LaTeX News, Issues 1–42

Contents

LaTeX News

Issue 1, June 1994 (LaTeX release 1994-06-01)

Welcome to LaTeX News

An issue of *LaTeX News* will accompany every future release of LaTeX. It will tell you about important events, such as major bug fixes, newly available packages, or any other LaTeX news.

LaTeX 2ε—the new LaTeX release

The most important news is the release of LaTeX 2ε, the new version of the LaTeX software. This version has better support for fonts, graphics and colour, and will be actively maintained by the LaTeX Project team. Upgrades will be issued every six months, in June and December.

Why a new LaTeX?

Over the years many extensions have been developed for LaTeX. This is, of course, a sure sign of its continuing popularity but it has had one unfortunate result: incompatible LaTeX formats came into use at different sites. Thus, to process documents from various places, a site maintainer was forced to keep LaTeX (with and without NFSS), SliTeX, AMS-LaTeX, and so on. In addition, when looking at a source file it was not always clear for which format the document was written.

To put an end to this unsatisfactory situation a new release of LaTeX was produced. It brings all such extensions back under a single format and thus prevents the proliferation of mutually incompatible dialects of LaTeX 2.09. The new release was available for several months as a test version, and the final release of 1 June officially replaces the old version.

Processing documents with LaTeX 2ε

Documents written for LaTeX 2.09 will still be read by LaTeX 2ε. Any such document is run in *LaTeX 2.09 compatibility mode*.

Unfortunately, compatibility mode comes with a price: it can run up to 50% slower than LaTeX 2.09 did. If you want to run your document in the faster *native mode*, you should try replacing the line:

```
\documentstyle[options,packages]{class}
```

with:

```
\documentclass[options]{class}
\usepackage{latexsym,packages}
```

Unfortunately, this will not always work, because some LaTeX 2.09 packages will only work in LaTeX 2ε compatibility mode. You should find out if there is a LaTeX 2ε version of the package available.

LaTeX 2ε native mode also gives access to the new features of LaTeX 2ε, described in *LaTeX 2ε for authors*.

New packages

LaTeX 2ε has much better support for graphics, colour, fonts, and multi-lingual typesetting. The following software should be available from the distributor who brought you LaTeX 2ε:

babel, for typesetting in many languages.

color, for colour support.

graphics, for including images.

mfnfss, for using bitmap fonts.

psnfss, for using Type 1 fonts.

tools, other packages by the LaTeX Project team.

The packages come with full documentation, and are also described in *LaTeX: A Document Processing System* or *The LaTeX Companion*.

Further information

More information about LaTeX 2ε is to be found in:

LaTeX: A Document Preparation System, Leslie Lamport, Addison Wesley, 2nd ed, 1994.

The LaTeX Companion, Goossens, Mittelbach and Samarin, Addison Wesley, 1994.

The LaTeX distribution comes with documentation on the new features of LaTeX:

LaTeX 2ε for authors, describes the new features of LaTeX documents, in the file `usrguide.tex`.

LaTeX 2ε for class and package writers, describes the new features of LaTeX classes and packages, in the file `clsguide.tex`.

LaTeX 2ε font selection, describes the new features of LaTeX fonts for class and package writers, in the file `fntguide.tex`.

For more information on TeX and LaTeX, get in touch with your local TeX Users Group, or the international TeX Users Group, P. O. Box 869, Santa Barbara, CA 93102-0869, USA, Fax: +1 805 963 8358, EMail: tug@tug.org.

LaTeX News

Issue 2, December 1994 (LaTeX release 1994-12-01)

Welcome to LaTeX News 2

An issue of *LaTeX News* will accompany every future release of LaTeX. It will tell you about important events, such as major bug fixes, newly available packages, or any other LaTeX news.

December 1994 release of LaTeX

December 1994 sees the second release of LaTeX 2ε. We are on schedule to deliver a release of LaTeX every six months, in December and June.

This release has seen quite a lot of activity, which is not too surprising as it's only been a year since the first test release of LaTeX 2ε. We don't expect so much activity in the next six months.

Many of the changes are minor improvements and bug-fixes—see *LaTeX 2ε for authors* (`usrguide.tex`), *LaTeX 2ε font selection* (`fntguide.tex`) and our change log (`changes.txt`) for more details.

However, there are two important new packages available for LaTeX: inputenc and AMS-LaTeX.

Accented input

One of the problems with writing non-English documents in LaTeX is the accent commands. Reading documents containing text like `na\"\i ve` is frustrating, especially if your keyboard allows you to type `naïve`.

In the past, LaTeX has not supported input containing accented characters such as `ï`, because Windows, Macintosh and Unix all have different ways of dealing with accented input, called *input encodings*.

However, the `inputenc` package allows you to specify which input encoding your document is written with, for example to use the ISO Latin-1 encoding, you type:

```
\usepackage[latin1]{inputenc}
```

At the moment, `inputenc` supports the `ascii` and `latin1` input encodings, but more will be added with future releases.

The `inputenc` package is currently a test release. The user interface for the full release will be upwardly compatible with the test version.

AMS-LaTeX

AMS-LaTeX is a set of miscellaneous extensions for LaTeX distributed by the American Mathematical Society. They provide superior information structure and superior printed output for mathematical documents.

There are far too many features of AMS-LaTeX to list here. AMS-LaTeX is described in the accompanying documentation, and in *The LaTeX Companion*.

Version 1.2beta of AMS-LaTeX was released for testing by intrepid users in October 1994. The full release of AMS-LaTeX 1.2 is expected in early January 1995.

It will be divided into two bundles:

- the amsfonts packages, which give access to hundreds of new mathematical symbols, and new math fonts such as blackboard bold and fraktur.
- the amsmath packages, which provide finer control over mathematical typesetting, such as multi-line subscripts, enhanced theorem and proof environments, and improved displayed equations,

For compatibility with older documents, an amstex package will be provided.

LaTeX on the internet

LaTeX has its own home page on the World Wide Web, with the URL:

```
http://www.tex.ac.uk/CTAN/latex/
```

This page describes LaTeX and the LaTeX3 project, and contains pointers to other LaTeX resources, such as the user guides, the TeX Frequently Asked Questions, and the LaTeX bugs database.

The electronic home of anything TeX-related is the Comprehensive TeX Archive Network (CTAN). This is a network of cooperating ftp sites, with over a gigabyte of TeX material:

```
ftp://ftp.tex.ac.uk/tex-archive/
ftp://ftp.shsu.edu/tex-archive/
ftp://ftp.dante.de/tex-archive/
```

For more information, see the LaTeX home page.

Further information

For more information on TeX and LaTeX, get in touch with your local TeX Users Group, or the international TeX Users Group, P. O. Box 869, Santa Barbara, CA 93102-0869, USA, Fax: +1 805 963 8358, EMail: tug@tug.org.

LaTeX News

Issue 3, June 1995 (LaTeX release 1995-06-01)

Welcome to LaTeX News 3

An issue of *LaTeX News* will accompany every future release of LaTeX. It will tell you about important events, such as major bug fixes, newly available packages, or any other LaTeX news.

June 1995 release of LaTeX

June 1995 sees the third release of LaTeX 2_ε. We are on schedule to deliver a release of LaTeX every six months, in December and June.

In the last *LaTeX News*, we said "we don't expect so much activity in the next six months," which has turned out not to be true!

Additional input encodings

In the last release of LaTeX we distributed a test version of the inputenc package which allows the use of input characters other than just a–z and A–Z. The package has proved to be robust, so we are now distributing an expanded version. The new release comes with a number of input encodings:

- `ascii` the standard encoding,
- `latin1` the ISO Western European alphabet,
- `latin2` the ISO Eastern European alphabet,
- `cp437` the IBM codepage 437,
- `cp850` the IBM codepage 850, and
- `applemac` the Apple Macintosh encoding.

These can be used by specifying an option to the inputenc package, for example:

```
\usepackage[latin1]{inputenc}
```

The new input encodings are currently being tested, but we don't expect any major changes.

LaTeX getting smaller

In the past releases of LaTeX 2_ε, the amount of memory LaTeX requires has increased, but we are pleased to say that this trend has been reversed. We hope that future releases of LaTeX will continue to get smaller.

For example, on this document, the December 1994 release used 52,622 words of memory, and the June 1995 release uses 51,216 words of memory, which is a 2.7% reduction.

We are currently experimenting with other ways of reducing the size of LaTeX. For example, we are experimenting with an option to remove the `picture` and `tabbing` environments from the LaTeX kernel, and to load them from a file the first time they are used. This should help LaTeX to run on machines with limited memory. See `autoload.txt` for details.

Distribution and modification

One topic of discussion that has kept us busy is the distribution and modification conditions of LaTeX. We are committed to keeping LaTeX as free reliable software, and ensuring that (as far as possible) LaTeX documents will produce the same results on all systems.

The modification conditions are currently under discussion, and we would like to hear from anyone interested. Please read `modguide.tex` for more information.

AMS-LaTeX full release

The AMS-LaTeX packages were still in beta test in the December 1994 release of LaTeX, and the full release came out in January 1995.

AMS-LaTeX is described in the *User's Guide* (`amsldoc.tex`) and in *The LaTeX Companion*.

PostScript fonts

There is a new test release of the PSNFSS packages for accessing PostScript fonts in LaTeX 2_ε. This includes an update to all of the fonts, to remove many of the underfull and overfull `\hbox` warnings, and improve the setting of non-English languages.

The new release of LaTeX removes all of the 'hidden' uses of Computer Modern mathematics. For example, the footnote markers used to use math mode, so always used Computer Modern digits rather than ones from the current text font. This has now been fixed.

Further information

For more information on TeX and LaTeX, get in touch with your local TeX Users Group, or the international TeX Users Group, P. O. Box 869, Santa Barbara, CA 93102-0869, USA, Fax: +1 805 963 8358, EMail: tug@tug.org.

The LaTeX home page is `http://www.tex.ac.uk/ctan/latex/` and contains links to other WWW resources for LaTeX.

LaTeX News

Issue 4, December 1995 (LaTeX release 1995-12-01)

Welcome to LaTeX News 4

An issue of *LaTeX News* will accompany every future release of LaTeX. It will tell you about important events, such as major bug fixes, newly available packages, or any other LaTeX news. This issue accompanies the fourth release of LaTeX 2_ε.

LaTeX getting smaller

The last release in June started a trend of LaTeX becoming smaller, we are pleased to announce that this has continued with this release. In particular the experimental 'autoload' version described in `autoload.txt` is much smaller as more parts of LaTeX are autoloaded.

New 'concurrent' docstrip

The time taken to 'unpack' this release from the documented sources should be much reduced (roughly half the time, depending on installation conditions). This is due to an improved version of the docstrip program that has been contributed by Marcin Woliński. This can write up to 16 files at once. The previous version could only write one file at a time which meant that it was very slow when producing many small files from the same source file as the source needed to be re-read for each file written.

New T1 encoded fonts

This year Jörg Knappen has completed a new release of the 'Cork' (T1) encoded Computer Modern fonts: the dc fonts release 1.2.

This release of the dc fonts fixes many bugs (including the missing `?‘` (¿) and `!‘` (¡) ligatures) and improves the fonts in many other ways. It is strongly recommended that you upgrade as soon as possible if currently you are using the old dc fonts, release 1.1 or earlier. The new fonts are available from the CTAN archives, in `tex-archive/fonts/dc`.

The names of the font files are *different*. This does not affect LaTeX documents but *does* affect the installation procedure as it assumes that you have the *new* fonts, and will write suitable 'fd' files for those fonts. If you have not yet upgraded your dc fonts then, after unpacking the distribution, you *must* `latex olddc.ins` to produce 'fd' files for the old dc fonts. This must be done *before* the format is made. Running the test document at `ltxcheck.tex` the end of the installation will inform you if the wrong set of 'fd' files has been installed.

Note that this change does not affect the standard 'OT1' Computer Modern fonts that LaTeX uses by default.

More robust commands

The commands `\cite` and `\sqrt` are now robust.

Although most commands with optional arguments are fragile, as documented, such commands defined using the second optional argument of `\newcommand` and its derivatives are now *robust*.

New Interface to building 'extension' classes

The mechanism provided by `\DeclareOption`, `\ProcessOptions` and `\LoadClass` has proved to be a powerful and expressive means of defining one class in terms of another 'base' class. However there have been some requests to simplify the declaration of the common case where you want the 'base' class to be called with *all* the options that were specified to the extension class. This is now provided by the new command `\LoadClassWithOptions`. A similar command `\RequirePackageWithOptions` is provided for package use. More details of this feature are provided in `clsguide.tex` and `ltclass.dtx`.

More Input Encodings

The experimental `inputenc` package allows a more natural style of input of accented and other characters.

Three new input encodings are now supported.

- `ansinew` the Windows ansi encoding, as used in Microsoft Windows 3.x.
- `cp437de` a variant of `cp437`, which uses ß rather than β in the appropriate slot.
- `next` the encoding used on Next computers.

Further information

For more information on TeX and LaTeX, get in touch with your local TeX Users Group, or the international TeX Users Group, 1850 Union Street, #1637, San Francisco, CA 94123, USA, Fax: +1 415 982 8559, EMail: tug@tug.org. The LaTeX home page is `http://www.tex.ac.uk/ctan/latex/` and contains links to other WWW resources for LaTeX.

LaTeX News

Issue 5, June 1996 (LaTeX release 1996-06-01)

Welcome to LaTeX News 5

This issue of *LaTeX News* accompanies the fifth release of the new standard LaTeX, LaTeX 2_ε.

Extra possibilities for section headings

Most LaTeX sectioning commands are defined using `\@startsection`. For example, the `article` class defines:

```
\newcommand\section{\@startsection
 {section}{1}{0pt}{-3.5ex plus-1ex minus-.2ex}%
 {2.3ex plus.2ex}{\normalfont\Large\bfseries}}
```

The last argument specifies the style in which the section heading is to be typeset.

The new feature added at this release is that at the *end* of this argument you may specify a command that *takes an argument*. This command will be applied to the section number and heading. For example, one could use the `\MakeUppercase` command to produce uppercase headings. A package or class file could contain:

```
\renewcommand\section{\@startsection
 {section}{1}{0pt}{-3.5ex plus-1ex minus-.2ex}%
 {2.3ex plus.2ex}{\normalfont\Large\MakeUppercase}}
```

to produce section headings using uppercase medium weight text, rather than the bold text used by `article`. Note that, like the font choice, the uppercasing applies only to the actual heading (including any automatically generated section number), not to the text as it may appear in the running head or table of contents.

The 'openany' option in the 'book' class

The `openany` option allows chapter and similar openings to occur on left hand pages. Previously this option only affected `\chapter` and `\backmatter`. It now also affects `\part`, `\frontmatter` and `\mainmatter`.

More font (output) encodings

The font encoding name `T3` has been allocated to the encoding used in the new 256-character IPA fonts (for the phonetic alphabet) produced by Rei Fukui. His package, tipa, gives access to these fonts and should soon be available. (The encoding named `OT3` is the 128-character encoding used in the IPA fonts produced by Washington State University.)

More input encodings supported

The inputenc package now supports the IBM codepage 852 used in Eastern Europe, with the option `[cp852]` contributed by Petr Sojka.

Also, the inputenc package now activates most 'control codes' with ASCII values below 32. Currently none of the encodings in the standard distribution makes use of these positions.

Fixes and improvements

The LaTeX kernel has only had minor changes, apart from `\@startsection` mentioned above. However, some small fixes have been incorporated removing the following problems:

- In tabular and array, previous versions of LaTeX 'lost' the inter-column space from an '`l`'-column, when that column was completely empty.
- Previously, the use of the `\nofiles` command could change the *vertical spacing* in a document. A side effect of fixing this is that when `\nofiles` is used, `\label` puts a blank line in the log file.
- LaTeX often loads fonts 'on demand'. Previously, this could happen inside the argument of an accent command and this would cause the accent to appear in the wrong place.

Changes to the 'tools' packages

- The longtable package now uses a modified algorithm, contributed by David Kastrup, to align the 'chunks' of a table. It is now unnecessary to edit the document to add `\setlongtables` before the final run of LaTeX. In certain cases of overlapping `\multicolumn` entries, the new algorithm will produce better column widths than the old (at the price of extra passes through LaTeX).
- The dcolumn package now has the extra possibility of specifying the number of digits both *before* and after the 'decimal point'. This makes it easy to centre the column of numbers under a wide heading.

New copy of the LaTeX bug database

`http://www.tex.ac.uk/ctan/latex/bugs.html` will soon have links to a copy of the searchable LaTeX bugs

database at Mainz (Germany) as well as the original copy at Sussex (England).

LaTeX News

Issue 6, December 1996 (LaTeX release 1996-12-01)

Welcome to LaTeX News 6

This issue of *LaTeX News* accompanies the sixth release of the new standard LaTeX, LaTeX 2_ε.

Mono-case file names

Previously LaTeX has used some files with 'mixed-case' file names such as `T1cmr.fd` and `T1enc.def`.

These file names cause problems on some systems (in particular they are illegal on the ISO 9660 CDROM format) and so in this release all file names have been made lowercase (for example `t1cmr.fd` and `t1enc.def`).

This change should *not* affect any document. Within LaTeX, encodings still have the usual uppercase names in uses such as `\usepackage[T1]{fontenc}` and `\fontencoding{T1}`. LaTeX will automatically convert to the lowercase form while constructing the file name. LaTeX will input the 'fd' file under the old name if it fails to find the file with the new name, so existing collections of fd files should still work with this new release.

The change *does* affect the configuration files that may be used to make the LaTeX format with initex. For example, the file `fonttext.ltx` previously specified `\input{T1cmr.fd}`. It now has `\input{t1cmr.fd}`. If you use a local file `fonttext.cfg` you will need to make similar changes, as `\input{T1cmr.fd}` will not work as `T1cmr.fd` is no longer in the distribution.

The files affected by this change all have names of the form `*.fd` or `*enc.def`.

Another input encoding

Thanks to work by Søren Sandmann, the inputenc package now supports the IBM codepage 865 used in Scandinavia.

Better user-defined math display environments

Suppose that you want to define an environment for displaying text that is numbered as an equation. A straightforward way to do this is as follows:

```
\newenvironment{texteqn}
  {\begin{equation}
     \begin{minipage}{0.9\linewidth}}
    {\end{minipage}
  \end{equation}}
```

However, if you have tried this then you will probably have noticed that it does not work perfectly when used in the middle of a paragraph because an inter-word space appears at the beginning of the first line after the environment.

There is now an extra command (with a very long name) available that you can use to avoid this problem; it should be inserted as shown here:

```
\newenvironment{texteqn}
  {\begin{equation}
     \begin{minipage}{0.9\linewidth}}
    {\end{minipage}
  \end{equation}
  \ignorespacesafterend}
```

Docstrip improvements

The docstrip program that is used to unpack the LaTeX sources has undergone further development. The new version should be able to process all old 'batchfiles' but it allows a simpler syntax in new 'batchfiles' (no need to define `\def\batchfile{`...`}`).

