LaTeX News

Issue 32, October 2020

Contents

Introduction

The 2020-10-01 release of LaTeX shows that work on improving LaTeX has again intensified. The two most important new features are the kernel support for xparse and the introduction of the new hook management system for LaTeX, but as you can see there are many more smaller enhancements and bug fixes added to the kernel and various packages.

Providing xparse in the format

The official interface in the LaTeX 2ε kernel for creating document-level commands has always been \newcommand. This was a big step forward from LaTeX 2.09. However, it was still very limited in the types of command it can create: those taking at most one optional argument in square brackets, then zero or more mandatory arguments. Richer syntaxes required use of the TeX \def primitive along with appropriate low-level macro programming.

The LaTeX team started work on a comprehensive document-command parser, xparse, in the late 1990s. In the past decade, the experimental ideas it provides have been carefully worked through and moved to a stable footing. As such, xparse is now used to define a very large number of document and package commands. It does this by providing a rich and self-consistent

syntax to describe a wide range interfaces seen in LaTeX packages.

The ideas developed in xparse are now sufficiently well-tested that the majority can be transferred into the LaTeX kernel. Thus the following commands have been added

- `\NewDocumentCommand, \RenewDocumentCommand,`
  `\ProvideDocumentCommand,`
  `\DeclareDocumentCommand`

- `\NewExpandableDocumentCommand,`
  `\RenewExpandableDocumentCommand,`
  `\ProvideExpandableDocumentCommand,`
  `\DeclareExpandableDocumentCommand`

- `\NewDocumentEnvironment,`
  `\RenewDocumentEnvironment,`
  `\ProvideDocumentEnvironment,`
  `\DeclareDocumentEnvironment`

- `\BooleanTrue \BooleanFalse`

- `\IfBooleanTF, \IfBooleanT, \IfBooleanF`

- `\IfNoValueTF, \IfNoValueT, \IfNoValueF`

- `\IfValueTF, \IfValueT, \IfValueF`

- `\SplitArgument, \SplitList, \TrimSpaces,`
  `\ProcessList, \ReverseBoolean`

- `\GetDocumentCommandArgSpec`
  `\GetDocumentEnvironmentArgSpec`

Most, but not all, of the argument types defined by xparse are now supported at the kernel level. In particular, the types g/G, l and u are *not* provided by the kernel code; these are deprecated but still available by explicitly loading xparse. All other argument types *are* now available directly within the LaTeX 2ε kernel.

A hook management system for LaTeX

With the fall 2020 release of LaTeX we provide a general hook management system for the kernel and for packages. This will allow packages to safely add code to various kernel and package hooks and if necessary define rules to reorder the code in the hooks to resolve typical package loading order issues. This hook system is written in the L3 programming layer and thus forms the first larger application within the kernel that makes use of the LaTeX3 functionality now available (if we discount xparse which was already available for a long time as a separate package).

The file `lthooks.dtx` holds the core management code for hooks and defines basic hooks for environments (as previously offered by etoolbox), `ltshipout.dtx` provides kernel hooks into the shipout process (making packages like atbegshi, etc., unnecessary) and the file `ltfilehook.dtx` holds redefinition for commands like `\input` or `\usepackage` so that they offer hooks in

a similar fashion to what is provided by the filehook package.

At the moment the integration is lightweight, overwriting definitions made earlier during format generation (though this will change after more thorough testing). For that reason the documention isn't in its final form either and you have to read through three different documents:

lthooks-doc.pdf Core management interface and basic hooks for environments provided by the kernel.

ltshipout-doc.pdf Hooks accessible while a page is being shipped out.

ltfilehook-doc.pdf Hooks used when reading a file.

For those who wish to also study the code, replace `-doc` with `-code`, e.g., `lthooks-code.pdf`. All documents should be accessible via `texdoc`, e.g.,

    texdoc lthooks-doc

should open the core documentation for you.

Other changes to the LaTeX kernel

\symbol in math mode for large Unicode values
The LaTeX 2ε kernel defines the command `\symbol`, which allows characters to be typeset by entering their 'slot number'. With the LuaTeX and XeTeX engines, these slot numbers can extend to very large values to accommodate Unicode characters in the upper Unicode planes (e.g., bold mathematical capital A is slot number `"1D400` in hex or `119808` in decimal). The XeTeX engine did not allow `\symbol` in math mode for values above 2^{16}, and this limitation has now been lifted.

