloring and spacing right:

```
\[ X = \color{red} \sum
% without {{ the superscript below is misplaced
                _{{\color{black} i=1}}
% without {{ the \sum is black
                ^{{\color{black} n}}
  \color{black}    % without it the x_i is red
  x_i           \]
```

Leaving out any of the `\color` commands or any of the `{{...}}` will give you a wrong result instead of the desired

$$X = \sum_{i=1}^{n} x_i$$

So even if this is possible, it is not a very practical solution and furthermore there are a number of cases where it is impossible to color a certain part of a formula, for example, an opening symbol such as `\left(` but not the corresponding `\right)`.

We have therefore added the command `\mathcolor` to the color and xcolor package, which has the same syntax as `\textcolor`, but is specially designed for use in math and handles sub and superscripts and other aspects correctly and preserves correct spacing. Thus, the above example can now be written as

```
\[ X = \mathcolor{red}{\sum}_{i=1}^n x_i \]
```

This command is *only* allowed in formulas. For details and further examples, see `mathcolor.pdf`.

Fix locating files with `\graphicspath`
If a call to `\includegraphics` asked for a file (say, `image`) without extension, and if both `A/image.pdf` and `B/image.tex` existed (both `A/` and `B/` in `\graphicspath`, but neither in a folder searched by TeX), then `A/image.pdf` would not be found, and a "file not found" error would be incorrectly thrown. The issue is now fixed and the graphics file is correctly found. *(github issue 776)*
(https://tex.stackexchange.com/q/630167)

Changes to packages in the tools category

multicol: Fix `\newcolumn`
The recently added `\newcolumn` didn't work properly if used in vertical mode, where it behaved like `\columnbreak`, i.e., spreading the column material out instead of running the column short. *(https://tex.stackexchange.com/q/624940)*

bm: Fix for amsmath operators
An internal command used in the definition of operator commands such as `\sin` in amsmath has been guarded in `\bm` to prevent internal syntax errors due to premature expansion. *(github issue 744)*

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

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

[4] Frank Mittelbach, Phelype Oleinik,
LaTeX Project Team: *The `ltfilehook` documentation.*
Run `texdoc ltfilehook-doc` to view.