It also allows 'target' directories to be specified when writing files. This directory support is disabled by default unless activated in a local `docstrip.cfg` configuration file. See `docstrip.dtx` for details.

AMS LaTeX update

Since the last LaTeX release in June, the American Mathematical Society have re-issued the 'AMSLaTeX' classes and packages, fixing several reported problems.

Graphics package update

The LaTeX color and graphics packages have been updated slightly, principally to support more dvi drivers, see the readme file in the graphics distribution.

EC Fonts released

The first release of the Extended Computer Modern fonts has just been made. (In January 1997.)

This release of LaTeX does *not* default to these 'ec' fonts as its T1 encoded fonts. By default it will use the 'dc' fonts if the T1 encoding is requested.

As noted in `install.txt` you may run TeX on the install file `ec.ins` *after* unpacking the base distribution but *before* making the LaTeX format. This will produce suitable 'fd' files making LaTeX (including, for the first time, the `slides` class) use the 'ec' fonts as the default T1 encoded font set.

LaTeX News

Issue 7, June 1997 (LaTeX release 1997-06-01)

T1 encoded Computer Modern fonts

As in the last release the base LaTeX distribution contains three different sets of 'fd' files for T1 encoded fonts.

In this release the default installation uses `ec.ins` and so installs files suitable for the current 'EC fonts' distribution. If you have still not updated to the EC fonts and are using the earlier test versions, known as DC then you should unpack `newdc.ins` (for DC release 1.2 or later) or `olddc.ins` (for the original releases of the DC fonts). This should be done after unpacking `unpack.ins` but before making the format by running iniTeX on `latex.ltx`. There are further details in `install.txt`.

T1 encoded Concrete fonts

The Metafont sources for T1 encoded 'Concrete' fonts have been removed from the mfnss distribution as they were based on the now obsolete DC fonts release 1.1. Similarly the `cmextra.ins` install file in the base distribution no longer generates fd files for the 'Concrete' fonts. To use these fonts in either T1 or OT1 encoding it is recommended that you obtain Walter Schmidt's ccfonts package and fonts from CTAN `macros/latex/contrib/supported/ccfonts`.

Further input encodings

Two more inputenc packages have been added: for latin5, thanks to H. Turgut Uyar; and for latin3, thanks to Jörg Knappen.

Normalising spacing after punctuation

The command `\normalsfcodes` was introduced at the last patch release. This is normally given the correct definition automatically and so need not be explicitly set. It is used to correct a problem, reported by Donald Arseneau, that punctuation in page headers has always (in all known TeX formats) been potentially incorrect if the page break happens while a local setting of the space codes (for instance by the command `\frenchspacing`) is in effect. A common example of this happening in LaTeX is in the `verbatim` environment.

Accessing Bold Math Symbols

The tools distribution contains a new package, bm, which defines a command `\bm` that allows individual bold symbols to be accessed within a math expression (in contrast to `\boldmath` which makes whole math expressions default to bold fonts). It is more general than the existing amsbsy package; however, to ease the translation of documents between these two packages, bm makes `\boldsymbol` an alias for `\bm`.

This package was previously made available from the 'contrib' area of the CTAN archives, and as part of Y&Y's LaTeX support for the MathTime fonts.

Policy on standard classes

Many of the problem reports we receive concerning the standard classes are not concerned with bugs but are suggesting, more or less politely, that the design decisions embodied in them are 'not optimal' and asking us to modify them.

There are several reasons why we have decided not to make such changes to these files.

- However misguided, the current behaviour is clearly what was intended when these classes were designed.
- It is not good practice to change such aspects of 'standard classes' because many people will be relying on them.

We have therefore decided not to even consider making such modifications, nor to spend time justifying that decision. This does not mean that we do not agree that there are many deficiencies in the design of these classes, but we have many tasks with higher priority than continually explaining why the standard classes for LaTeX cannot be changed.

We would, of course, welcome the production of better classes, or of packages that can be used to enhance these classes.

New addresses for TUG

For information about joining the TeX Users Group, and about lots of other LaTeX-related matters, please contact them at their new address:

TeX Users Group, P.O. Box 1239,
Three Rivers, CA 93271-1239, USA
Fax: +1 209 561 4584
E-mail: `tug@mail.tug.org`
URL: `http://www.tug.org/`

LaTeX News

Issue 8, December 1997 (LaTeX release 1997-12-01)

New supported font encodings

Two new font encodings are supported as options to the fontenc package:

OT4 This is a seven-bit encoding designed for Polish. The LaTeX support was developed by Mariusz Olko.

TS1 This is the 'Text Companion Encoding'; it contains symbols designed to be used in text, as opposed to mathematical formulas, and some accents designed for uppercase letters. It is currently supported by the 'tc' fonts, which match the T1 encoded 'ec' text fonts. A subset of the glyphs in this encoding is supported by virtual fonts distributed with the PostScript font metrics on the CTAN archives. (This is the '8c' encoding in Karl Berry's fontname scheme.) The textcomp package provides access to this encoding but here is a warning to current users of that package: some of the internal names for the characters have changed.

New input encodings

These additions to the inputenc package are `decmulti` (the DEC Multinational Character Set, contributed by M. Y. Chartoire) and `cp1250` (an MS-Windows encoding for Central and Eastern Europe, contributed by Marcin Woliński). There is also a `cp1252` encoding that is identical to `ansinew`.

Tools

The calc package (used in many examples in *The LaTeX Companion*) has been contributed to this distribution by Kresten Krab Thorup and Frank Jensen. This is essentially the same as the version that has been available from the CTAN archives for some time, with one minor change: to use LaTeX-style error messages. It enables the use of arithmetic expressions within arguments to standard LaTeX commands where a length or a counter value is required. For example:

```
\setcounter {page} { \value{page} * 2 + 1 }
\parbox { 3in - ( 2mm + \textwidth / 9 ) }
```

There have also been some improvements to several other packages in this collection. In particular, bm now works correctly with constructions such as `\bm{f'}` involving `'` or other characters which use TeX's special "`\mathcode"8000`" feature. Also, multicol sets the length `\columnwidth` to an appropriate value; this enables it to work with classes that support two-column setting, e.g., the AMS classes.

Graphics

The special `oztex.def` driver file has been removed, and OzTeX support has been merged with dvips, following advice from Andrew Trevorrow about OzTeX 3.x.

The keyval package has had some internal improvements: to use LaTeX format error messages; and to avoid '`#` doubling'. This latter change means that the `command` key for the graphicx version of `\includegraphics` should now be used with one `#` rather than two. For example, `command = 'gunzip #1`. Fortunately this key is almost never used in practice, so few if any documents should be affected by this change.

LaTeX3 experimental programming conventions

As announced at the TeX Users Group meeting (Summer 1997), a group of highly experimental packages will soon be released to allow experienced TeX programmers to experiment with, and comment on, a proposed set of syntax conventions and basic data-types that might form the basis for programming large scale projects in TeX. They will be located in this CTAN directory:

```
CTAN:macros/latex/packages/expl3
```

The documentation of this material is as follows: individual package files provide outline, draft documentation; there is an article that gives an overview of the syntax and related concepts; there is a `readme.txt` file containing a brief description of the collection.

All aspects of these packages are liable, indeed likely, to change. They should not be used at this stage for anything that requires a stable system. However, we do encourage people to experiment with these packages, and to send comments on them to the `LaTeX-L` mailing list. To subscribe to this list, mail to:

```
listserv@urz.uni-heidelberg.de
```

the following one line message:

```
subscribe LATEX-L <first-name> <second-name>
```


LaTeX News

Issue 9, June 1998 (LaTeX release 1998-06-01)

New math font encodings

A joint working group of the TeX Users Group and the LaTeX Project is developing a new 8-bit math font encoding for TeX. It is designed to overcome several limitations and implementation problems of the old math font encodings and to simplify switching between different sets of math fonts, much as the LaTeX font selection interface has simplified switching between text fonts.

Since the work on this project relies entirely on volunteer work, we cannot give a specific release date yet. However, a prototype implementation already exists. This contains several sets of virtual fonts, some LaTeX packages and a kernel module; we hope to integrate it into the main LaTeX distribution for the next release.

Documents using only standard LaTeX commands for math symbols should not be affected by switching to the new math font encodings However, documents, classes or packages making specific assumptions about the encoding of math symbol fonts are likely to break.

Further information about the Math Font Group may be found on the World Wide Web at `http://www.tug.org/twg/mfg/`.

A new math accent

A new math accent, `\mathring`, has been added. This is a math mode version of the ring accent (˚) which is available in text mode with the command `\r`.

Extended `\DeclareMathDelimiter`

The command `\DeclareMathDelimiter` has been extended. Normally this command takes six arguments. Previously, when being used to declare a character (such as `[`) as a delimiter, a variant form was used with only five arguments. The argument specifying the default 'math class' was omitted. Now the full six-argument form may be used in this case. The extra information is used to implicitly declare the character via `\DeclareMathSymbol` for use when the symbol is not used with `\left` or `\right`.

The old five-argument form is detected and will work as before.

Tools distribution

The multicol package now supports the production of multiple columns without balancing the last page. To get this effect use the `multicols*` environment.

The layout package was partly recoded by Hideo Umeki to display page layout effects in a better way.

As suggested by Donald Arseneau, the calc package was extended to support the new commands `\widthof{`*text*`}`, `\heightof{`*text*`}`, and `\depthof{`*text*`}` within a calc-expression. At the same time we modified a few kernel commands so that calc-expressions can now be used in various useful places such as the dimension arguments to the `tabular` environment and the `\rule` command. For many other standard LaTeX commands this was already possible.

Support for Cyrillic encodings

We are very pleased that, after a lengthy period of development, a set of fonts, encodings and support files for using LaTeX with Cyrillic characters will soon be available.

Test versions of the 'LH' fonts for these Cyrillic encodings, based on the Computer Modern design, are available from CTAN archives in the directory `fonts/cyrillic/lh-test`. The LaTeX support files (by Werner Lemberg and Vladimir Volovich) are also available from CTAN archives in `macros/latex/contrib/supported/t2`

Default docstrip header

Many LaTeX users now distribute packages in documented source form using the docstrip system. Docstrip allows a header to be placed on generated package files, suitable for giving copyright information, or distribution conditions.

We have changed the default version of this header so that it allows stripped files to be distributed in ready-to-run installations such as the TeXLive CD. If you use the default header for distributing your files you should check that the new copyright text is acceptable to you. The file `docstrip.dtx` explains how to produce your own header if you wish to do so.

LaTeX News

Issue 10, December 1998 (LaTeX release 1998-12-01)

Five years of LaTeX 2ε

Since this is the 10th edition of LaTeX News, the (no longer) New Standard LaTeX must have hit the streets almost this long ago. In fact it was only the beta-version that some people got just in time for Christmas 1993, and since then there has been a lot of tidying-up and smoothing of rough edges (not to mention a few bug fixes!).

Maybe it is time for something more radically different to emerge and be hungrily adopted by the world; but don't panic, we shall be maintaining what you have now for a long time yet. Amongst the more polite things that have been written about our efforts, we found that this quote (somewhat censored to protect the guilty) well reflects some of our feelings about working on LaTeX over the years: *the mere existence of LaTeX 2ε is a great miracle.*

Restructuring the LaTeX distribution

Since the (once) 'new' standard LaTeX has reached such a venerable age, we are reviewing the way in which the system is presented to the world.

An early intention is to define, given the wide variety of good packages now available, what now constitutes a useful installation of LaTeX. We also hope that such a definition will help document portability if it leads to a future in which a LaTeX class designer can reasonably assume that a known list of facilities will be there for all users (so that each class need not supply them).

As a first small step towards this definition, we shall replace the `latex/packages` subdirectory on CTAN. This directory was a curious mixture of the important, such as the LaTeX tools, that any self-respecting LaTeX installation ought to have, and the esoteric or experimental.

The esoterica from packages will be moved to new locations, as follows:

expl3 to `latex/exptl/project`
mfnfss to `latex/contrib/supported/mfnfss`

The subdirectory that replaces packages will be called `latex/required`; all the other sub-directories of `packages` will be moved there.

LaTeX Project on the Internet

A new `latex-project.org` domain has been registered. The web site is not yet fully functional but the old LaTeX pages from CTAN are available at `http://www.latex-project.org/` and the LaTeX bug reporting address has been changed to `latex-bugs@latex-project.org`.

Restructuring the LaTeX package licenses

Several people have requested an easy mechanism for the distribution of LaTeX packages and other software "under the same conditions as LaTeX". The old `legal.txt` file was unsuitable as a general licence as it referred to specific LaTeX authors, and to specific files.

Therefore, in this release `legal.txt` contains just the copyright notice and a reference to the new *LaTeX Project Public License* (LPPL) for the distribution and modification conditions. The tools, graphics, and mfnfss packages also now refer to this license in their distribution notices.

Support for Cyrillic encodings

Basic Cyrillic support, as announced in LaTeX News 9, is now finally an official part of LaTeX. It includes support for the following standard Cyrillic font encodings (this list may grow): `T2A T2B T2C X2`.

It also includes various Cyrillic input encodings (20 in total, including commonly used variants and Mongolian Cyrillic encodings). This provides platform independent and sophisticated basic support for high-quality typesetting in various Cyrillic-based languages.

For further information see the file `cyrguide.tex`.

Tools distribution

The varioref package has been extended to support textual page references to a range of objects: e.g., if `eq-first` and `eq-last` are the label names for the first and last equation in a sequence, then you can now write

```
see~\vrefrange{eq-first}{eq-last}
```

This results in different text depending on whether both labels fall on the same page.

Some additional user commands, as well as building-blocks for writing private extensions, are described in the accompanying documentation.

LaTeX News

Issue 11, June 1999 (LaTeX release 1999-06-01)

Back in sync

The last release of LaTeX was delayed even longer than you have come to expect. We hope that it proved worth waiting for. It required a major integration of the code from several people and, independently, the introduction of the LPPL (see LaTeX News 10) plus several related changes to our internal systems. It therefore seemed sensible to wait until everything was complete rather than do things in too much hurry.

This seem to have been a successful strategy as the recent patch release was related to an isolated change that was done many months previously. If this release does not appear a lot closer to its nominal date then ... well, you will not be reading this sentence!

Yearly release cycles

With the year 2000 rapidly approaching, we intend to switch to a release frequency of just one per year (with patches if necessary) for the core of LaTeX 2_ε. These days the system is sufficiently stable that the original update policy is costing everybody more time than is now warranted.

LPPL update

Thanks to extensive and valuable input from Matt Swift (`swift@alum.mit.edu`) we now have a clearer and more detailed form of the LaTeX Project Public Licence. This release contains both the original version (in `lppl-1-0.txt`) and the updated version, LPPL 1.1.

The future of SliTeX

We still get a very small trickle of reports about this part of the system (if you are no longer able to recall LaTeX 2.09 then you will know it as the `slides` class). We have not classified them (in our minds at least) as bugs since we have always known that there are many problems with this class. It is clear to us that the only sensible action would be to redesign the system completely; in particular, to remove much of its complexity whose purpose is to support 10-year-old overlay technology. However, this would take a lot too much time and would be completely out of proportion to its current usage.

We are therefore planning to make the `slides` class unsupported in the sense that any problem related to the use of invisible fonts is considered to be a feature (The LaTeX 2_ε manual by Leslie Lamport doesn't even describe this part of the class any more). Of course, if it still has its enthusiasts then we are happy to cede it to their loving care (somewhat like a preserved steam locomotive, in some parts of the world).

Fontenc package peculiarities

The `\usepackage` interface normally ensures that a package is loaded only once. The fontenc package has become an exception to this rule: it can be loaded several times using different options, e.g., allowing the user to add a font encoding in the preamble. This comes at a price for package writers: the low-level commands (see `ltclass.dtx`) used to check if a package was loaded, and with which options, do not work for the fontenc package.

New math font encodings

As we announced in LaTeX News 9, a joint working group of the TeX Users Group and the LaTeX Project has developed a new 8-bit math font encoding for TeX. The reason why this work is not yet released is because of other exciting developments in the world of math fonts and math characters. It is obviously wise to ensure that the encoding work is fully integrated with the available fonts.

Those interested are reminded that further information about the Math Font Group may be found on the World Wide Web at:
http://www.tug.org/twg/mfg/.

Tools distribution

The multicol package has now got a small but useful extension which allows you to force a column break where this is really necessary. This is done with the command `\columnbreak`, which can be used like `\pagebreak` (e.g., within paragraphs) except that it cannot have an optional argument and thus it always forces a new column.

Coming soon

Major work on a new class file structure to support flexible designs is well under way; some of this work will be presented at the TUG'99 conference in Vancouver, Canada. With a bit of luck much of this work could be ready for integration into the next release—so watch this space!

LaTeX News

Issue 12, December 1999 (LaTeX release 1999-12-01)

LPPL update

Since the release of the LaTeX Project Public Licence version 1.1, we have received a small number of queries which resulted in some minor changes to improve the wording or explain the intentions better. As a consequence this release now contains LPPL 1.2 in the file `lppl.txt` and the previous versions as `lppl-1-0.txt` and `lppl-1-1.txt`.

fixltx2e package

This package provides fixes to LaTeX 2$_\varepsilon$ which are desirable but cannot be integrated into the LaTeX 2$_\varepsilon$ kernel directly as they would produce a version incompatible to earlier releases (either in formatting or functionality).

By having these fixes in the form of a package, users can benefit from them without the danger that their documents will fail, or produce unexpected results, at other sites; this works because a document will contain a clear indication (the `\usepackage` line, preferably with a required date) that at least some of these fixes are required to format it.

Outcome of TUG '99 (Vancouver)

The slides from the TUG'99 presentation we gave on *a new interface for LaTeX class designers* are available from the LaTeX Project website; look for the file `tug99.pdf` at:

http://www.latex-project.org/talks/

Please note that this document was intended only to be informal "speaker's notes" for our own use. We decided to make them available (the speaker's notes as well as the slides that were presented) because several people requested copies after the talk. However, they are *not* in a polished copy-edited form and are not intended for publication.

Prototype implementations of parts of this interface are now available from:

http://www.latex-project.org/code/experimental/

We are continuing to add new material at this location so as to stimulate further discussion of the underlying concepts. As of December 1, 1999 the following parts can be downloaded.

xparse Prototype implementation of the interface for declaring document command syntax. See the `.dtx` files for documentation.

template Prototype implementation of the template interface (needs parts of xparse).