(github issue 124)

Correct Unicode value of \=y (ȳ)
The Unicode slot for ȳ was incorrectly pointing to the slot for Ȳ. This has been corrected. *(github issue 326)*

Add support for Unicode soft hyphens
For a long time, the UTF-8 option for inputenc made the Unicode soft hyphen character (U+00AD) an alias for the LaTeX soft hyphen `\-`. The Unicode engines XeTeX and LuaTeX behaved different though: They either ignored U+00AD or interpreted it as an unconditional hyphen. This inconsistency is fixed now and LaTeX always treats U+00AD as `\-`. *(github issue 323)*

Fix capital accents in Unicode engines
In Unicode engines the capital accents such as `\capitalcedilla`, etc., have been implemented as trival shorthands for the normal accents (because other than Computer Modern virtually no fonts support them), but that failed when hyperref got loaded. This has been corrected. *(github issue 332)*

Support calc in various kernel commands

The `\hspace`, `\vspace`, `\addvspace`, `\textbackslash` and other commands simply passed their argument to a TeX primitive to produce the necessary space. As a result it was impossible to specify anything other than a simple dimension value in such arguments. This has been changed, so that now `calc` syntax is also supported with these commands. *(github issue 152)*

Support ε-TeX length expressions in `picture` coordinates

Picture mode coordinates specified with `(_,_)` previously accepted multiples of `\unitlength`. They now also allow ε-TeX length expressions (as used by the `\glueexpr` primitive although all uses in `picture` mode are non stretchy).

So valid uses include `\put(2,2)` as previously, but now also uses such as `\put(\textwidth-5cm,0.4\texteight)`.

Note that you can only use expressions with lengths, `\put(1+2,0)` is not supported.

Spaces in filenames of included files

File names containing spaces lead to unexpected results when used in the commands `\include` and `\includeonly`. This has now been fixed and the argument to `\include` can contain file name containing spaces. Leading or trailing spaces will be stripped off but spaces within the file name are kept. The argument to `\includeonly`, which is a comma-separated list of files to process, can also contain spaces with any leading and trailing spaces stripped from the individual filenames while the spaces *in* the file names will remain intact. *(github issues 217 and 218)*

Avoid extra line in `\centering`, `\raggedleft` or `\raggedright`

If we aren't justifying paragraphs then a very long word (longer than a line) could result in an unnecessary extra line in order to prevent a hyphen in the second-last line of the paragraph. This is now avoided by setting `\finalhyphendemerits` to zero in unjustified settings. *(github issue 274)*

Set a non-zero `\baselineskip` in text scripts

As `\textsuperscript` and `\textsubscript` usually contain only a few characters on a single line the `\baselineskip` was set to zero. However, `hyperref` uses that value to determine the height of a link box which consequently came out far too small. This has been adjusted. *(github issue 249)*

Spacing issues when using `\linethickness`

In some circumstances the use of `\linethickness` introduced a spurious space that shifted objects in a `picture` environment to the right. This has been corrected. *(github issue 274)*

Better support for the legacy series default interface

In the initial implementation of LaTeX's font selection scheme (NFSS) changes to any default where carried out by redefining some commands, e.g., `\seriesdefault`. In 2019 we introduced various extensions and with it new methods of customising certain parts of NFSS, e.g., the recommended way for changing the series default(s) is now through `\DeclareFontSeriesDefault` [1]. In this release we improved the support for legacy documents using the old method to cover additional edge cases. *(github issues 306,315)*

Support for uncommon font series defaults

If a font family was set up with fairly unusual font series defaults, e.g.,

```
\renewcommand\ttdefault{lmvtt}
\DeclareFontSeriesDefault[tt]{md}{lm}
\DeclareFontSeriesDefault[tt]{bf}{bm}
```

then a switch between the main document families, e.g., `\ttfamily...\rmfamily` did not always correctly continue typesetting in medium or bold series if that involved adjusting the values used by `\mdseries` or `\bfseries`. This has now been corrected. *(github issue 291)*

Checking the current font series context

Sometimes it is necessary to define commands that act differently when used in bold context (e.g., inside `\textbf`. Now that it is possible in LaTeX to specify different "bf" defaults based for each of the three meta families (`rm`, `sf` and `tt`) via `\DeclareFontSeriesDefault`, it is not any longer easy to answer the question "am I typesetting in a bold context?". To help with this problem a new command was provided:

$$\texttt{\textbackslash IfFontSeriesContextTF}\{\langle context\rangle\}$$
$$\{\langle true\ code\rangle\}\{\langle false\ code\rangle\}$$