The file `template.dtx` in that directory has a large section of documentation at the front describing the commands in the interface and giving a 'worked example' building up some templates for caption formatting.

xcontents Interface description for table of contents data (no code yet). Coding examples have been thoroughly discussed on the `latex-l` list.

xfootnote Working examples for generating footnotes, etc. Needs xparse and template.

All examples are organised in subdirectories and additionally available as `gzip tar` files.

Please remember that this material is intended only for experimentation and comments; thus any aspect of it, e.g., the user interface or the functionality, may change and, in fact, is very likely to change. For this reason it is explicitly forbidden to place this material on CD-ROM distributions or public servers.

These concepts, as well as their implementation, are under discussion on the list `LATEX-L`. You can join this list, which is intended solely for discussing ideas and concepts for future versions of LaTeX, by sending mail to `listserv@URZ.UNI-HEIDELBERG.DE` containing the line

`SUBSCRIBE LATEX-L` *Your Name*

This list is archived and, after subscription, you can retrieve older posts to it by sending mail to the above address, containing a command such as:

`GET LATEX-L LOGyymm`

where `yy`=Year and `mm`=Month, e.g.

`GET LATEX-L LOG9910`

for all messages sent in October 1999.

LaTeX News

Issue 13, June 2000 (LaTeX release 2000-06-01)

Yearly release cycle

We announced in *LaTeX News 11* that we intended to switch to a 12-monthly release schedule. With the present (June 2000) release, this switch is being made: thus the next release of LaTeX will be dated June 2001. We shall of course continue, as in the past, to release patches as needed to fix significant bugs.

PSNFSS: Quote of the Month

> You should say in the LaTeX News that Walter Schmidt has taken over PSNFSS from me. It gives me a certain pleasure to be able to draw a line under that part of my life...
>
> Sebastian Rahtz

The PSNFSS material, which supports the use of common PostScript fonts with LaTeX, has been thoroughly updated. Most noticeably, the mathpple package, which used to be distributed separately, is now part of the basic PSNFSS bundle; this package provides mathematical typesetting with the Palatino typeface family. In addition, numerous bugs and flaws have been fixed and the distribution has been 'cleaned up'. The file `changes.txt` contains a detailed list of these changes.

The documentation (in `psnfss2e.pdf`) has been completely rewritten to provide a comprehensive introduction to the use of PostScript fonts.

Notice that the new PSNFSS needs updated files for font metrics, virtual fonts and font definitions. If you received the new version (8.1) as part of a complete TeX system then these new font files should also have been installed. However, if you intend to install or update PSNFSS yourself, please read the instructions in the file `00readme.txt` of the new PSNFSS distribution.

Support for commercial PostScript fonts, such as Lucida Bright, has been removed from the basic distribution; it is now available from CTAN: `http://mirror.ctan.org/macros/latex/contrib/supported/psnfssx`.

New AMS-LaTeX

Version 2.0 of AMS-LaTeX was released on December 1, 1999. It can be obtained via ftp://ftp.ams.org/pub/tex/ or http://www.ams.org/tex/amslatex.html, as well from CTAN: `http://mirror.ctan.org/macros/latex/required/amslatex`.

This release consists chiefly of bug fixes and consolidation of the existing features. The division of AMS-LaTeX into two main parts (the math packages; the AMS document classes) has been made more pronounced. The files `diffs-m.txt`, `diffs-c.txt`, `amsmath.faq`, and `amsclass.faq` describe the changes and address some common questions.

The primary documentation files remain `amsldoc.tex`, for the amsmath package, and `instr-l.tex`, for the AMS document classes. The documentation for the amsthm package, however, has been moved from `amsldoc.tex` to a separate document `amsthdoc.tex`.

New input encoding `latin4`

The package inputenc has, thanks to Hana Skoumalová, been extended to cover the `latin4` input encoding; this covers Baltic and Scandinavian languages as well as Greenland Inuit and Lappish.

New experimental code

In *LaTeX News 12* we announced some ongoing work towards a 'Designer Interface for LaTeX' and we presented some early results thereof. Since then, at Gutenberg 2000 in Toulouse and TUG 2000 in Oxford, we described a new output routine and an improved method of handling vertical mode material between paragraphs. In combination these support higher quality *automated*[1] page-breaking and page make-up for complex pages—the best yet achieved with TeX!

A paper describing the new output routine is at `http://www.latex-project.org/papers/xo-pfloat.pdf` All code examples and documentation are available at `http://www.latex-project.org/code/experimental/`

This directory has been extended to contain

galley Prototype implementation of the interface for manipulating vertical material in galleys.

xinitials Prototype implementation of the interface for paragraph initials (needs the galley package.

xtheorem Contributed example using the template package to provide a designer interface for theorem environments.

xoutput A prototype implementation of the new output routine as described in the `xo-pfloat.pdf` paper. Expected availability: at or shortly after the TUG 2000 conference.

[1] The stress here is on automated!

LaTeX News

Issue 14, June 2001 (LaTeX release 2001-06-01)

Future releases

We are currently exploring how to best support the very large community of individuals, organisations and enterprises that depend on the robustness and availability of the current standard LaTeX distribution. The results of this may lead to some changes in the regular release schedule and the handling of bug reports during the next year.

New release of Babel (required)

Earlier this year a new release of Babel (3.7) became available. You can read about its new features in
`http://www.ctan.org/tex-archive/macros/latex/required/babel/announce.txt`

One of the bugs that got fixed in this release deals with how labels are handled by LaTeX. Because this part of the kernel is modified by babel, the relevant changes need to be coordinated. Therefore to use Babel with this release of LaTeX you will need to update your version of babel to at least 3.7.

New input encoding `latin9`

The package inputenc has, thanks to Karsten Tinnefeld, been extended to cover the `latin9` input encoding. The ISO-Latin 9 encoding is a useful modern replacement for ISO-Latin 1 that contains a few characters needed for French and Finnish. Of wider interest, it also contains the euro currency sign; this could be the killer argument for many 8-bit texts to use Latin-9 in the future.

According to a Linux manpage, ISO Latin-9 supports Albanian, Basque, Breton, Catalan, Danish, Dutch, English, Estonian, Faroese, Finnish, French, Frisian, Galician, German, Greenlandic, Icelandic, Irish Gaelic, Italian, Latin, Luxembourgish, Norwegian, Portuguese, Rhaeto-Romanic, Scottish Gaelic, Spanish and Swedish. The characters added in `latin9` are (in LaTeX notation):

```
\texteuro  \v S  \v s  \v Z  \v z  \OE  \oe  \" Y
```

They displace the following characters from `latin1`:

```
\textcurrency  \textbrokenbar  \"{}  \'{}  \c{}
\textonequarter  \textonehalf  \textthreequarters
```

New tools

The new package trace provides many commands to control LaTeX's tracing and debugging output, including the excellent new information available with ε-TeX such as the extremely useful tracing of local assignments. You will find it in the tools distribution.

It offers the command `\traceon`, which is similar to `\tracingall` but suppresses uninteresting stuff such as font loading by NFSS (which can go on for pages if you are unlucky). It also offers `\traceoff` to ... guess what! Full details are in the documented source file, `trace.dtx`.

In the base ifthen package we have added the uppercase synonyms `\NOT` `\AND` and `\OR`.

New experimental code

In *LaTeX News 12* we announced some ongoing work towards a 'Designer Interface for LaTeX' and we presented some early results thereof. Since then, at Gutenberg 2000 in Toulouse and TUG 2000 in Oxford, we described a new output routine and an improved method of handling vertical mode material between paragraphs. In combination these support higher quality *automated*[1] page-breaking and page make-up for complex pages—the best yet achieved with TeX!

More recently we have added material to handle the complex front matter requirements of journal articles; this was presented at Gutenberg 2001 in Metz.

A paper describing the new output routine is at
`http://www.latex-project.org/papers/xo-pfloat.pdf`
All code examples and documentation are available at
`http://www.latex-project.org/code/experimental`

This directory has been extended to contain the following.

galley Prototype implementation of the interface for manipulating vertical material in galleys.

xinitials Prototype implementation of the interface for paragraph initials (needs the galley package).

xtheorem Contributed example using the template package to provide a designer interface for theorem environments.

xor A prototype implementation of the new output routine as described in the `xo-pfloat.pdf` paper.

xfrontm A prototype version of the new font matter interface.

[1] The stress here is on automated!

LaTeX News

Issue 15, December 2003 (LaTeX release 2003-12-01)

Anniversary release

Yes, it's now 10 years since the first release in this series and, for Knuthists, this release also contains *Issue 16* !

Meanwhile this *Issue 15* describes the major new features in the current release whilst *Issue 16* looks a little way into the future of LaTeX.

LPPL – new version

Most importantly, there is now a new version, 1.3, of the LaTeX Project Public Licence. Many of you will be thrilled to know that, following the exchange of over 1600 e-mail messages dissecting various aspects of its philosophy such as 'how many angels can appear in the name of a file before it becomes non-free', this version is now officially a DFSG (Debian Free Software Guidelines) approved license. The discussions start at http://lists.debian.org/debian-legal/2002/debian-legal-200207/threads.html with high traffic throughout August to October 2002 and further heated discussions starting in April 2003 and concluding around June at http://lists.debian.org/debian-legal/2003/debian-legal-200306/msg00206.html.

The important features of the new version are useful clarifications in the wording, and revised procedures for making a change to the Current Maintainer of a package. Special thanks to all those people from Debian Legal who worked constructively with us on this onerous task, especially but not exclusively Jeff Licquia and Branden Robinson.

Small updates to varioref

The English has been corrected in `\reftextbefore` (an incompatible change). There are other extensions such as `\labelformat`, `\Ref`, `\Vref` and `\vpagerefnum`. Some Dutch text has also been changed and two new options added: `slovak` and `slovene`.

New and more robust commands

Many of the math mode commands for compound symbols have been made robust and a new robust command has been added: `\nobreakdashes`. This last is a low-level command, borrowed from the amsmath package, for use only before hyphens or dashes. It prevents the line break that is normally allowed after the following sequence of dashes.

Fixing font sizes

The new fix-cm package, by Walter Schmidt, changes the CM font definition (`.fd`) files so that similar design sizes are used in both the `OT1` and `T1` encodings.

Font encodings

A number of options have been added to the textcomp package, enabling only available glyphs to be used. Also, the 'NFSS font families' are now divided into five different groups according to the subset of glyphs each provides from the full collection of symbols in the TS1 encoding. Given sufficient information about a font family textcomp will use this in order to limit the typesetting to those glyphs that are available.

Use of this mechanism has also enhanced `\oldstylenums` to use the current font if possible.

Displaying font tables

With the nfssfont package you can now specify the font to display by giving its 'NFSS classification', rather than needing to know its external font file's name. It is also now possible to generate large collections of font tables in batch mode by providing a suitable input file.

New input encodings

The inputenc package has been extended as follows: `macce` input encoding (Apple Central European), thanks to Radek Tryc and Marcin Wolinski; `cp1257` for Baltic languages; latin10, thanks to Ionel Ciobîcă. The euro symbol has by now been added to several encodings: `ansinew`, `cp1250` and `cp1252` (which also has another addition), whilst `cp858` adds it to `cp850`.

Unicode input

Partial, experimental support for text files that use the Unicode encoding form UTF-8 is now provided by the option `utf8` for the inputenc package.

The only Unicode text file characters supported by the current version are those based on the most common inputs for glyphs from the small collection of standard LaTeX Latin encodings.

And finally . . . pict2e

The old, non-functional version of this package has been removed as there is now a fully working version from Hubert Gäßlein and Rolf Niepraschk. It is described in *The LaTeX Manual*.

LaTeX News

Issue 16, December 2003 (LaTeX release 2003-12-01)

Anniversary news

This anniversary *Issue 16* takes a brief look into the future work of the LaTeX Project Team, both short and longer range. Please let us know if you want to get involved with us in any of this work (see below).

An overview of the 10th Anniversary Release, dated 2003/12/01, is can be found in *Issue 15*.

TLC2: The LaTeX Companion – 2nd edition!

Since you are reading this newsletter, there is a good chance that you, or a friend, has already bought this encyclopedic volume: the incomparable Second Edition of this work that is every LaTeXie's ultimate lucky charm.

If by some chance you have not yet purchased your own copy then get into training, get shopping, and get flexing your muscles (both physical—it's 1100+ pages, and intellectual) by using it to discover masses of invaluable 'insider information' about:

- the latest release of Standard LaTeX;
- over 200 extension packages;
- plus related software and systems.

For more information on this all new (??... OK, not *all*, but over 90%!!), all accurate (we hope!) 10th Anniversary Edition, check out http://www.awprofessional.com/titles/0201362996.

Future maintenance

We are currently exploring how best to support the very large and rapidly growing community of individuals, organisations and enterprises that depend on the robustness and availability of the current standard LaTeX distribution. Although we remain firmly resolved not to make changes in the base distribution (the kernel) of Standard LaTeX, there is still much that needs doing to maintain its reliability and utility and to keep up the necessary level of communication with users and supporters. Also, as with all advanced software systems, bugs are still turning up occasionally so some fixes are still essential.

One major impediment to providing adequate service levels in this area is, of course, the difficulties inherent in obtaining the time and commitment of skilled minds—hence the appeal above to anyone interested in getting involved.

LPPL certification

There are still some outstanding diplomatic tasks around the LaTeX Project Public Licence: these include e.g., getting it 'OSF certified' and ensuring that it gains more support and wider use, even in the FSF world where it has long been tolerated.

Use of ε-TeX/pdfTeX

We expect that within the next two years, releases of LaTeX will change modestly in order to run best under an extended TeX engine that contains the ε-TeX primitives, e.g., ε-TeX or pdfTeX. The details of this possible upgrade need further work so we are not making a definite announcement yet.

Although the current release does not *require* ε-TeX features, we certainly recommend using an extended TeX, especially if you need to debug macros.

End of 'autoload' support

As computer systems generally grow in capacity, requirements change and so we believe that the `autoload` variant of LaTeX is no longer required. Thus, although the code remains it is no longer supported. We hope this does not cause any problems.

New models, new code

In the period 1999–2001 we published many results of our work over the previous decade on the development of new concepts and models for automated typesetting based on TeX as the underlying platform. These can be found at http://www.latex-project.org/papers/ and http://www.latex-project.org/code/experimental/.

Since then a very large proportion of The Team's efforts have been diverted to provide the core author team for TLC2, which provides over 1000 pages of carefully researched and tested documentation of many aspects of the vast world of LaTeX related software that was developed over that same time period and that continues to grow and improve prodigiously.

Completion of that task ... until TLC3!! ... presents the possibility of getting back to this more exciting development work, or even to more radical work on non-TeX-based models and implementations.

Of course, any such ideas are predicated on our ability to organise (with you, we hope) an efficient but responsive maintenance and support system for Standard LaTeX.

LaTeX News

Issue 17, December 2005 (LaTeX release 2005-12-01)

Project licence news

The LaTeX Project Public License has been updated slightly so that it is now version 1.3c. In the warranty section the phrase "unless required by applicable law" has been reinstated, having got lost at some point. Also, it now contains three clarifications: of the difference between "maintained" and "author-maintained"; of the term "Base Interpreter"; and when clause 6b and 6d shall not apply.

Following requests, we now also provide the text of the licence as a LaTeX document (in the file `lppl.tex`). This file can be processed either as a stand-alone document or it can be included (without any modification) into another LaTeX document, e.g., as an appendix, using `\input` or `\include`.

New guide on font encodings

Way back in 1995 work was started on a guide to document the officially allocated LaTeX font encoding names. However, for one reason or another this guide (named *LaTeX font encodings*) was, until now, not added to the distribution. It describes the major 7-bit and 8-bit font encodings used in the LaTeX world and explains the restrictions required of conforming text font encodings. It also lists all the 'encoding specific commands' (the LICR or LaTeX Internal Character Representation) for characters supported by the encodings `OT1` and `T1`.

When the file `encguide.tex` is processed by LaTeX, it will attempt to typeset an encoding table for each encoding it describes. For this to be possible, LaTeX must be able to find `.tfm` files for a representative example font for each encoding. If LaTeX cannot find such a file then a warning is issued and the corresponding table is omitted.

Robust commands in math

The font changing commands in text-mode have been robust commands for years, but the same has not been true for the math versions such as `\mathbf`. While the math-mode commands worked correctly in section heads, they could cause problems in other places such as index entries. With this release, these math-mode commands are now robust in the same way as their text-mode counterparts.

Updates of required packages

Several of the packages in the tools bundle have been updated for this release.

The xspace package has some new features. One is an interface for adding and removing the exceptions it knows about and another is that it works with active characters. These remove problems of incompatibility with the babel system.

In *LaTeX News 16* we announced that some packages might begin to take advantage of ε-TeX extensions on systems where these are available: and the latest version of xspace does just that. Note also that fixltx2e will make use of the facilities in ε-TeX whenever these are present (see below).

The calc package has also been given an update with a few extra commands. The commands `\maxof` and `\minof`, each with two brace-delimited arguments, provide the usual numeric max and min operations. The commands `\settototalheight` and `\totalheightof` work like `\settoheight` and `\heightof`. There are also some internal improvements to make calc work with some more primitive TeX constructs, such as `\ifcase`.

The varioref package has acquired a few more default strings but there are still a number of languages for which good strings are still missing.

The showkeys package has also been updated slightly to work with more recent developments in varioref. Also, it now provides an easy way to define the look of the printed labels with the command `\showkeyslabelformat`.

Work on LaTeX fixes

The package known as fixltx2e has three new additions. A new command `\textsubscript` has been added as a complement to the command `\textsuperscript` in the kernel. Secondly, a new form of `\DeclareMathSizes` that allows all of its arguments to have a dimension suffix. This means you can now use expressions such as `\DeclareMathSizes{9.5dd}{9.5dd}{7.4dd}{6.6dd}`.

The third new addition is the robust command `\TextOrMath` which takes two arguments and executes one of them when typesetting in text or math mode respectively. This command also takes advantage of ε-TeX extensions if available; more specifically, when the ε-TeX extensions are available, it does not destroy kerning between previous letters and the text to be

typeset. The command is also used internally in fixltx2e to resolve a problem with `\fnsymbol`.

Also, further work has been done on reimplementing the command `\addpenalty`, which is used internally in several places: we hope it is an improvement!

The graphics bundle

The graphics bundle now supports the `dvipdfmx` post-processor and Jonathan Kew's XETEX program. By support we mean that the graphics packages recognize the new options `xetex` and `dvipdfmx` but we do not distribute the respective driver files.

This leads elegantly to a description of the new policy concerning such driver files in the graphics bundle. Most driver files for our graphics packages are maintained by the developers of the associated post-processor or TEX programs. The teams developing these packages are working very hard: their rapid development offers a stark contrast to the current schedule of LATEX releases. It is therefore no longer practical for the LATEX Team to be responsible for distributing the latest versions of these driver files.