The $\langle context\rangle$ can be either `bf` (bold) or `md` (medium) and depending on whether or not the current font is recognized as being selected through `\bfseries` or `\mdseries` the $\langle true\ code\rangle$ or $\langle false\ code\rangle$ is executed. As an example

```
\usepackage{bm}  % (bold math)
\newcommand\vbeta{\IfFontSeriesContextTF{bf}%
          {\ensuremath{\bm{\beta}}}%
          {\ensuremath{\beta}}}
```

This way you can write `\vbeta-isotopes` and if used in a heading it comes out in a bolder version. *(github issue 336)*

Avoid spurious package option warning

When a package is loaded with a number of options, say `X`, `Y` and `Z`, and then later another loading attempt was made with a subset of the options or no options, it

was possible that you got an error message that option X is not known to the package. This obviously incorrect error was due to some timing issue where the list of available options got lost prematurely. This has now been fixed. *(github issue 22)*

Adjusting `fleqn`

In amsmath the `\mathindent` parameter used with the `fleqn` design is a rubber length paramter allowing for setting it to a value such as `1em minus 1em`, i.e., so the normal indentation can be reduced in case of very wide math displays. This is now also supported by the LaTeX standard classes.

In addition a compressible space between formula and equation number in the `equation` environment got added when the `fleqn` option is used so that a very wide formula doesn't bump into the equation number.
(github issue 252)

Provide `\clap`

LaTeX has inherited `\llap` and `\rlap` from plain TeX (zero-sized boxes whose content sticks out to the left or right, respectively) but there isn't a corresponding `\clap` command that centers the material. This missing command was added by several addon packages, e.g., mathtools, and has now been added to the kernel.

Fix to legacy math alphabet interface

When using the LaTeX 2.09 legacy math alphabet interface, e.g., `$\sf -1$` instead of `$\mathsf{-1}$`, an extra math Ord atom was added to the formula in case the math alphabet was used for the first time. In some cases this math atom would change the spacing, e.g., change the unary minus sign into a binary minus in the above example. This has finally been fixed.
(gnats issue latex/3357)

Added tests for format, package and class dates

To implement compatibility code or to ensure that certain features are available it is helpful and often necessary to check the date of the format or that of a package or class and execute different code based on the result. For that LaTeX only had some internal commands (`\@ifpackagelater` and `\@ifclasslater`) for testing package or class names but nothing really for testing the format date. For the latter one had to resort to some obscure command `\@ifl@t@r` that, given its cryptic name, was clearly never intended for use even in package or class code. Furthermore, even the existing interface commands were defective as they are testing for "equal or later" and not for "later" as their names indicate.

We have therefore introduced three new CamelCase commands as the official interface for such tests

> `\IfFormatAtLeastTF{`⟨*date*⟩`}`
> `        {`⟨*true code*⟩`}{`⟨*false code*⟩`}`

and for package and class tests

> `\IfClassAtLeastTF{`⟨*class name*⟩`}{`⟨*date*⟩`}`
> `        {`⟨*true code*⟩`}{`⟨*false code*⟩`}`
> `\IfPackageAtLeastTF{`⟨*package name*⟩`}{`⟨*date*⟩`}`
> `        {`⟨*true code*⟩`}{`⟨*false code*⟩`}`

For compatibility reasons the legacy commands remain available, but we suggest to replace them over time and use the new interfaces in new code. *(github issue 186)*

Avoid a problem with `\verb`

If a user typed `\verb␣!~!␣foo` instead of `\verb!~!␣foo` by mistake, then surprisingly the result was "`!~!foo`" without any warning or error. What happened was that the ␣ became the argument delimiter due to the rather complex processing done by `\verb` to render verbatim. This now got fixed and spaces directly following the command `\verb` or `\verb*` are ignored as elsewhere.
(github issue 327)

Provide a way to copy robust commands...

With the previous LaTeX 2ε release, several user-level commands were made robust, so the need for a way to create copies of these commands (often to redefine them) increased, and the LaTeX 2ε kernel didn't have a way to do so. Previously this functionality was provided in part by Heiko Oberdiek's letltxmacro package, which allows a robust command `\foo` to be copied to `\bar` with `\LetLtxMacro\bar\foo`.

From this release onwards, the LaTeX 2ε kernel provides `\NewCommandCopy` (and `\Renew...` and `\Declare...` variants) which functions almost like `\LetLtxMacro`. To the end user, both should work the same way, and one shouldn't need to worry about the definition of the command: `\NewCommandCopy` should do the hard work.