Therefore the installation files for graphics have been split: there is now `graphics.ins` to install the package files and `graphics-drivers.ins` for the driver files (located in `drivers.dtx`). There is no need to install all those provided in the file `drivers.dtx`.

Please also note that, as requested by the maintainers of PStricks, we have removed the package pstcol as current versions of PSTricks make it obsolete.

Future development

The title of this section is a little misleading as it actually describes *current* development. In 1998 the expl3 bundle of packages was put on CTAN to demonstrate a possible LATEX3 programming environment. These packages have been lying dormant for some time while the LATEX Project Team were preoccupied by other things such as developing the experimental packages xor, template, etc., (and also writing that indispensable and encyclopaedic volume, The LATEX Companion – 2nd edition).

In October 2004 work on this code base was resumed with the goal of some day turning it into a kernel for LATEX3. This work can now also make full use of the widely accepted ε-TEX extensions. Currently two areas are central to this work.

- Extending the kernel code of LATEX3.
- Converting the experimental packages such as xor, template to use the new syntax internally.

Beware! Development of expl3 is happening so fast that the descriptions above might be out of date when you read this! If you wish to see what's going on then go to http://www.latex-project.org/code.html where you can download fully working code (we hope!).

LaTeX News

Issue 18, December 2007 (LaTeX release 2007-12-01)

This news never existed.

LaTeX News

Issue 19, September 2009 (LaTeX release 2009-09-24)

New LaTeX release

This issue of LaTeX News marks the first release of a new version of LaTeX 2ε since the publication of The LaTeX Companion in 2005–2006.

Just in time for TeX Live 2009, this version is a maintenance release and introduces no new features. A number of small changes have been made to correct minor bugs in the kernel, slightly extend the Unicode support, and improve various aspects of some of the tools packages.

New code repository

Since the last LaTeX release, the entire code base has been moved to a public SVN repository[1] and the entire build architecture re-written. In fact, it has only been possible for us to consider a new LaTeX release since earlier this year when the test suite was finally set up with the new system. In the process, a bug in the LaTeX picture fonts distributed with TeX Live was discovered, proving that the tests are working and are still very valuable.

Now that we can easily generate new packaged versions of the LaTeX 2ε distribution, we expect to be able to roll out bug fixes in a much more timely manner than over the last few years. New versions should be distributed yearly with TeX Live. Having said this, the maintenance of the LaTeX 2ε kernel is slowing down as the bugs become fewer and more subtle. Remember that we cannot change any of the underlying architecture of the kernel or any design decisions of the standard classes because we must preserve backwards compatibility with legacy documents at all costs.

Even new features cannot be added, because any new documents using them will not compile in systems (such as journal production engines) that are generally not updated once they've been proven to work as necessary.

None of this is to say that we consider LaTeX 2ε to be any less relevant for document production than in years past: a stable system is a useful one. Moreover, the package system continues to provide a flourishing and stable means for the development of a wide range of extensions.

Babel

One area of the LaTeX 2ε code base that is still receiving feedback to be incorporated into the main distribution is the Babel system for multilingual typesetting. While the Babel sources have already been added to the SVN repository the integration of the test system for Babel is still outstanding.

The future

While work on LaTeX 2ε tends to maintenance over active development, the LaTeX3 project is seeing new life. Our goals here are to provide a transition from the LaTeX 2ε document processing model to one with a more flexible foundation. Work is continuing in the expl3 programming language and the xpackages for document design. Future announcements about LaTeX3 will be available via the LaTeX Project website and in TUGboat.

[1] http://www.latex-project.org/svnroot/latex2e-public/

LaTeX News

Issue 20, June 2011 (LaTeX release 2011-06-27)

Scheduled LaTeX bug-fix release

This issue of LaTeX News marks the first bug-fix release of LaTeX 2ε since shifting to a new build system in 2009. Provided sufficient changes are made each year, we expect to repeat such releases once per year to stay in sync with TeX Live. Due to the excitement of TeX's 2^5-th birthday last year, we missed our window of opportunity to do so for 2010. This situation has been rectified this year!

Continued development

The LaTeX 2ε program is no longer being actively developed, as any non-negligible changes now could have dramatic backwards compatibility issues with old documents. Similarly, new features cannot be added to the kernel since any new documents written now would then be incompatible with legacy versions of LaTeX.

The situation on the package level is quite different though. While most of us have stopped developing packages for LaTeX 2ε there are many contributing developers that continue to enrich LaTeX 2ε by providing or extending add-on packages with new or better functionality.

However, the LaTeX team certainly recognises that there are improvements to be made to the kernel code; over the last few years we have been working on building, expanding, and solidifying the expl3 programming layer for future LaTeX development. We are using expl3 to build new interfaces for package development and tools for document design. Progress here is continuing.

Release notes

In addition to a few small documentation fixes, the following changes have been made to the LaTeX 2ε code; in accordance with the philosophy of minimising forwards and backwards compatibility problems, most of these will not be noticeable to the regular LaTeX user.

Font subsets covered by Latin Modern and TeX Gyre The Latin Modern and TeX Gyre fonts are a modern suite of families based on the well-known Computer Modern and 'PostScript 16' families with many additional characters for high-quality multilingual typesetting.[1]

[1] See their respective TUGboat articles for more information: http://www.tug.org/TUGboat/tb24-1/jackowski.pdf

Information about their symbol coverage in the `TS1` encoding is now included in textcomp's default font definitions.

Redefinition of `\enddocument` Inside the definition of `\end{document}` the `.aux` file is read back in to resolve cross-references and build the table of contents etc. From 2.09 days this was done using `\input` without any surrounding braces which could lead to some issues in boundary cases, especially if `\input` was redefined by some package. It was therefore changed to use LaTeX 2ε's internal name for this function. As a result, packages that modify `\enddocument` other than through the officially provided hooks may need to get updated.

Small improvement with split footnotes in ftnright If in the first column there is more than a full column worth of footnote material the material will be split resulting in footnotes out of order. This issue is now at least detected and generates an error but the algorithm used by the package is unable to gracefully handle it in an automated fashion (some alternatives for resolving the problem if it happens are given in the package documentation).

Improvement in xspace and font-switching The `xspace` package provides the command `\xspace` which attempts to be clever about inserting spaces automatically after user-defined control sequences. An important bug fix has been made to this command to correct its behaviour when used in conjunction with font-switching commands such as `\emph` and `\textbf`. Previously, writing

```
\newcommand\foo{foo\xspace}
... \emph{\foo}  bar baz
... \emph{\foo}, bar baz
```

would result in an extraneous space being inserted after 'foo' in both cases; this has now been corrected.

RTL in `multicol` The 1.7 release of `multicol` adds support for languages that are typeset right-to-left. For those languages the order of the columns on the page also needs to be reversed—something that wasn't possible in earlier releases.

http://www.tug.org/TUGboat/tb27-2/tb87hagen-gyre.pdf

The new feature is supported through the commands `\RLmulticolcolumns` (switching to right-to-left typesetting) and `\LRmulticolcolumns` (switching to left-to-right typesetting) the latter being the default.

Improve French `babel` interaction with `varioref`
Extracting and saving the page number turned out to be a source of subtle bugs. Initially it was done through an `\edef` with a bunch of `\expandafter` commands inside. This posed a problem if the page number itself contained code which needed protection (e.g., pr/4080) so this got changed in the last release to use `\protected@edef`. However, that in turn failed with Babel (bug report/4093) if the label contained active characters, e.g., a ":" in French. So now we use (after one failed attempt pr/4159) even more `\expandafter` commands and `\romannumeral` trickery to avoid any expansion other than what is absolutely required—making the code in that space absolutely unreadable.

```
\expandafter\def\expandafter#1\expandafter{%
\romannumeral
  \expandafter\expandafter\expandafter
\z@
\expandafter \@cdr
\romannumeral
  \expandafter\expandafter\expandafter
\z@
\csname r@#2\endcsname\@nil}%
```

Code like this nicely demonstrates the limitations in the programming layer of LaTeX 2_ε and the advantages that expl3 will offer on this level.

LaTeX News

Issue 21, May 2014 (LaTeX release 2014-05-01)

Scheduled LaTeX bug-fix release

This issue of LaTeX News marks the second bug-fix release of LaTeX 2ε (standard LaTeX) since shifting to a new build system in 2009. Provided sufficient changes are made, we expect to make such releases yearly or every two years, in sync with TeX Live.

Release notes

This release makes no changes to the core code in the LaTeX 2ε format but there are a small number of documentation fixes (not listed here). In addition several packages in the base and required areas have been updated as detailed below.

This has been done in accordance with the philosophy of minimising problems in both forwards and backwards compatibility, so most of these changes should not be noticed by the regular LaTeX user.

References in the text below of the form "graphics/3873" are to bug reports listed at: http://latex-project.org/cgi-bin/ltxbugs2html

fixltx2e updates

There are a number of bugs and faulty design decisions in LaTeX 2ε that should have been corrected long ago in the kernel code. However, such corrections cannot be done as this would break backwards compatibility in the following sense. A large number of documents exist by now that have worked around the bug or have even made use of a particular misfeature. Thus changing the kernel code would break too many existing documents.

The corrections for these types of bug have therefore been collected together in a package that can be loaded only when needed; its name is fixltx2e. For this release we made the following changes to this package:

- Misspelled float placement specifiers such as `\begin{figure}[tv]` instead of `tb` are silently ignored by the kernel code. Now we test for such letters and issue an error message.
- LaTeX's float handling algorithm can get out of sync if you mix single and double-column floats (as they are placed independently of each other). This was corrected in fixltx2e a few years ago but the fix was not perfect as one situation using `\enlargethispage` generated a low-level TeX error. This behaviour of the package is now improved.

New fltrace package

For years the file `ltoutput.dtx` contained some hidden code to trace the detailed behaviour of the float placement algorithm of LaTeX. Prompted by questions on StackExchange we now extract this code into a new fltrace package. To see the float algorithm in action (or to understand why it decides to place all your floats at the very end of the document) use

```
\usepackage{fltrace} \tracefloats
```

To stop tracing somewhere in the document use `\tracefloatsoff` and to see the current value of various float parameters use `\tracefloatvals`. As the package is identical to the kernel code with tracing added, it may or may not work if you load any other package that manipulates that part of the kernel code. In such a case your best bet is to load fltrace first.

inputenc package updates

The inputenc package allows different input encodings for LaTeX documents to be specified including the important `utf8` option used to specify the Unicode UTF-8 encoding. A common mistake in documents has been to also include this option when using the Unicode-based TeX engines LuaTeX and XeTeX producing strange errors as these engines natively deal with UTF-8 characters.

If a document stored in an 8bit encoding is processed by pdfTeX, it needs the inputenc package to work correctly. However, if such a document is processed unchanged by LuaTeX or XeTeX, then accented characters may silently get dropped from the output.

The package has been modified so that if used with LuaTeX or XeTeX, then it just issues a warning if `utf8` or `ascii` is specified, and stops with an error for any other encoding requested.

One further improvement has been made to the encoding definition files (`.def`) used by inputenc: the catcode of `@` is now saved and restored when reading them instead of always using `\makeatother` inside the files (latex/4192).

The tools directory

In the past each of the sub-directories in the "required" section of the LaTeX distribution contained a single `.ins` file to generate the code files from the source files. We have now started to provide individual `.ins` files for each of those packages that are likely to require updates outside a major LaTeX release.

multicol updates

Version 1.8 of multicol implements some improvements/fixes and one extension. In the past the balancing algorithm enlarged the column height until it found a solution that satisfied all constraints. If there were insufficient break points then the final column height could have been much larger than expected and if that happened near the end of the page it resulted in the text overflowing into the bottom margin. This situation is now detected and in that case a normal page is cut and balancing is resumed on the next page. Some overflow is still allowed and controlled via the parameter `\maxbalancingoverflow`.

The use of `\enlargethispage` is now properly supported within the environment. Finally a new command `\docolaction` was added to allow the execution of code depending on the column in which the command is executed. See the documentation for details.

Bug fixes: the new version fixes both a color leak that could happen in certain situations and the problem that multicols could mess up the positioning of `\marginpar`s that followed the environment.

tabularx updates

The restrictions on embedding `\tabularx` `\endtabularx` into the definition of a new environment have been relaxed slightly. See the package documentation for details.

showkeys updates

The showkeys package has been updated to fix problems if used at the start of list items, and to work if brace groups (`{` and `}`) are used in the optional argument of `\cite`. (tools/4162, tools/4173)

color updates

The `\nopagecolor` command suggested by Heiko Oberdiek, available for some years in the `pdftex` option, has been added to the core package as suggested in graphics/3873. Currently this is supported in the driver files for dvips and pdftex. Patches to support other drivers are welcome.

graphicx updates

The graphicx version of `\rotatebox` now allows `\par` (and blank lines) in values, to match the change made to the graphics version some years ago. See graphics/4296.

keyval updates

All parsing used in the keyval package has been changed to allow `\par` (and blank lines) in values. (A second change, to parsing of brace groups in a construct such as `key={{{value}}}`, was reverted in v1.15.) See graphics/3446.

Standard LaTeX (LaTeX 2ε) and expl3

The substantial collection of innovative code in expl3 implements a new programming language that has for a while now been used by some writers of LaTeX 2ε packages. This code has recently also been made available for use on top of plain TeX or ConTeXt, largely to support generic packages that are supposed to work with different flavours of TeX. These uses in no way affect authors of LaTeX documents and such LaTeX 2ε packages will continue to work as advertised by their authors with standard LaTeX.

This code base will also become an important foundation for the kernel of LaTeX3 and so the new programming language can be described as 'The LaTeX3 Programming Language'. However, if you see or hear that a package 'uses LaTeX3' then it remains very unlikely (as yet) to mean that the package is part of some 'new version of LaTeX'.

News about the development and use of expl3 and about other developments in the LaTeX3 code base is reported regularly in the LaTeX3 News series (http://latex-project.org/l3news/), the most recent issue of which was published in March 2014.

LaTeX News

Issue 22, January 2015 (LaTeX release 2015-01-01)

New LaTeX 2ε bug-fix policy

Introduction

For some years we have supplied bug fixes to the LaTeX 2ε kernel via the fixltx2e package. This kept the kernel stable, but at the expense of meaning that most users did not benefit from bug fixes, and that some compromises which were made to save space in the machines of the time are still affecting most users today.

In this release we have started a new update policy. All the fixes previously available via fixltx2e are now enabled *by default* in the format, as are some further extensions for extended TeX engines, ε-TeX, XeTeX and LuaTeX. Compatibility and stability are still important considerations, and while most users will not notice these improvements, or will want to benefit from them, a new latexrelease package is provided that will revert all the changes and re-instate the definitions from earlier releases. The package can also be used with older releases to effectively *update* the kernel to be equivalent to this 2015 release.

A new document, `latexchanges`, is distributed with the release that documents all the changes to documented commands since the 2014 LaTeX release, and will be updated in future releases if further changes have been made.

The latexrelease package

As noted above a new package is available to manage differences between LaTeX releases. If you wish to revert all changes back to the definitions as they were in previous releases you may start your document requesting the LaTeX release from May 2014:

```
\RequirePackage[2014/05/01]{latexrelease}
\documentclass{article}
```

Conversely if you start a large project now and want to protect yourself against possible future changes, you may start your document

```
\RequirePackage[2015/01/01]{latexrelease}
\documentclass{article}
```

Then the version of latexrelease distributed with any future LaTeX release will revert any changes made in that format, and revert to the definitions as they where at the beginning of 2015.

If you wish to share a document using the latest features with a user restricted to using an older format, you may use the form above and make the latexrelease package available on the older installation. The package will then update the format definitions as needed to enable the older format to work as if dated on the date specified in the package option.

The `\IncludeInRelease` command

The mechanism used in the latexrelease package is available for use in package code. If in your zzz package you have

```
\RequirePackage{latexrelease}
\IncludeInRelease{2015/06/01}
    {\zzz}{\zzz definition}
  \def\zzz......new code
\EndIncludeInRelease
\IncludeInRelease{0000/00/00}
  {\zzz}{\zzz definition}
  \def\zzz....original
\EndIncludeInRelease
```

then in a document using a format dated 2015/06/01 or later, the "new code" will be used, and for documents being processed with an older format, the "original" code will be used. Note the format date here may be the original format date as shown at the start of every LaTeX run, or a format date specified as a package option to the latexrelease package.

So if the document has

```
\RequirePackage[2014/05/01]{latexrelease}
\documentclass{article}
\usepackage{zzz}
```

then it will use the *original* definition of `\zzz` even if processed with the current format, as the format acts as if dated 2014/05/01.

Limitations of the approach

The new concept provides full backward and forward compatibility for the LaTeX format, i.e., with the help of a current latexrelease package the kernel can emulate all released formats (starting with 2014/06/01[1]).

However, this is not necessarily true for all packages. Only if a package makes use of the `\IncludeInRelease` functionality will it adjust to the requested LaTeX release date. Initially this will only be true for a few selected packages and in general it may not even be

[1] Patching an older format most likely works too, given that the changes in the past have been minimal, though this isn't guaranteed and hasn't been tested.

advisable for packages that have their own well-established release cycles and methods.

Thus, to regenerate a document with 100 % compatible behavior it will still be necessary to archive it together with all its inputs, for example, by archiving the base distribution trees (and any modifications made). However, the fact that a document requests a specific LaTeX release date should help identifying what release tree to use to achieve perfect accuracy.

Updates to the kernel

Updates incorporated from fixltx2e

The detailed list of changes incorporated from fixltx2e is available in the new `latexchanges` document that is distributed with this release. The main changes are that 2-column floats are kept in sequence with one column floats, corrections are made to the `\mark` system to ensure correct page headings in 2-column documents, several additional commands are made robust.

ε-TeX register allocation

LaTeX has traditionally used allocation routines inherited from plain TeX that allocated registers in the range 0–255. Almost all distributions have for some years used ε-TeX based formats (or XeTeX or LuaTeX) which have 2^{15} registers of each type (2^{16} in the case of LuaTeX). The etex package has been available to provided an allocation mechanism for these extended registers but now the format will by default allocate in a range suitable for the engine being used. The new allocation mechanism is different than the etex package mechanism, and supports LuaTeX's full range and an allocation mechanism for LaTeX floats as described below.

On ε-TeX based engines, an additional command, `\newmarks` is available (as with the etex package) that allocates extended ε-TeX marks, and similarly if XeTeX is detected a new command `\newXeTeXintercharclass` is available, this is similar to the command previously defined in the `xelatex.ini` file used to build the `xelatex` format.