`\NewCommandCopy` knows about the different types of definitions from the LaTeX 2ε kernel, and also from other packages, such as xparse's command declarations like `\NewDocumentCommand`, and etoolbox's `\newrobustcmd`, and it can be extended to cover further packages.
(github issue 239)

... and a way to `\show` them

It is sometimes necessary to look the definition of a command, and often one not only doesn't know where that command is defined, but doesn't know if it gets redefined by some package, so often enough looking at the source doesn't help. The typical way around this problem is to use TeX's `\show` primitive to look at the definition of a command, which works fine until the command being `\show`n is robust. With `\show\frac` one sees

```
> \frac=macro:
->\protect \frac  .
```

which is not very helpful. To show the actual command the user needed to notice that the real definition of \frac is in the \frac␣ macro and do \expandafter\show\csname frac\space\endcsname.

But with the machinery for copying robust commands in place it is already possible to examine a command and detect (as far as a macro expansion language allows) how it was defined. \ShowCommand knows that and with \ShowCommand\frac the terminal will show

```
> \frac=robust macro:
->\protect \frac  .

> \frac =\long macro:
#1#2->{\begingroup #1\endgroup \over #2}.
```

(github issue 373)

Merge l3docstrip into docstrip
The file l3docstrip.tex offered a small extension over the original docstrip.tex file supporting the %<@@=⟨*module*⟩> syntax of expl3. This has been merged into docstrip so that it can now be used for both traditional .dtx files and those containing code written in the L3 programming layer language. *(github issue 337)*

Support vertical typesetting with doc
The macrocode environment uses a trivlist internally and as part of this sets up the \@labels box to contain some horizontal skips, but that box is never used. As a result this generates an issue in some circumstances if the typesetting direction is vertical. This has now been corrected to support such use cases as well. *(github issue 344)*

Record the counter name stepped by \refstepcounter
\refstepcounter now stores the name of the counter in \@currentcounter. This allows packages like zref and hyperref to store the name without having to patch \refstepcounter. *(github issue 300)*

Native LuaTEX behaviour for \-
LATEX changes \- to add a discretionary hyphen even if \hyphenchar is set to −1. This change is not necessary under LuaTEX because in there \- is not affected by \hyphenchar in the first place. Therefore this behaviour has been changed to ensure that LuaTEX's (language specific) hyphenation characters are respected by \-.

Allow \par commands inside \typeout
\typeout used to choke when seeing an empty line or a \par command in its argument. However, sometimes it is used to display arbitrary user input or code (wrapped, for example, in \unexpanded) which may contain explicit \par commands. This is now allowed. *(github issue 335)*

Spacing commands moved from amsmath to the kernel
Originally LATEX only provided a small set of spacing commands for use in text and math, some of the commands like \; were only supported in math mode. amsmath normalized and provided all of them in text and math. This code has now been moved to the kernel so that it is generally available. *(github issue 303)*

command name(s)	math	text
\, \thinspace	$x\,x$	x x
\! \negthinspace	$x\!x$	xx
\: \> \medspace	$x\:x$	x x
\negmedspace	$x\!x$	xx
\; \thickspace	$x\;x$	x x
\negthickspace	$x\!x$	xx

Access raw glyphs in LuaTEX without reloading fonts
LATEX's definitions for \textquotesingle, \textasciigrave, and \textquotedbl for the TU encoding in LuaTEX need special handling to stop the shaper from replacing these characters with curly quotes. This used to be done by reloading the current font without the tlig feature, but that came with multiple disadvantages: It behaves differently than the corresponding XƎTEX code and it is not very efficient. This code has now been replaced with an implementation which injects a protected glyph node which is not affected by font shaping. *(github issue 165)*

Added a fourth empty argument to \contentsline
LATEX's writes with \addcontentsline a \contentsline command with three arguments to the toc and similar files. hyperref redefines \addcontentsline to write a fourth argument. The change unifies the number of arguments by writing an additional empty brace group.

LuaTEX callback new_graf made exclusive
Corrected an incorrect callback type which caused return values from the new_graf callback to be ignored and paragraph indentation to be suppressed. In the new version, only one new_graf callback handler can be active at a time which allows this handler to take full control of paragraph indentation. *(github issue 188)*

Changes to packages in the graphics category

Generate a warning if existing color definition is changed
If a color is defined twice using \DefineNamedColor, no info text Redefining color ... in named color model ... is written to the log file, because of a typo in the check. This has been corrected. *(gnats issue graphics/3635)*

Specifying viewport in the graphics package
Specifying a BoundingBox does not really have meaning when including non-EPS graphics in pdfTEX

and LuaTeX. For some years the `graphicx` package `bb` key has been interpreted (with a warning) as a `viewport` key. This feature has been added to the two-argument form of `\includegraphics` which is mostly used in the `graphics` package. `\includegraphics[1,2][3,4]{file}` will be interpreted in pdfTeX and LuaTeX in the same way as `graphicx \includegraphics[viewport=1 2 3 4]{file}`.

Normalizing `\endlinechar`

If `\endlinechar` is set to −1 so that ends of lines are ignored in special contexts, then a low level TeX error would be generated by code parsing BoundingBox comments. The package now locally sets `\endlinechar` to its standard value while reading files. *(github issue 286)*

Changes to packages in the tools category

array: Support stretchable glue in `w`-columns

If stretchable glue, e.g., `\dotfill`, is used in `tabular` columns made with the `array` package, it stretches as it would in normal paragraph text. The one exception was `w`-columns (but not `W`-columns) where it got forced to its nominal width (which in case if `\hfill` or `\dotfill` is 0 pt). This has been corrected and now `w`-columns behave like all other column types in this respect.
(github issue 270)

array: Use math mode for `w` and `W`-cells in `array`

The `w` and `W`-columns are LR-columns very similar to `l`, `c` and `r`. It is therefore natural to expect their cell content to be typeset in math mode instead of text mode if they are used in an `array` environment. This has now been adjusted. Note that this is a breaking change in version v2.5! If you have used `w` or `W`-columns in older documents either add `>{$}...<{$}` for such columns or remove the `$` signs in the cells. Alternatively, you can roll back to the old version by loading `array` with

    \usepackage{array}[=v2.4]

in such documents. *(github issue 297)*

array: Fix for `\firsthline` and `\lasthline`

Replacing `\hline` with `\firsthline` or `\lasthline` could lead in some cases to an increase of the tabular width. This has now been corrected. *(github issue 322)*

varioref: Support Japanese as a language option

The package now recognizes `japanese` as a language option. The extra complication is that for grammatical reasons `\vref`, `\Vref`, `\vrefrange` and `\fullref` need a structure different from all other languages currently supported. To accommodate for this `\vrefformat`, `\Vrefformat`, `\vrefrangeformat`, and `\fullrefformat` was added to all languages. *(github issue 352)*

xr: Support for spaces in filenames

The commannd `\externaldocument`, provided by xr, now also supports filenames with spaces, just like `\include` and `\includeonly`. *(github issue 223)*

Changes to packages in the amsmath category

Placement corrections for two accent commands

The accent commands `\dddot` and `\ddddot` (producing triple and quadruple dot accents) moved the base character vertically in certain situations if it was a single glyph, e.g., $Q \dddot{Q}$ were not at the same baseline. This has been corrected. *(github issue 126)*

Fixes to `aligned` and `gathered`

The environments `aligned` and `gathered` have a trailing optional argument to specify the vertical position of the environment with respect to the rest of the line. Allowed values are `t`, `b` and `c` but the code only tested for `b` and `t` and assumed anything else must be `c`. As a result, a formula starting with a bracket group would get mangled without warning—the group being dropped and interpreted as a request for centering. After more than 25 years this has now been corrected. If such a group is found a warning is given and the data is processed as part of the formula. *(github issue 5)*

Detect unicode engines when setting `\std@minus` and `\std@equal`

`amsmath` now detects the unicode engines and use their extended commands to define `\std@minus` and `\std@equal`. This avoids that package like `unicode-math` have to patch the code in the begin document hook to exchange the commands.

Changes to the babel package

Multilingual typesetting has much evolved in the past years, and `babel`, like LaTeX itself, has followed the footsteps of Unicode and the W3C consortia to produce proper output in many languages.

Furthermore, the traditional model to define and select languages (which can be called "vertical"), based on closed files, which is still the preferred one in monolingual documents, is being extended with a new model (which can be called "horizontal") based on *services* provided by `babel`, which allows to define and redefine locales with the help of simple `ini` files based on key/value pairs. The `babel` package provides about 250 of these files, which have been generated with the help of the Unicode Common Language Data Repository.

Thanks to the recent advances in `lualatex` and `luaotfload`, `babel` currently provides *services* for bidi typesetting, line breaking for South East Asian and CJK scripts, non-standard hyphenation (like ff to ff-f), alphabetic and additive counters, automatic selection

of fonts and languages based on the script, etc. This means babel can be used to typeset such a variety of languages as Russian, Arabic, Hindi, Thai, Japanese, Bangla, Amharic, Greek, and many others.

And since these ini files they are easily parsable, they can serve as a source for other packages.

For further details take a look at the babel package documentation [4].

References

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

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

[3] *LaTeX issue tracker.*
    `https://github.com/latex3/latex2e/issues/`

[4] Javier Bezos and Johannes Braams. *Babel —
    Localization and internationalization.*
    `https://www.ctan.org/pkg/babel`