Additional LaTeX float storage

LaTeX's float placement algorithm needs to store floats (figures and tables) until it finds a suitable page to output them. It allocates 18 registers for this storage, but this can often be insufficient. The contributed morefloats package has been available to extend this list; however, it also only allocates from the standard range 0–255 so cannot take advantage of the extended registers. The new allocation mechanism in this release incorporates a new command `\extrafloats`. If you get the error: `Too many unprocessed floats.` then you can add (say) `\extrafloats{500}` to the document preamble to make many more boxes available to hold floats.

Built-in support for Unicode engines

The kernel sources now detect the engine being used and adjust definitions accordingly, this reduces the need for the "`.ini`" files used to make the formats to patch definitions defined in `latex.ltx`.

As noted above the format now includes extended allocation routines.

The distribution includes a file `unicode-letters.def` derived from the Unicode Consortium's Unicode Character Data files that details the upper and lower case transformation data for the full Unicode range. This is used to set the `lccode` and `uccode` values if a Unicode engine is being used, rather than the values derived from the T1 font encoding which are used with 8-bit engines.

Finally `\typein` is modified if LuaTeX is detected such that it works with this engine.

l3build

This release has been tested and built using a new build system implemented in Lua, intended to be run on the texlua interpreter distributed with modern TeX distributions. It is already separately available from CTAN. This replaces earlier build systems (based at various times on make, cons, and Windows bat files). It allows the sources to be tested and packaged on a range of platforms (within the team, OS X, Windows, Linux and Cygwin platforms are used). It also allows the format to be tested on XeTeX and LuaTeX as well as the standard pdfTeX/ε-TeX engines.

Hyperlinked documentation and TDS zip files

As well as updating the build system, the team have looked again at exactly what gets released to CTAN. Taking inspiration from Heiko Oberdiek's latex-tds bundle, the PDF documentation provided now includes hyperlinks where appropriate. This has been done without modifying the sources such that users without hyperref available can still typeset the documentation using only the core distribution. At the same time, the release now includes ready-to-install TDS-style zip files. This will be of principal interest to TeX system maintainers, but end users with older machines who wish to manually update LaTeX will also benefit.

LaTeX News

Issue 23, October 2015 (LaTeX release 2015-10-01)

Contents

Enhanced support for LuaTeX

As noted in LaTeX News 22, the 2015/01/01 release of LaTeX introduced built-in support for extended TeX systems.

The range of allocated register numbers (for example, for count registers) is now set according to the underlying engine capabilities to 256, 32768 or 65536. Additional allocators were also added for the facilities added by ε-TeX (`\newmark`) and XeTeX (`\newXeTeXintercharclass`). At that time, however, the work to incorporate additional allocators for LuaTeX was not ready for distribution.

The main feature of this release is that by default it includes allocators for LuaTeX-provided features, such as Lua functions, bytecode registers, catcode tables and Lua callbacks. Previously these features have been provided by the contributed luatex (Heiko Oberdiek) and luatexbase (Élie Roux, Manuel Pégourié-Gonnard and Philipp Gesang) packages. However, just as noted with the etex package in the previous release, it is better if allocation is handled by the format to avoid problems with conflicts between different allocation schemes, or definitions made before a package-defined allocation scheme is enabled.

The facilities incorporated into the format with this release, and described below, are closely modelled on the luatexbase package and we thank the authors, and especially Élie Roux, for help in arranging this transition.

The implementation of these LuaTeX features has been redesigned to match the allocation system introduced in the 2015/01/01 LaTeX release, and there are some other differences from the previous luatexbase package. However, as noted below, luatexbase is being updated in line with this LaTeX release to provide the previous interface as a wrapper around the new implementation, so we expect the majority of documents using luatexbase to work without change.

Names of LuaTeX primitive commands

The 2015/01/01 LaTeX release for the first time initialised LuaTeX in `latex.ltx` if LuaTeX is being used. Following the convention used in the contributed `lualatex.ini` file used to set up the format for earlier releases, most LuaTeX-specific primitives were defined with names prefixed by `luatex`. This was designed to minimize name clashes but had the disadvantage that names did not match the LuaTeX manual, or the names used in other formats, and produced some awkward command names such as `\luatexluafunction`. From this release the names are enabled without the `luatex` prefix.

In practice this change should not affect many documents; relatively few packages access the primitive commands, and many of those are already set up to work with prefixed or unprefixed names, so that they work with multiple formats.

For package writers, if you want to ensure that your code works with this and earlier releases, use unprefixed names in the package and ensure that they are defined by using code such as:

```
\directlua{tex.enableprimitives("",
             tex.extraprimitives(
               "omega", "aleph", "luatex"))}
```

Conversely if your document uses a package relying on prefixed names then you can add:

```
\directlua{tex.enableprimitives("luatex",
             tex.extraprimitives(
               "omega", "aleph", "luatex"))}
```

to your document.

Note the compatibility layer offered by the luatexbase package described below makes several commands available under both names.

As always, this change can be reverted using:
`\RequirePackage[2015/01/01]{latexrelease}`
at the start of the document.

TEX commands for allocation in LuaTEX

For detailed descriptions of the new allocation commands see the documented sources in `ltluatex.dtx` or chapter N of `source2e`; however, the following new allocation commands are defined by default in LuaTEX: `\newattribute`, `\newcatcodetable`, `\newluafunction` and `\newwhatsit`. In addition, the commands `\setattribute` and `\unsetattribute` are defined to set and unset Lua attributes (integer values similar to counters, but attached to nodes). Finally several catcode tables are predefined: `\catcodetable@initex`, `\catcodetable@string`, `\catcodetable@latex`, `\catcodetable@atletter`.

Predefined Lua functions

If used with LuaTEX, LATEX will initialise a Lua table, `luatexbase`, with functions supporting allocation and also the registering of Lua callback functions.

Support for older releases and plain TEX

The LuaTEX allocation functionality made available in this release is also available in plain TEX and older LATEX releases in the files `ltluatex.tex` and `ltluatex.lua` which may be used simply by including the TEX file: `\input{ltluatex}`. An alternative for old LATEX releases is to use:
`\RequirePackage[2015/10/01]{latexrelease}`
which will update the kernel to the current release, including LuaTEX support.

Additional LuaTEX support packages

In addition to the base LATEX release two packages have been contributed to the contrib area on CTAN. The ctablestack package offers some commands to help package writers control the LuaTEX `catcodetable` functionality, and the luatexbase package replaces the previously available package of the same name, providing a compatible interface but implemented over the ltluatex code.

More Floats and Inserts

If ε-TEX is available, the number of registers allocated in the format to hold floats such as figures is increased from 18 to 52.

The extended allocation system introduced in 2015/01/01 means that in most cases it is no longer necessary to load the etex package. Many classes and packages that previously loaded this package no longer do so. Unfortunately in some circumstances where a package or class previously used the etex `\reserveinserts` command, it is possible for a document that previously worked to generate an error "no room for a new insert". In practice this error can always be avoided by declaring inserts earlier, before the registers below 256 are all allocated. However, it is better not to require packages to be re-ordered and in some cases the re-ordering is complicated due to delayed allocations in `\AtBeginDocument`.

In this release, a new implementation of `\newinsert` is used which allocates inserts from the previously allocated float lists once the classical register allocation has run out. This allows an extra 52 (or in LuaTEX, 64 thousand) insert allocations which is more than enough for practical documents (by default, LATEX only uses two insert allocations).

Updated Unicode data

The file `unicode-letters.def` recording catcodes, upper and lower case mappings and other properties for Unicode characters has been regenerated using the data files from Unicode 8.0.0.

Support for Comma Accent

The command `\textcommabelow` has been added to the format. This is mainly used for the Romanian letters Ș ș Ț ț. This was requested in latex/4414 in the LATEX bug tracker.

Extended inputenc

The `utf8` option for inputenc has been extended to support the letters s and t with comma accent, U+0218 – U+021b. Similarly circumflex w and y U+0174 – U+0177 are defined. Also U+00a0 and U+00ad are declared by default, and defined to be `\nobreakspace` and `\-` respectively.

The error message given on undefined UTF-8 input characters now displays the Unicode number in U+*hex* format in addition to showing the character.

Pre-release Releases

The patch level mechanism has been used previously to identify LATEX releases that have small patches applied to the main release, without changing the main format date.

The mechanism has now been extended to allow identification of pre-release versions of the software (which may or may not be released via CTAN) but can be identified with a banner such as
`LaTeX2e <2015/10/01> pre-release-1`
Internally this is identified as a patch release with a negative patch level.

Updates in tools

The multicol package has been updated to fix the interaction with "here" floats that land on the same page as the start or end of a `multicols` environment.

LaTeX News

Issue 24, February 2016 (LaTeX release 2016-02-01)

Contents

LuaTeX support

This release refines the LuaTeX support introduced in the 2015/10/01 release. A number of patches have been added to improve the behavior of ltluatex (thanks largely to code review by Philipp Gesang). The kernel code has been adjusted to allow for changes in LuaTeX v0.85–v0.88. Most notably, newer LuaTeX releases allow more than 16 write streams and these are now enabled for use by `\newwrite`, but also the experimental `newtoken` Lua library has been renamed back to `token` which required small adjustments in the LuaTeX setup.

The biggest change in LuaTeX v0.85–v0.87 compared to previous versions is that all the primitives (originally defined in pdfTeX) dealing with the PDF "back end" are no longer defined, being replaced by a much smaller set of new primitives. This does not directly affect the core LaTeX files in this release but has required major changes to the `.ini` files used by TeX Live and similar distributions to set up the format files. These changes in the LuaTeX engine will affect any packages using these back end commands (packages such as graphics, color, hyperref, etc.). Until all contributed packages are updated to the new syntax users may need to add aliases for the old pdfTeX commands. A new luapdftexalias package has been contributed to CTAN (not part of the core LaTeX release) that may be used for this purpose.

See also the sections below for related changes in the tools and graphics bundles.

Unicode data

As noted in LaTeX News 22, the 2015/01/01 release of LaTeX introduced built-in support for extended TeX systems. In particular, the kernel now loads appropriate data from the Unicode Consortium to set `\lccode`, `\uccode`, `\catcode` and `\sfcode` values in an automated fashion for the entire Unicode range.

The initial approach taken by the team was to incorporate the existing model used by (plain) XeTeX and to pre-process the "raw" Unicode data into a ready-to-use form as `unicode-letters.def`. However, the relationship between the Unicode Consortium files and TeX data structures is non-trivial and still being explored. As such, it is preferable to directly parse the original (`.txt`) files at point of use. The team has therefore "spun-out" both the data and the loading to a new generic package, unicode-data. This package makes the original Unicode Consortium data files available in the `texmf` tree (in `tex/generic/unicode-data`) and provides generic loaders suitable for reading this data into the plain, LaTeX 2_ε, and other, formats.

At present, the following data files are included in this new package:

- `CaseFolding.txt`
- `EastAsianWidth.txt`
- `LineBreak.txt`
- `MathClass.txt`
- `SpecialCasing.txt`
- `UnicodeData.txt`

These files are used either by LaTeX 2_ε or by expl3 (i.e. they represent the set currently required by the team). The Unicode Consortium provides various other data files and we would be happy to add these to the generic package, as it is intended to provide a single place to collect this material in the `texmf` tree. Such requests can be mailed to the team as usual or logged at the package home page: https://github.com/latex3/unicode-data.

The new approach extends use of Unicode data in setting TeX information in two ways. First, the `\sfcode` of all end-of-quotation/closing punctuation is now set to 0 (transparent to TeX). Second, `\Umathcode` values are now set using `MathClass.txt` rather than setting up only letters (which was done using an arbitrary plane 0/plane 1 separation). There are also minor refinements to the existing code setting, particularly splitting the concepts of case and letter/non-letter category codes.

For XeTeX, users should note that `\xtxHanGlue` and `\xtxHanSpace` are *no longer defined*, that no

assignments are made to `\XeTeXinterchartoks` and that no `\XeTeXintercharclass` data is loaded into the format. The values which were previously inherited from the plain XeTeX setup files are *not* suitable for properly typesetting East Asian text. There are third-party packages addressing this area well, notably those in the CTeX bundle. Third-party packages may need adjustment to load the data themselves; see the unicode-data package for one possible loader.

More support for east European accents

As noted in LaTeX News 23, comma accent support was added for `s` and `t` in the 2015/10/01 release. In this release a matching `\textcommaabove` accent has been added for U+0123 (`\c{g}`, ģ) which is the lower case of U+0122 (`\c{G}`, Ģ). In the OT1 and T1 encodings the combinations are declared as composites with the `\c` command, which matches the Unicode names "latin (capital|small) letter g with cedilla" and also allows `\MakeUppercase{\c{g}}` to produce `\c{G}`, as required. In T1 encoding, the composite of `\c` with `k`, `l`, `n` and `r` are also declared to use the comma below accent rather than cedilla to match the conventional use of these letters.

The UTF-8 inputenc option `utf8` has been extended to support all latin combinations that can be reasonably constructed with a (single) accent command an a base character for the T1 encoding so ģ, ų and similar characters may be directly input using UTF-8 encoding.

Changes in Graphics

The changes in LuaTeX v0.87 mean that the color and graphics packages no longer share the `pdftex.def` file between LuaTeX and pdfTeX. A separate file `luatex.def` (distributed separately) has been produced, and distributions are encouraged to modify `graphics.cfg` and `color.cfg` configuration files to default to the `luatex` option if LuaTeX v0.87 or later is being used. The team has contributed suitable `.cfg` files to CTAN to be used as models.

Normally it is best to let the local `graphics.cfg` automatically supply the right option depending on the TeX engine being used; however the color and graphics (and so graphicx) packages have been extended to have an explicit `luatex` option comparable to the existing `pdftex` and `xetex` options.

The trig package has been updated so that pre-computed values such as sin(90) now expand to digits (`1` rather than the internal token `\@one` in this case). This allows them to be used directly in PDF literal strings.

Changes in Tools

LuaTeX from version v0.87 no longer supports the `\write18` syntax to access system commands. A new package shellesc has been added to tools that defines a new command `\ShellEscape` that may be used in all TeX variants to provide a consistent access to system commands. The package also defines `\write18` in LuaTeX so that it continues to access system commands as before; see the package documentation for details.

Improving support for Unicode engines

Stability concerns are always paramount when considering any change to the LaTeX 2_ε kernel. At the same time, it is important that the format remains usable and gives reliable results for users. For the Unicode TeX engines XeTeX and LuaTeX there are important differences in behavior from classical (8-bit) TeX engines which mean that identical default behaviors are not appropriate. Over the past 18 months the team has addressed the most pressing of these considerations (as detailed above and in LaTeX News 22 and 23), primarily by integrating existing patches into the kernel. There are, though, important areas which still need consideration, and which *may* result in refinements to kernel support in this area in future releases.

The default font setup in LaTeX 2_ε at present is to use the `OT1` encoding. This assumes that hyphenation patterns have been read using appropriate codes: the `T1` encoding is assumed. The commonly-used hyphenation patterns today, `hyph-utf8`, are set up in this way for 8-bit engines (pdfTeX) but for Unicode engines use Unicode code points. This means that hyphenation will be incorrect with Unicode engines unless a Unicode font is loaded. This requires a concept of a Unicode font encoding, which is currently provided by the fontspec package in two versions, `EU1` and `EU2`. The team is working to fully understand what is meant by a "Unicode font encoding", as unlike a classical TeX encoding it is essentially impossible to know what glyphs will be provided (though each slot is always defined with the same meaning). There is also an overlap between this area and ideas of language and writing system, most obviously in documents featuring mixed scripts (for example Latin and Cyrillic).

As well as these font considerations, the team is also exploring to what extent it is possible to allow existing (8-bit) documents to compile directly with Unicode engines without requiring changes in the sources. Whether this is truly possible remains an open question.

It is important to stress that changes will only be made in this area where they do *not* affect documents processed with ε-TeX/pdfTeX (i.e. documents which are written for "classical" 8-bit TeX engines). Changes will also be made only where they clearly address deficiencies in the current setup for Unicode engines (i.e. where current behaviors are wrong).

LaTeX News

Issue 25, March 2016 (LaTeX release 2016-03-01)

LuaTeX

This LaTeX release sees several internal changes designed to ensure that the system is still usable with LuaTeX versions greater than 0.80, which have introduced many changes into the engine, most notably the removal or renaming of most of the primitive commands introduced by pdfTeX. Also the lists of Lua callbacks handled by the callback allocation mechanism has been updated to match the callbacks defined in LuaTeX version 0.90.

These changes have also required updates in tools and amsmath as described below.

This is the first release of LaTeX for which the test suite reports no failures when used with LuaTeX.

Documentation checksums

The doc package has always provided two mechanisms that were mainly intended to guard against file truncation or corruption when files were commonly distributed by email through unreliable mail gateways: a Character Table of the ASCII character set could be inserted (and checked) and a "checksum" (count of the number of backslashes in the code sections) could be checked. These features are not really needed with modern distribution mechanisms and can be a distraction when reading the source code and so have been removed. The doc package has been updated so that if you use a `\CheckSum` command then, as before, the number is checked; however, if you omit the command then no error or warning is given.

Updates to inputenc

The UTF-8 support in inputenc has been further extended with support for non-breaking hyphens and more dashes.

Updates in Tools

The varioref package has been updated with improved documentation of multilingual support, and avoiding unnecessary warnings in some cases with `\reftextfaraway`.

The tabularx package's handling of `\endtabularx` in environment definitions has been fixed to again match its documentation.

The bm package has been updated as required by the changes to `\mathchardef` in LuaTeX.

amsmath

Since the launch of LaTeX 2ε in 1993, the amsmath bundle has been part of the *required* packages in the core LaTeX distribution, with bug reports handled by the LaTeX bug database at https://latex-project.org/bugs-upload.html.

The amsmath packages and the `amscls` classes have been maintained by the American Mathematical Society.

With this release a new arrangement has been agreed between the American Mathematical Society and the LaTeX3 project. The LaTeX3 project will take over maintenance of the amsmath bundle, with the American Mathematical Society retaining maintenance of `amscls`.

The recommended installation of these files in the TeX directory structure remains unchanged as `tex/latex/amsmath` and `tex/latex/amscls` respectively.

This release of amsmath includes several updates so that amsmath does not generate errors when math is used with LuaTeX v0.87+, which has changes to `\mathchardef` that are incompatible with the previous version of amsmath. It also improves `\dots` handling so that `\long` macros are correctly handled (for example, `\dots \Rightarrow` now uses centered dots), as well as commands expanding to character tokens (for example, `\times \dots \times` will use centered dots with `\times` defined as in the unicode-math package).

Related updates

In addition to the updates in the core LaTeX release, some files in the CTAN "contrib" area have also been updated. Notably there have been further updates to the `unicode-data` files; also, the files required to build plain and LaTeX formats have now been submitted to CTAN as `tex-ini-files`. The addition of a new `luatex` option for graphics-related packages (`luatex-def` on CTAN) has required updates to the configuration files to select a default option and these have similarly been uploaded to CTAN as `graphics-cfg`. (Previously these files were maintained directly in the TeX Live repository, and were not available on CTAN.)

LaTeX News

Issue 26, January 2017 (LaTeX release 2017-01-01)

Contents

ε-TeX

In LaTeX News 16 (December 2003) the team announced

> We expect that within the next two years, releases of LaTeX will change modestly in order to run best under an extended TeX engine that contains the ε-TeX primitives, e.g., ε-TeX or pdfTeX.

and also said

> Although the current release does not *require* ε-TeX features, we certainly recommend using an extended TeX, especially if you need to debug macros.

For many years the team have worked on the basis that users will have ε-TeX available but had not revisited the above statements formally. As of the January 2017 release of LaTeX 2ε, ε-TeX is *required* to build the format, and attempting to build a format without the extensions will fail.

Practically, modern TeX distributions provide the extensions in all engines other than the "pure" Knuth `tex`, and indeed parts of the format-building process already require ε-TeX, most notably some of the UTF-8 hyphenation patterns. As such, there should be no noticeable effect on users of this change.

The team expect to make wider use of ε-TeX within the kernel in future; details will be announced where they impact on end users in a visible way.

Default encodings in XeLaTeX and LuaLaTeX

The default encoding in LaTeX has always been the original 128-character encoding `OT1`. For Unicode based TeX engines, this is not really suitable, and is especially problematic with XeLaTeX as in the major distributions this is built with Unicode based hyphenation patterns in the format. In practice this has not been a major problem as documents use the contributed fontspec package in order to switch to a Unicode encoded font.

In this release we are adding `TU` as a new supported encoding in addition to the previously supported encodings such as `OT1` and `T1`. This denotes a Unicode based font encoding. It is essentially the same as the `TU` encoding that has been on trial with the experimental `tuenc` option to fontspec for the past year.

The XeLaTeX and LuaLaTeX formats will now default to `TU` encoding and `lmr` (Latin Modern) family. In the case of LuaLaTeX the contributed `luaotfload` Lua module will be loaded at the start of each run to enable the loading of OpenType fonts.

The fontspec package is being adjusted in a companion release to recognise the new encoding default arrangements.

Note that in practice no font supports the full Unicode range, and so `TU` encoded fonts, unlike fonts specified for `T1`, may be expected to be incomplete in various ways. In the current release the file `tuenc.def` that implements the `TU` encoding-specific commands has made some basic assumptions for (for example) default handling of accent commands, and the set of command names is derived from the command names used for the UTF-8 support in the inputenc package, restricted roughly to the character ranges classically provided by the `T1` and `TS1` encodings, but is part of a longer term plan seen over recent releases to increase support for Unicode based TeX engines into the core LaTeX support.

If for any reason you need to process a document with the previous default `OT1` encoding, you may switch encoding in the usual ways, for example

```
\usepackage[OT1]{fontenc}
```

or you may roll back all the changes for this release by starting the document with

```
\RequirePackage[2016/12/31]{latexrelease}
```

\showhyphens in XeLaTeX

Due to the way XeLaTeX interfaces to font libraries, the standard definition of `\showhyphens` does not work. A variant definition has been available in the contributed xltxtra package, however a (slightly different) definition for `\showhyphens` is now included in XeLaTeX by default. As usual this change will be undone if an earlier release is specified using the latexrelease package.

The fixltx2e package

As described in LaTeX News 22, the fixltx2e package has become obsolete with the new update policy. Since 2015 it has just made a warning and exited. In this release we have re-introduced all the code from the original fixes in the 2014 LaTeX but guarded by `\IncludeInRelease{2015/01/01}`. So for current releases fixltx2e still just displays a warning but for old releases, whether that is an old format, or a format with the version date reset via the latexrelease package, the fixes in the original fixltx2e will be applied.

This improves the ability to run old documents in a way that is compatible with contemporary formats. If you have a 2014 document that used `\usepackage{fixltx2e}` and you add `\RequirePackage[2014/01/01]{latexrelease}` and process it with the current format then latexrelease will undo most changes made since 2014, but now when the document includes fixltx2e it will act like a 2014 version of the package and apply the code fixes, not just give a warning that the package is obsolete.

The latexbug package

As explained in more detail at the LaTeX Project website[1] a new package, latexbug, has been produced to help produce test files to accompany bug reports on the core LaTeX distribution. This is being published separately to CTAN at the same time as this release. By using the latexbug package you can easily check that the packages involved in the test are all part of the core release. The LaTeX project cannot handle bug reports on contributed packages, which should be directed to the package maintainer as given in the package documentation.

Updates to amsmath

The amsmath package has two updates at this release.

- The spacing to the left of the `aligned` and `gathered` environments has been fixed: the spurious thin space is no longer added by default. Package options control this to revert to the original behaviour where required; see the `amsldoc` guide for further details.
- The large delimiters around generalised fractions (for example in the `\binom` construct) did not work in previous releases if using LuaTeX or XeTeX with OpenType math fonts. This is related to the lack of specific metrics for this use in the OpenType Math table. In principle LuaTeX has two additional named metrics to control the delimiters but these are not initialised by default, and in XeTeX it does not seem possible to make them work at all. So for Unicode TeX systems, a new implementation of `\genfrac` is used at this release that uses `\left\right` internally but parameterised to give spacing as close to the original as possible. The implementation in (pdf)TeX is unaffected.

Updates to tools

The array package has been updated to fix a longstanding but previously unreported issue with unwanted interactions between tables in the page head or foot and the body of the page, as reported in PR tools/4488. There is also an update to the LuaTeX support in bm.

An addendum to the release changes in 2015: page breaks and vertical spacing

In 2015 we announced the introduction of the roll-back/roll-forward concept to manage bug fixes and additions to core LaTeX in a manageable way. We also announced at that time that we now incorporate all fixes from fixltx2e into the kernel (as the old mechanism produced problems instead of improving the situation). Refer to ltnews22.pdf for details.

One of the fixes from fixltx2e was for a glaring bug in `\addvspace` that was originally detected in the mid-nineties and back then added to the fixltx2e support package. In certain situations `\addvspace` would result in a page/column break below the baseline of the last line. As a result documents using `\flushbottom` would show a clear misalignment (even more prominent when typesetting in two-column mode).

Starting with release 2015/01/01 this is now finally corrected already in the kernel and not only in fixltx2e. In nearly all circumstances this will either make no difference to existing documents, or it will locally improve the visual appearance of that document without changing anything on other pages. However, by the nature of the change it is also possible that there are further non-local changes to the page breaks due to the different break positions introduced by the fix.

Thus, for documents that have been written before 2015 and that should be preserved unchanged at all costs you may have to add

```
\RequirePackage[2014/01/01]{latexrelease}
```

at the top of the document, to roll back the format to a date before the policy change.

[1] https://www.latex-project.org/bugs/

LaTeX News

Issue 27, April 2017 (LaTeX release 2017-04-15)

Contents

ISO 8601 Date format

Since before the first releases of LaTeX 2ε, LaTeX has used a date format in the form YYYY/MM/DD. This has many advantages over more conventional formats, as it is easy to sort and avoids the unfortunate ambiguity between different communities as to whether 01/02/2017 is the 1st of February or 2nd of January.

However there is another date format, formalised by the International Standard ISO 8601. The basic format defined by this standard is functionally equivalent to the LaTeX format, but using `-` rather than `/`. This date format is now supported in many Operating Systems and applications (for example the `date --iso-8601` command in Linux and similar systems).

From this release, LaTeX will accept ISO format date strings in the date argument of `\ProvidesPackage`, `\usepackage`, etc. Currently we recommend that you do not use this format in any packages that need to work with older LaTeX releases; the latexrelease package may be used with older releases to add this functionality. This change is handled in a special way by latexrelease: The package always adds support for ISO dates whatever format date is requested; this is required so that the necessary date comparisons may be made.

The new functionality can be seen in the startup banner which advertises `LaTeX2e <2017-04-15>`.

Further TU encoding improvements

The 2017/01/01 release saw the introduction of the new TU encoding for specifying Unicode fonts with LuaTeX and XeTeX. There were a number of small corrections and additions in the patch releases updating 2017/01/01, and a further addition in this release, notably extended support for the dot-under accent, `\d`.

Disabling hyphenation

The existing LaTeX code for `\verb` and `verbatim` had some issues when used with fonts that were not loaded with hyphenation disabled via setting `\hyphenchar` to −1. In this release these verbatim environments use a `\language` setting, `\l@nohyphenation`, that has no hyphenation patterns associated.

The format ensures that a language has been allocated with this name. For most users this will in fact be no change as the standard babel language has for a long time allocated a language with this name.

In order that page breaks in `verbatim` do not influence the language used in the page head and foot, the format now normalises the language used in the output routine to a default language as described below.

Discretionary hyphenation

The LaTeX definition of `\-` has been adjusted so that it will insert the current font's `\hyphenchar`, as would the TeX primitive. A comment in `source2e` has given this new definition since the first releases of LaTeX 2ε, and in this release we finally acted upon this comment. Previously `\-` always inserted a `-` at a break point even if a different character would be used for automatic hyphenation with the current font.

Default document language

A new integer parameter `\document@default@language` is introduced; this is initialised to −1 but is set at `\begin{document}` to the language in force at that time if it has not been set by preamble code. This is very similar to the handling of the default color, and is used in a similar way to normalise the settings for page head and foot as described above. Users should not normally need to set this explicitly but it is expected that language packages such as babel may set this if the default behaviour is not suitable.

Line spacing in parboxes

Inside a `\parbox` LaTeX normalises the baseline spacing. However it has not previously reset `\lineskiplimit`. This meant that lines of a paragraph that have ascenders or descenders could be set with *closer* line spacing than lines without. This can easily happen if you use a `\parbox` in an AMS alignment, as they use a relatively large value of `\lineskiplimit`. As usual, the latexrelease package may be used to force the older behavior.

LaTeX News

Issue 28, April 2018 (LaTeX release 2018-04-01)

Contents

A new home for LaTeX 2ε sources

In the past the development version of the LaTeX 2ε source files has been managed in a Subversion source control system with read access for the public. This way it was possible to download in an emergency the latest version even before it was released to CTAN and made its way into the various distributions.

We have recently changed this setup and now manage the sources using Git and placed the master sources on GitHub at

https://github.com/latex3/latex2e

where we already store the sources for expl3 and other work. As before, direct write access is restricted to LaTeX Project Team members, but everything is publicly accessible including the ability to download, clone (using Git) or checkout (using SVN). More details are given in [1].

Bug reports for core LaTeX 2ε

For more than two decades we used GNATS, an open source bug tracking system developed by the FSF. While that has served us well in the past it started to show its age more and more. So as part of this move we also decided to finally retire the old LaTeX bug database and replace it with the standard "Issue Tracker" available at Github.

The requirements and the workflow for reporting a bug in the core LaTeX software is documented at

https://www.latex-project.org/bugs/

and with further details also discussed in [1].

UTF-8: the new default input encoding

The first TeX implementations only supported reading 7-bit ASCII files—any accented or otherwise "special" character had to be entered using commands, if it could be represented at all. For example to obtain an "ä" one would enter `\"a`, and to typeset a "ß" the command `\ss`. Furthermore fonts at that time had 128 glyphs inside, holding the ASCII characters, some accents to build composite glyphs from a letter and an accent, and a few special symbols such as parentheses, etc.

With 8-bit TeX engines such as pdfTeX this situation changed somewhat: it was now possible to process 8-bit files, i.e., files that could encode 256 different characters. However, 256 is still a fairly small number and with this limitation it is only possible to encode a few languages and for other languages one would need to change the encoding (i.e., interpret the character positions 0–255 in a different way). The first code points 0–127 were essentially normed (corresponding to ASCII) while the second half 128–255 would vary by holding different accented characters to support a certain set of languages.

Each computer used one of these encodings when storing or interpreting files and as long as two computers used the same encoding it was (easily) possible to exchange files between them and have them interpreted and processed correctly.

But different computers may have used different encodings and given that a computer file is simply a sequence of bytes with no indication for which encoding is intended, chaos could easily happen and has happened. For example, the German word "Größe" (height) entered on a German keyboard could show up as "GrŤàe" on a different computer using a different encoding by default.

So in summary the situation wasn't at all good and it was clear in the early nineties that LaTeX 2ε (that was being developed to provide a LaTeX version usable across the world) had to provide a solution to this issue.

The LaTeX 2ε answer was the introduction of the inputenc package [2] through which it is possible to

provide support for multiple encodings. It also allows to correctly process a file written in one encoding on a computer using a different encoding and even supports documents where the encoding changes midway.

Since the first release of LaTeX 2$_\varepsilon$ in 1994, LaTeX documents that used any characters outside ASCII in the source (i.e. any characters in the range of 128–255) were supposed to load inputenc and specify in which file encoding they were written and stored. If the inputenc package was not loaded then LaTeX used a "raw" encoding which essentially took each byte from the input file and typeset the glyph that happened to be in that position in the current font—something that sometimes produces the right result but often enough will not.

In 1992 Ken Thompson and Rob Pike developed the UTF-8 encoding scheme which enables the encoding of all Unicode characters within 8-bit sequences. Over time this encoding has gradually taken over the world, replacing the legacy 8-bit encodings used before. These days all major computer operating systems use UTF-8 to store their files and it requires some effort to explicitly store files in one of the legacy encodings.

As a result, whenever LaTeX users want to use any accented characters from their keyboard (instead of resorting to `\"a` and the like) they always have to use

```
\usepackage[utf8]{inputenc}
```

in the preamble of their documents as otherwise LaTeX will produce gibberish.

The new default

With this release, the default encoding for LaTeX files has been changed from the "fall through raw" encoding to UTF-8 if used with classic TeX or pdfTeX. The implementation is essentially the same as the existing UTF-8 support from `\usepackage[utf8]{inputenc}`.

The LuaTeX and XeTeX engines always supported the UTF-8 encoding as their native (and only) input encoding, so with these engines inputenc was always a no-op.

This means that with new documents one can assume UTF-8 input and it is no longer required to always specify `\usepackage[utf8]{inputenc}`. But if this line is present it will not hurt either.

Compatibility

For most existing documents this change will be transparent:

- documents using only ASCII in the input file and accessing accented characters via commands;
- documents that specified the encoding of their file via an option to the inputenc package and then used 8-bit characters in that encoding;
- documents that already had been stored in UTF-8 (whether or not specifying this via inputenc).

Only documents that have been stored in a legacy encoding and used accented letters from the keyboard *without* loading inputenc (relying on the similarities between the input used and the T1 font encoding) are affected.

These documents will now generate an error that they contain invalid UTF-8 sequences. However, such documents may be easily processed by adding the new command `\UseRawInputEncoding` as the first line of the file. This will re-instate the previous "raw" encoding default.

`\UseRawInputEncoding` may also be used on the command line to process existing files without requiring the file to be edited

```
pdflatex '\UseRawInputEncoding \input'  file
```

will process the file using the previous default encoding.

Possible alternatives are reencoding the file to UTF-8 using a tool (such as recode or iconv or an editor) or adding the line

```
\usepackage[⟨encoding⟩]{inputenc}
```

to the preamble specifying the ⟨*encoding*⟩ that fits the file encoding. In many cases this will be `latin1` or `cp1252`. For other encoding names and their meaning see the inputenc documentation.

As usual, this change may also be reverted via the more general latexrelease package mechanism, by specifying a release date earlier than this release.

BOM: byte order mark handling

When using Unicode the first bytes of a file may be a, so called, BOM character (byte order mark) to indicate the byte order used in the file. While this is not required with UTF-8 encoded files (where the byte order is known) it is nevertheless allowed by the standard and some editors add that byte sequence to the beginning of a file. In the past such files would have generated a "Missing begin document" error or displayed strange characters when loaded at a later stage.

With the addition of UTF-8 support to the kernel it is now possible to identify and ignore such BOMs characters even before `\documentclass` so that these issues will no longer be showing up.

A general rollback concept for packages and classes

In 2015 a rollback concept for the LaTeX kernel was introduced. Providing this feature allowed us to make corrections to the software (which more or less didn't happen for nearly two decades) while continuing to maintain backward compatibility to the highest degree.

In this release we have now extended this concept to the world of packages and classes which was not covered initially. As the classes and the extension packages have different requirements compared to the kernel, the approach is different (and simplified). This should make it easy for package developers to apply it to their packages and authors to use when necessary.

The documentation of this new feature is given in an article submitted to TUGboat and also available from our website [3].

Integration of remreset and chngcntr packages into the kernel

With the optional argument to `\newcounter` LaTeX offers to automatically reset counters when some counter is stepped, e.g., stepping a `chapter` counter resets the `section` counter (and recursively all other heading counters). However, what was until now missing was a way to undo such a link between counters or to link two counters after they have been defined.

This can be now be done with `\counterwithin` and `\counterwithout`, respectively. In the past one had to load the chngcntr package for this. For the programming level we also added `\@removefromreset` as the counterpart of the already existing `\@addtoreset` command. Up to now this was offered by the remreset package.

Testing for undefined commands

LaTeX packages often use a test `\@ifundefined` to test if a command is defined. Unfortunately this had the side effect of *defining* the command to `\relax` in the case that it had no definition. The new release uses a modified definition (using extra testing possibilities available in ε-TeX). The new definition is more natural, however code that was relying on the side effect of the command being tested being defined if it was previously undefined may have to add `\let\`⟨*command*⟩`\relax`.

Changes to packages in the tools category

LaTeX table columns with fixed widths

Frank published a short paper in TUGboat [4] on producing tables that have columns with fixed widths. The outlined approach using column specifiers "`w`" and "`W`" has now been integrated into the array package.

Obscure overprinting with multicol fixed

A rather peculiar bug was reported on StackExchange for multicol. If the column/page breaking was fully controlled by the user (through `\columnbreak`) instead of letting the environment do its job and if then more `\columnbreak` commands showed up on the last page then the balancing algorithm was thrown off track. As a result some parts of the columns did overprint each other.

The fix required a redesign of the output routines used by multicol and while it "should" be transparent in other cases (and all tests in the regression test suite came out fine) there is the off-chance that code that hooked into internals of multicol needs adjustment.

Changes to packages in the amsmath category

With this release of LaTeX a few minor issues with amsmath have been corrected.

Updated user's guide

Furthermore, `amsldoc.pdf`, the AMS user's guide for the amsmath package [5], has been updated from version 2.0 to 2.1 to incorporate changes and corrections made between 2016 and 2018.

References

[1] Frank Mittelbach: *New rules for reporting bugs in the LaTeX core software*. In: TUGboat, 39#1, 2018. https://www.latex-project.org/publications/

[2] Frank Mittelbach: *LaTeX 2ε Encoding Interface — Purpose, concepts, and Open Problems*. Talk given in Brno June 1995. https://www.latex-project.org/publications/

[3] Frank Mittelbach: *A rollback concept for packages and classes*. Submitted to TUGboat. https://www.latex-project.org/publications/

[4] Frank Mittelbach: *LaTeX table columns with fixed widths*. In: TUGboat, 38#2, 2017. https://www.latex-project.org/publications/

[5] American Mathematical Society and The LaTeX Project: *User's Guide for the amsmath package* (Version 2.1). April 2018. Available from https://www.ctan.org and distributed as part of every LaTeX distribution.

LaTeX News

Issue 29, December 2018 (LaTeX release 2018-12-01)

Contents

Introduction

The December 2018 release of LaTeX is a maintenance release in which we have fixed a few bugs in the software: some are old, some newer, and they are mostly rather obscure.

Bug reports for core LaTeX 2ε and packages maintained by the Project Team

In Spring 2018 we established a new issue tracking system (Github issues at https://github.com/latex3/latex2e/issues) for both the LaTeX core and the packages maintained by the LaTeX Project Team, with an updated procedure for how to report a bug or problem.

Initial experience with this system is good, with people who report problems following the guidelines and including helpful working examples to show the problem—thanks for doing this.

The detailed requirements and the workflow for reporting a bug in the core LaTeX software is documented at

https://www.latex-project.org/bugs/

with further details and discussion in [1].

Changes to the LaTeX kernel

UTF-8: updates to the default input encoding

In the April 2018 release of LaTeX we changed the default encoding from 7-bit ASCII to UTF-8 when using classic TeX or pdfTeX; see *LaTeX News 28* [2] for details.

Now, after half a year of experience with this new default, we have made a small number of adjustments to further improve the user experience. These include:

- Some improvements when displaying error messages about UTF-8 characters that have not been set up for use with LaTeX, or are invalid for some other reason; *(github issues 60, 62 and 63)*
- The addition of a number of previously missing declarations for characters that are in fact available with the default fonts, e.g., \j "ȷ" (0237), \SS "SS" (1E9E), \k{} "˛" (02DB) and \.{} "˙" (02D9);
- Correcting the names for \guillemetleft "«" and \guillemetright "»" in all encoding files. These correct names are in addition to the old (but wrong) Adobe names: Adobe mistakenly called them Guillemot, which is a sea bird. *(github issue 65)*
- Added \Hwithstroke ("Ħ") and \hwithstroke ("ħ") necessary for typesetting Maltese. *(https://tex.stackexchange.com/q/460110)*

Fixed \verb* and friends in XeTeX and LuaTeX

The original \verb* and verbatim* in LaTeX were coded under the assumption that the position of the space character (i.e., ASCII 32) in a typewriter font contains a visible space glyph "␣". This is correct for pdfTeX with the most used font encodings OT1 and T1. However, this unfortunately does not work for Unicode engines using the TU encoding since the space character slot (ASCII 32) then usually contains a real (normal) space, which

has the effect that `\verb*` produces the same results as `\verb`.

The `\verb*` code now always uses the newly introduced command `\verbvisiblespace` to produce the visible space character and this command will get appropriate definitions for use with the different engines. With pdfTeX it will simply use `\asciispace`, which is a posh name for "select character 32 in the current font", but with Unicode engines the default definition is

```
\DeclareRobustCommand\verbvisiblespace
   {\leavevmode
    {\usefont{OT1}{cmtt}{m}{n}\asciispace}}
```

which uses the visible space from the font Computer Modern Typewriter, regardless of the currently chosen typewriter font. Internally the code ensures that the character used has exactly the same width as the other characters in the current (monospaced) font; thus, for example, code displays line up properly.

It is possible to redefine this command to select your own character, for example

```
\DeclareRobustCommand\verbvisiblespace
   {\textvisiblespace}
```

will select the "official" visible space character of the current font. This may look like the natural default, but it wasn't chosen as our default because many fonts just don't have that Unicode character, or they have one with a strange shape. *(github issues 69 and 70)*

Error message corrected

Trying to redefine an undefined command could in a few cases generate an error message with a missing space, e.g., `\renewcommand\1{...}` gave

```
LaTeX Error: \1undefined.
```

This is now fixed. *(github issue 41)*

Fixed fatal link error with hyperref

If an `\href` link text gets broken across pages, pdfTeX and LuaTeX will generate a fatal error unless both parts of the link are internally at the same boxing level. In two-column mode that was not the case if one of the pages had spanning top floats. This has now been changed so that the error is avoided. *(github issue 94)*

Avoid page breaks caused by invisible commands

Commands like `\label` or `\index` could generate a potential page break in places where a page break was otherwise prohibited, e.g., when used between two consecutive headings. This has now been corrected. If for some reason you really want a break and you relied on this faulty behavior, you can always add one using `\pagebreak`, with or without an optional argument. *(github issue 81)*

Prevent spurious spaces when reading table of contents data

When table of contents data is read in from a `.toc` file, the new-line character at the end of each line is converted by TeX to a space. In normal processing this is harmless (as TeX is doing this input reading whilst in vertical mode and each line in the file represents a single line (paragraph) in the table of contents. If, however, this is done in horizontal mode, which is sometimes the case, then these spaces will appear in the output. If you then omit some of the input lines (e.g., because you do not display TOC data below a certain level), then these spaces accumulate in the typeset output and you get surprising, and unwanted, gaps inside the text.

The new code now adds a `%` sign at the end of problematic lines in the `.toc` file so that TeX will not generate such spaces that may survive to spoil the printed result. As some third party packages have augmented or changed the core LaTeX functionality in that area (for example, by adding additional arguments to the commands in TOC files) the code uses a conservative approach and the `%` signs are added only when certain conditions are met. Therefore some packages might require updates if they want to benefit from this correction, especially if they unconditionally overwrite LaTeX's `\addcontentsline` definition. *(github issue 73)*

Prevent protrusion in table of contents lines

In TeX's internal processing model, paragraph data is one of the major data structures. As a result, many things are internally modeled as paragraphs even if they are not conceptually "text paragraphs" in the traditional sense. In a few cases this has some surprising effects that are not always for the better. One example is standard TOC entries, where you have heading data followed by some dot leaders and a page number at the right, produced, for example, from this:

Error message corrected 2

The space reserved for the page number is of a fixed width, so that the dots always end in the same place. Well, they did end in the same place until the advent of protrusion support in the TeX engines. Now, with the microtype package loaded, it is possible that the page number will protrude slightly into the margin (even though it's typeset inside a box) and as a result this page number box gets shifted. With enough bad luck this can get you another dot in the line, sticking out like the proverbial sore thumb, as exhibited in the question on StackExchange that triggered the correction.

LaTeX now takes care that there will be no protrusion happening on such lines, even if it is generally enabled for the whole document. *(https://tex.stackexchange.com/q/172785)*

Start L-R mode for \thinspace and friends

In LaTeX, commands that are intended only for paragraph (L-R) mode are generally careful to start paragraph mode if necessary; thus they can be used at the start of a paragraph without surprising and unwanted consequences. This important requirement had been overlooked for a few horizontal spacing commands, such as `\thinspace` (a.k.a. "`\,`"), and for some other support commands such as `\smash` or `\phantom`. Thus they ended up adding vertical space when used at the beginning of a paragraph or, in the case of `\smash`, creating a paragraph of their own. This has now been corrected, and a corresponding update has been made to the amsmath package, in which these commands are also defined. *(github issues 49 and 50)*

Guarding \pfill in doc

For presenting index entries pointing to code fragments and the like, the doc package has a `\pfill` command that generates within the index a line of dots leading from the command name to the page or code line numbers. If necessary it would automatically split the entry over two lines. That worked well enough for a quarter century, but we discovered recently that it is broken inside the ltugboat class, where it sometimes produces bad spacing within continuation lines.

The reason turned out to be a redefinition of the LaTeX command `\nobreakspace` (`~`) inside the class ltugboat, which removed any preceding space (and thus unfortunately also removed the dots on the continuation line). While one can argue that this is a questionable redefinition (if only done by a single class and not generally), it has been in the class so long that changing it would certainly break older documents. So instead we now guard against that removal of space. *(github issues 25 and 75)*

Changes to packages in the tools category

Sometimes the trace package turned off too much

The trace package is a useful little tool for tracing macro execution: it hides certain lengthy and typically uninteresting expansions resulting from font changes and similar activities. However, it had the problem that it also reset other tracing settings such as `\showoutput` in such situations, so that you couldn't use `\showoutput` in the preamble to get symbolic output of all the pages in the document. This has now been corrected.

Update to xr

The xr package has been updated so that the code that reads the `.aux` file has been made more robust. It now correctly ignores conditionals (added by hyperref and other packages) rather than generating low level parsing errors. *(https://tex.stackexchange.com/a/452321)*

Column data for multicols sometimes vanished*

In certain situations involving `multicols*`, when there are more explicit `\columnbreak` requests than there are columns on the current page, data could vanish due to the removal of an internal penalty marking the end of the environment. This has been corrected by explicitly reinserting that penalty if necessary. *(github issue 53)*

Extension to \docolaction in multicol

The `\docolaction` command can be used to carry out actions depending on the column you are currently in, i.e., first, any inner one (if more than two) or last. However, if the action generates text then there is the question: is this text part of the current column or the one after? That is, on the next run, do we test before or after it, to determine in which column we are?

This is now resolved as follows: if you use `\docolaction*` any generated text by the chosen action is considered to be after the test point. But if you use the command without the star then all the material it generates will be placed before the test point to determine the current column, i.e., the text will become part of the current column and may affect the test result on the next run.

Prevent color leak in array

In some cases the color used inside a `tabular` cell could "leak out" into the surrounding text. This has been corrected. *(github issue 72)*

Support fragile commands in array or tabular column templates

The preamble specifiers `p`, `m` and `b` each receives a user supplied argument: the width of the paragraph column. Normally that is something harmless, like a length or a simple length expression. But in more complicated settings involving the calc package it could break with a low-level error message. This has now been corrected. *(https://tex.stackexchange.com/q/459285)*

Changes to packages in the amsmath category

The changes in the kernel made for `\thinspace`, `\smash`, etc. (see above) have been reflected in the amsmath package code, so that loading this package doesn't revert them. *(github issues 49 and 50)*

Website updates

Publications area reorganized and extended

To help readers to find relevant information in more convenient and easy ways, the area of the website covering publications by the LaTeX Project Team was reorganized and extended (many more abstracts added). We now provide the articles, talks and supplementary data structured both by year and also by major topics [4]. Feel free to take a look.

Japanese translations of the user's guide
Yukitoshi Fujimura has kindly translated into Japanese two documents that are distributed with standard LaTeX. These are:

- LaTeX 2ε for authors;
- User's Guide for the amsmath Package [5].

They can be found on the website documentation page [3]. You will now also find there a typeset version of the full LaTeX 2ε source code (with index etc.) and a number of other goodies.

References

[1] Frank Mittelbach: *New rules for reporting bugs in the LaTeX core software*. In: TUGboat, 39#1, 2018. https://latex-project.org/publications/

[2] *LaTeX News, Issue 28*. In: TUGboat, 39#1, 2018. https://latex-project.org/news/latex2e-news/

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

[4] *LaTeX Project publications on the LaTeX Project Website*. https://latex-project.org/publications/

[5] American Mathematical Society and The LaTeX Project: *User's Guide for the amsmath Package* (Version 2.1). April 2018. Available from https://www.ctan.org and distributed as part of every LaTeX distribution.

LaTeX News

Issue 30, October 2019 (LaTeX release 2019-10-01)

Contents

LaTeX-dev formats now available

We know that many of you, especially developers and maintainers of important packages, have a strong interest in a stable LaTeX environment.

In order to keep LaTeX very stable for users whilst allowing for further development to continue, we now have a development branch of LaTeX on GitHub containing development code for the upcoming release. When this code is ready for wider consumption and testing, we generate a pre-release of LaTeX from this development branch and make it available on CTAN.

For users of the TeX Live and MiKTeX distributions it is therefore now straightforward to test their documents and code against the upcoming LaTeX release with ease, simply by selecting a different program name (when using the command line) or by selecting a menu entry (after setting it up; see below).

If you do this then the latest version of the LaTeX development format will be used to process your document, allowing you to test the upcoming release with your own documents and packages. For example, if you run

```
pdflatex-dev myfile
```

then you will be greeted on the screen with something like `LaTeX2e <2019-10-01> pre-release-2` (identifying the pre-release format) instead of the normal `LaTeX2e <2018-12-01>`. In this pre-release you will find the latest new features that we have developed.

Our hopes

We don't expect everybody to start using the development formats to participate in testing, but we hope that people with a strong interest in a stable LaTeX environment (especially developers and maintainers of important packages) will use the new facilities and help us to ensure that future public releases of LaTeX do not (as has happened in the past) require some immediate patches because of issues that were not identified by our internal regression test suite or by other testing we do.

Any issue identified when using the development format should preferably be logged as an issue on GitHub, following the procedure outlined on our website at https://www.latex-project.org/bugs/ including the use of the latexbug package as described.

Our bug reporting process normally states that issues involving third-party software are out of scope as we can't correct external packages; see [1]. However, in the particular case of the development format showing an incompatibility with a third-party package, it is fine to open an issue with us (in addition, please, to informing the maintainer of that package) so that we know about the problem and can jointly work on resolving it.

Details please . . .

More details and some background information about the concepts and the process are available in an upcoming *TUGboat* article: "The LaTeX release workflow and the LaTeX dev formats" [2].

Setting up menu items

While the command line call works out of the box if you have a recent TeX Live or MiKTeX installation, its use within an integrated editing environment doesn't at this point in time (maybe the developers of these editors will include it in the future). However, it is normally fairly simple to enable it as most (or even all?) of them provide simple ways to call your own setup. How this works in detail depends very much on the environment you use, so we can't give much help here.

But as an example: to provide an additional menu entry for `XeLaTeX-dev` on a MacBook all that is necessary is to copy the file `XeLaTeX.engine` to `XeLaTeX-dev.engine` and change the call from `xelatex` to `xelatex-dev` inside.

Improving Unicode handling in pdfTeX

Perhaps the most important improvement in this release is even better support for UTF-8 characters when using pdfTeX.[1]

When using a "Unicode engine", any Unicode character (that is not acting as a command, i.e., is not "active") can be used as part of the `\label`/`\ref` mechanism or can be displayed in a message or written to a file. In 8-bit engines, however, this was severely restricted: essentially you had to limit yourself to using ASCII letters, digits and a few punctuation symbols. With the new release, most of these restrictions have been removed and you now can write labels such as

```
\label{eq:größer}
```

or use accented characters, etc., as part of a `\typeout` message. The only requirement remaining is that only those UTF-8 characters that are also available for typesetting can be used, i.e., only those characters for which adequate font support is loaded. Otherwise you will get an error message stating that the particular Unicode character is not set up for use with LaTeX.

Note, however, that the restrictions on what characters can be used in the names of commands have not changed.

What is not possible when using an 8-bit engine such as pdfTeX is to use characters other than ASCII letters as part of a command name. This is due to the fact that all other characters in such engines are not single character tokens, but in fact consist of a sequence of bytes and this is not supported in command names.

[1] The Japanese engines e-pTeX and e-upTeX can't use these features yet as they don't support the primitive `\ifincsname`. Work is under way to resolve this in the engines.

Improving file name handling in pdfTeX

A related change is that file names used as part of `\input`, `\includegraphics`, etc., commands can now contain any Unicode characters allowed by the file system in use, including spaces. In this case, even characters that can't be typeset (due to lack of font support) can be used.

Improving the `filecontents` environment

The `filecontents` environment now supports an optional argument in which you can specify that it is allowed to `overwrite` an already existing file; by default nothing is written if a file with the given name exists anywhere in the search tree. An alternative name for this option is `force`. Even then the environment will refuse to write to `\jobname.tex` to avoid clobbering its own input file. However, if you use a different extension on your input file you could still overwrite it (there is no way to test for that).

There is also an option `nosearch`, which specifies that only the current directory is examined for an existing file, not the whole TeX inputs tree. This is useful if you want to write a local copy of a standard system file. Finally, `noheader` prevents writing a preamble to the file (this is the same as using the star form of the environment).

Another change is that this environment is now allowed anywhere in the document, which means it provides everything (and more) of what the now obsolete filecontents package provided.

Making more user commands robust

In the early days of LaTeX many commands were fragile, i.e., they needed `\protect` in front of them when used in places such as section headings and other "moving arguments", etc. In LaTeX 2_ε many of these commands were made robust, but still a fairly large number remained unnecessarily fragile.

In this release of LaTeX we have now made a lot more commands robust. There is a very small collection of commands that must stay fragile because their expansion (maybe partially) at just the right time is critical. Yet others are unlikely to ever be needed in a "moving argument".

Doing this for `\begin` and `\end` was rather tricky as the standard mechanism with `\DeclareRobustCommand` doesn't work here, at least not for `\end` as that needs to expand during typesetting without generating a `\relax` (from the `\protect`). Such a token would start a new row in table environments, such as `tabular`, etc. Furthermore, some packages try to look into the definition of `\end` by expanding it several times. Thus expansion with `\expandafter` had to produce exactly the same result as before. But in the end we overcame

that hurdle too, so now environments are automatically robust if used in places like headings or `\typeout` and so forth.

What hasn't been tackled yet is the redefinitions in amsmath: this package redefines a number of basic math constructs that are now robust, so that they become fragile again once the package is loaded. This area will be addressed in a followup release. *(github issue 123)*

Other changes to the LaTeX kernel

Guard against `\unskip` in tabular cells

If a `tabular` or `array` cell started with a command that started with an `\unskip` then centering the column broke because the stretching glue on the left got removed. The fix for this was to add a minuscule, and hence unnoticeable, additional space after the stretching space: removing this extra space causes no problems.

This change was also applied in the array package. *(github issue 102)*

Fix Unicode table data

`U+012F` which is "i with ogonek" produced a "dotless i with ogonek" by mistake. This has been corrected. *(github issue 122)*

The Unicode slots `27E8` and `27E9` have been mapped to `\textlangle` and `\textrangle` which is the recommended mapping. In the past they raised a LaTeX error. *(github issue 110)*

When doing cut-and-paste from other documents or websites, f-ligatures and others ligatures might end up as single Unicode characters in your file. In the past those got rejected by LaTeX. We now define those Unicode slots and map them back to the sequence of individual characters constituting the ligature. If supported by the current font (which is normally the case) they are then reconstructed as ligatures and thus get typeset as desired. Otherwise they will come out as individual characters which is still better than an error message. *(github issue 154)*

Improve `\InputIfFileExists`'s handling of file names

In rare circumstances it was possible that `\InputIfFileExists` would work incorrectly, e.g., a construction such as

```
\InputIfFileExists{foo}{\input{bar}}{}
```

would not load the files `foo.tex` and `bar.tex` but would load `bar.tex` twice. This has been corrected. *(github issue 109)*

Improve interface for cross-references

The packages fncylab and varioref provided a slightly improved definition of `\refstepcounter` which allowed the internal `\p@..` commands to receive the counter value as an argument, instead of acting as a simple prefix. This supports more complex formatting of the value in the reference.

These packages also provided the command `\labelformat` to help in the specification of such formatting in an easy way. For example, `\labelformat{equation}{eq.~(#1)}` specifies that references to equations automatically come out as "eq. (5)" or similar. As such a `\labelformat` declaration means a `\ref` command can no longer be successfully used at the start of a sentence, the packages also provided `\Ref` for such scenarios.

Both of these commands, `\labelformat` and `\Ref`, are now removed from the packages and instead made available in the kernel so there is no need to load additional packages.

Improve wording of a warning message

The kernel now says "Trying to load ..." instead of "Try loading ..." in one of its informal messages to match style of similar messages. *(github issue 107)*

Avoid bad side-effects of `\DeclareErrorFont`

As a side effect of setting up the error font for NFSS, this declaration also changed the current font size back to 10pt. In most circumstances that doesn't matter, because that declaration was meant to be used only during the format generation and not during a LaTeX run. However, it has turned out to be used by some developers in other places (incorrectly in fact: e.g., inside some `.fd` files) where resetting the size causes havoc seemingly at random. The command has now changed to not produce such side effects. *(gnats issue latex/4399)*

nfssfont: Make font table generation the default action

With the small file `nfssfont.tex` it is possible to produce font tables and other font tests in the style set up by Don Knuth. In nearly all cases a font table is wanted, so this action has been made the default. Now one can simply hit enter instead of having to write `\table\bye`.

trace: Add package support in the kernel

The trace package implements the commands `\traceon` and `\traceoff` that work like `\tracingall` but skip certain code blocks that produce a lot of tracing output. This is useful when debugging, to suppress uninteresting tracing from, for example, loading a font. Code blocks that should not be traced need to be surrounded by the commands `\conditionally@traceoff` and `\conditionally@traceon`.

The LaTeX kernel now provides dummy definitions for these two commands so that package writers can use them in their packages regardless of trace being loaded or not.

Changes to packages in the tools category

array: Warn if primitive column specifiers are overwritten

With `\newcolumntype` it is possible to define your own column specifiers for a `tabular` preamble; it is also possible to change existing ones. However, doing that for a primitive column specifier, such as `c`, is seldom a good idea, since then its functionality becomes unavailable. The package was therefore supposed to warn the user in this case, but due to a missing `\expandafter` in the code it never did—now it does. *(github issue 148)*

multicol: Introduce `minrows` counter for balancing

When there are only a few lines of text on a page at the end of a `multicols` environment, balancing the columns often looks rather odd: such as three columns each containing a single line. The balancing behavior can now be controlled through the counter `minrows` (default is 1) which specifies that, after balancing, there must be at least that many lines in the first column. Thus, if you set `minrows` to `2` then you would get a distribution of `2+1+0` lines and if set to three, the result would be `3+0+0` instead of the default `1+1+1`.

What is most appropriate really depends on the circumstances, but this now gives you the tools to make local or global adjustments.

varioref: Better support for cleveref

The varioref package has been internally updated to provide better interfaces for packages such as hyperref and cleveref.

It also has a new package option `nospace` that stops varioref from meddling with space in front of its commands. The original behavior was always somewhat problematical and it is suggested that all new documents use this option (which should really have been the default).

Support was also added for the Arabic language through the option `arabic`.

xr: Support citations to bibliographies in external documents

The xr package can be used to cross-reference an external LaTeX document. This means that even when a work is split over different documents (that need to be processed separately), `\ref` or `\pageref` can use labels from any document, creating links between them. This facility has now been extended so that `\cite` commands and their cousins can now also reference bibliographies in external documents; this feature was first provided in the package xcite by Enrico Gregorio.

Note that for technical reasons xr doesn't work with hyperref. Use xr-hyper instead if you need the latter package.

Changes to packages in the amsmath category

amsmath: Introduce `\overunderset` command

The amsmath package has always offered the commands `\overset` and `\underset` to produce binary operators with something set above or below. But sometimes one needs to put something above and something below: The newly added `\overunderset` makes this easily possible.

Documentation updates

There are a number of documentation updates in files on the documentation page of the project website [4].

Highlighting the standard NFSS codes for series

The *Font Selection Guide* [3] has been updated to strongly recommend that the standard codes should be used when providing font support. The reason for this recommendation is explained here.

The font selection scheme uses a number of standard codes for `\fontseries` and `\fontshape` to ensure that different fonts are comparable, e.g., that you get a "light" weight if you specify `l` and "extra bold" when you write `eb`, etc. Over the years people came up with a number of other creative short codes like `k`, `j`, `t` and others with the result that changing a font family required different codes and thus prevented users from easily mixing and matching different families. Some work has been undertaken to get back to a coherent scheme and all the font families supported through the program `autoinst` are now producing the standard codes again.

LaTeX base and doc distribution reunited

For a long time the LaTeX distribution available from CTAN was split into several parts to allow them to be uploaded or downloaded separately. As this is these days more confusing than helpful we have recombined the base part with the documentation part (as both are anyway always updated together). Thus the package latex-doc is no longer separately available from CTAN but contained in the latex-base distribution.

References

[1] Frank Mittelbach: *New rules for reporting bugs in the LaTeX core software*. In: TUGboat, 39#1, 2018. https://latex-project.org/publications/

[2] Frank Mittelbach: *The LaTeX release workflow and the LaTeX dev formats*. In: TUGboat, 40#2, 2019. https://latex-project.org/publications/

[3] LaTeX Project Team: *LaTeX 2ε font selection*. https://latex-project.org/documentation/

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

LaTeX News

Issue 31, February 2020 (LaTeX release 2020-02-02)

Contents

Experiences with the LaTeX -dev formats

As reported in the previous *LaTeX News*, we have made a pre-release version of the LaTeX kernel available as `LaTeX-dev`. Overall, the approach of having an explicit testing release has been positive: it is now readily available in TeX systems and is getting real use beyond the team.

The current release has been tested by a number of people, and we have had valuable feedback on a range of the new ideas. This has allowed us to fix issues in several of the new features, as described below.

We wish to thank all the dedicated users who have been trying out the development formats, and we encourage others to do so. Pre-testing in this way does mean that, for the vast majority of users, problems are solved before they even appear!

Concerning this release ... (LuaLaTeX engine)

The new LuaHBTeX engine is LuaTeX with an embedded HarfBuzz library. HarfBuzz can be used by setting a suitable renderer in the font declaration. A basic interface for that is provided by fontspec. This additional font renderer will greatly improve the shaping of various scripts when using LuaLaTeX, many of which are currently handled correctly only by XeTeX, which always uses HarfBuzz.

To simplify testing of the new engine, binaries have already been added to MiKTeX and TeX Live 2019 and both distributions have already now changed the LuaLaTeX-dev format to use it.

Going forward, LuaLaTeX (and LuaLaTeX-dev) will both use the LuaHBTeX engine. The timing of the switch to the LuaHBTeX engine depends on the distribution you use (for TeX Live this will be with TeX Live 2020).

Improved load-times for expl3

The LaTeX3 programming layer, expl3, has over the past decade moved from being largely experimental to broadly stable. It is now used in a significant number of third-party packages, most notably xparse, for defining interfaces in cases where no expl3 code is "visible". In addition, most LaTeX documents compiled using XeTeX or LuaTeX load fontspec, which is written using expl3.

The expl3 layer contains a non-trivial number of macros, and when used with the XeTeX and LuaTeX engines, it also loads a large body of Unicode data. This means that even on a fast computer, there is a relatively large load time when using expl3.

For this release, the team have made adjustments in the LaTeX 2_ε kernel to pre-load a significant portion of expl3 when the format is built. This is transparent to the user, other than the significant decrease in document processing time: there will be no "pause" whilst loading

the Unicode data files. Loading expl3 in documents and packages can continue to be done as usual; eventually, it will be possible to omit

```
\RequirePackage{expl3}
```

entirely but, to support older formats, this is still recommended at present.

Improvements to LaTeX's font selection mechanism (NFSS)

Extending the shape management in NFSS

Over time, more and more fonts have become available for use with LaTeX. Many such font families offer additional shapes such as small caps italic (`scit`), small caps slanted (`scsl`) or swash (`sw`). By using `\fontshape` those shapes can be explicitly selected. For the swash shapes there is also `\swshape` and `\textsw` available.

In the original font selection implementation a request to select a new shape always overrode the current shape. With the 2020 release of LaTeX this has changed and `\fontshape` can now be used to combine small capitals with italic, slanted or swash letters, either by explicitly asking for `scit`, etc., or by asking for italics when typesetting already in small caps, and so forth.

Using `\upshape` will still change italics or slanted back to an upright shape but will not any longer alter the small caps setting. To change small capitals back to upper/lower case you can now use `\ulcshape` (or `\textulc`) which in turn will not change the font with respect to italics, slanted or swash. There is one exception: for compatibility reasons `\upshape` will change small capitals back to upright (`n` shape), if the current shape is `sc`. This is done so that something like `\scshape...\upshape` continues to work as before, but we suggest that you don't use that deprecated method in new documents.

Finally, if you want to reset the shape back to normal you can use `\normalshape` which is a shorthand for `\upshape\ulcshape`.

The way that shapes combine with each other is not hardwired; it is customizable and extensible if there is ever a need for this. The mappings are defined through `\DeclareFontShapeChangeRule` and the details for developers are documented in `source2e.pdf`.

The ideas for this interface extension have been pioneered in fontspec by Will Robertson for Unicode engines, and in fontaxes by Andreas Bühmann and Michael Ummels for pdfTeX; they are by now used in many font support packages.

Extending the font series management in NFSS

Many of the newer font families also come provided with additional weights (thin, semi-bold, ultra-bold, etc.) or several running widths, such as condensed or extra-condensed. In some cases the number of different values for series (weight plus width) is really impressive: for example, Noto Sans offers 36 fonts, from ultra-light extra condensed to ultra-bold medium width.

Already in its original design, NFSS supported 9 weight levels, from ultra-light (`ul`) to ultra-bold (`ub`), and also 9 width levels, from ultra-condensed (`uc`) to ultra-expanded (`ux`): more than enough, even for a font family like Noto Sans. Unfortunately, some font support packages nevertheless invented their own names, so in recent years you have been able to find all kinds of non-standard series names (`k`, `i`, `j` and others), making it impossible to combine different fonts successfully using the standard NFSS mechanisms.

Over the course of the last year a small number of individuals, notably, Bob Tennent, Michael Sharpe and Marc Penninga, have worked hard to bring this unsatisfactory situation back under control; so today we are happy to report that the internal font support files for more than a hundred font families are all back to following the standard NFSS conventions. Combining them is now again rather nice and easy, and from a technical perspective they can now be easily matched; but, of course, there is still the task of choosing combinations that visually work well together.

In the original font selection implementation, a request to select a new series always overrode the current one. This was reasonable because there were nearly no fonts available that offered anything other than a medium or a bold series. Now that this has changed and families such as Noto Sans are available, combining weight and width into a single attribute is no longer appropriate. With the 2020 release of LaTeX, the management of series therefore changed to allow independent settings of the weight and the width attributes of the series.

For most users this change will be largely transparent as LaTeX offers only `\textbf` or `\bfseries` to select a bolder face (and `\textmd` and `\mdseries` to return to a medium series): there is no high-level command for selecting a condensed face, etc. However, using the NFSS low-level interface it is now possible to ask for, say, `\fontseries{c}\selectfont` to get a condensed face (suitable for a marginal note) and that would still allow the use of `\textbf` inside the note, which would select a bold-condensed face (and not a rather odd-looking bold-extended face in the middle of condensed type).

The expectation is that this functionality will be used largely by class and package designers but, given that the low-level NFSS commands are usable on the document level and that they are not really difficult to apply, there are probably also a number of users who will enjoy using these new possibilities that bring LaTeX back into the premier league for font usage.

The ways in which the different series values combine with each other is not hardwired but is again

customizable and extensible. The mappings are defined through `\DeclareFontSeriesChangeRule` and the details for developers are documented in `source2e.pdf`.

Font series defaults per document family

With additional weights and widths now being available in many font families, it is more likely that somebody will want to match, say, a medium weight serif family with a semi-light sans serif family, or that with one family one wants to use the bold-extended face when `\textbf` is used, while with another it should be bold (not extended) or semibold, etc.

In the past this kind of extension was provided by Bob Tennent's mweights package, which has been used in many font support packages. With the 2020 release of LaTeX this feature is now available out of the box. In addition we also offer a document-level interface to adjust the behavior of the high-level series commands `\textbf`, `\textmd`, and of their declaration forms `\bfseries` and `\mdseries`, so that they can have different effects for the serif, sans serif and typewriter families used in a document.

For example, specifying

```
\DeclareFontSeriesDefault[rm]{bf}{sb}
\DeclareFontSeriesDefault[tt]{md}{lc}
```

in the document preamble would result in `\textbf` producing semi-bold (`sb`) when typesetting in a roman typeface. The second line says that the typewriter default face (i.e., the medium series `md`) should be a light-condensed face. The optional argument here can be either `rm`, `sf` or `tt` to indicate one of the three main font families in a document; if omitted you will change the overall document default instead. In the first mandatory argument you specify either `md` or `bf` and the second mandatory argument then gives the desired series value in NFSS nomenclature.

Handling of nested emphasis

In previous releases of LaTeX, nested `\emph` commands automatically alternated between italics and upright. This mechanism has now been generalised so that you can now specify for arbitrary nesting levels how emphasis should be handled.

The declaration `\DeclareEmphSequence` expects a comma separated list of font declarations corresponding to increasing levels of emphasis. For example,

```
\DeclareEmphSequence{\itshape,%
                     \upshape\scshape,\itshape}
```

uses italics for the first, small capitals for the second, and italic small capitals for the third level (provided you use a font that supports these shapes). If there are more nesting levels than provided, LaTeX uses the declarations stored in `\emreset` (by default `\ulcshape\upshape`) for the next level and then restarts the list.

The mechanism tries to be "smart" by verifying that the given declarations actually alter the current font. If not, it continues and tries the next level—the assumption being that there was already a manual font change in the document to the font that is now supposed to be used for emphasis. Of course, this only works if the declarations in the list's entries actually change the font and not, for example, just the color. In such a scenario one has to add `\emforce` to the entry, which directs the mechanism to use the entry, even if the font attributes appear to be unchanged.

Providing font family substitutions

Given that pdfTeX can only handle fonts with up to 256 glyphs, a single font encoding can only support a few languages. The `T1` encoding, for example, does support many Latin-based scripts, but if you want to write in Greek or Cyrillic then you will need to switch encodings to `LGR` or `T2A`. Given that not every font family offers glyphs in such encodings, you may end up with some default family (e.g., Computer Modern) that doesn't blend in well with the chosen document font. For such cases NFSS now offers `\DeclareFontFamilySubstitution`, for example:

```
\DeclareFontFamilySubstitution{LGR}
     {Montserrat-LF}{IBMPlexSans-TLF}
```

tells LaTeX that if you are typesetting in the sans serif font `Montserrat-LF` and the Greek encoding `LGR` is asked for, then LaTeX should use `IBMPlexSans-TLF` to fulfill the encoding request.

The code is based on ideas from the substitutefont package by Günter Milde, but the implementation is different.

Providing all text companion symbols by default

The text companion encoding `TS1` was originally not available by default, but only when the textcomp package was loaded. The main reason for this was limited availability of fonts with this encoding other than Computer Modern; another was the memory restrictions back in the nineties. These days neither limitation remains, so with the 2020 release all the symbols provided with the textcomp package are available out of the box.

Furthermore, an intelligent substitution mechanism has been implemented so that glyphs missing in some fonts are automatically substituted with default glyphs that are sans serif if you typeset in `\textsf` and monospaced if you typeset using `\texttt`. In the past they were always taken from Computer Modern Roman if substitution was necessary.

This is most noticeable with `\oldstylenums` which are now taken from `TS1` so that you no longer get 1234 but `1234` when typesetting in sans serif fonts and `1234` when using typewriter fonts.

If there ever is a need to use the original (inferior) definition, then that remains available as `\legacyoldstylenums`; and to fully revert to the old behavior there is also `\UseLegacyTextSymbols`. The latter declaration reverts `\oldstylenums` and also changes the footnote symbols, such as `\textdagger`, `\textparagraph`, etc., to pick up their glyphs from the math fonts instead of the current text font (this means they always keep the same shape and do not nicely blend in with the text font).

With the text companion symbols as part of the kernel, it is normally no longer necessary to load the textcomp package, but for backwards compatibility this package will remain available. There is, however, one use case where it remains useful: if you load the package with the option `error` or `warn` then substitutions will change their behavior and result in a LaTeX error or a LaTeX warning (on the terminal), respectively. Without the package the substitution information only appears in the `.log` file. If you use the option `quiet`, then even the information in the transcript is suppressed (which is not really recommended).

New `alias` size function for use in `.fd` files

Most of the newer fonts supported in TeX have been set up with the `autoinst` tool by Marc Penninga. In the past, this program set up each font using the face name chosen by that font's designer, e.g., "`regular`", "`bold`", etc. These face names were then mapped by substitution to the standard NFSS series names, i.e., "`m`" or "`b`". As a result one got unnecessary substitution warnings such as "`Font T1/abc/bold/n not found, using T1/abc/b/n instead`".

We now provide a new NFSS size function, `alias`, that can and will be used by `autoinst` in the future. It provides the same functionality as the `subst` function but is less vocal about its actions, so that only significant font substitutions show up as warnings.

Suppress unnecessary font substitution warnings

Many sans serif fonts do not have real italics but usually only oblique/slanted shapes, so the substitution of slanted for italics is natural and in fact many designers talk about italic sans serif faces even if in reality they are oblique.

With nearly all sans serif font families, the LaTeX support files therefore silently substitute slanted if you ask for `\itshape` or `\textit`. This is also true for Computer Modern in `T1` encoding but in `OT1` you got a warning on the terminal even though there is nothing you can do about it. This has now been changed to an information message only, written to the `.log` file. *(github issue 172)*

Other changes to the LaTeX kernel

UTF-8 characters in package descriptions

In 2018 we made UTF-8 the default input encoding for LaTeX but we overlooked the case of non-ASCII characters in the short package descriptions used in declarations, e.g., in the optional argument to `\ProvidesPackage`. They worked (sometimes) before, but the switch to UTF-8 made them always generate an error. This has been corrected. *(github issue 52)*

Fix inconsistent hook setting when loading packages

As part of loading a package, the command `\`*package*`.sty-h@@k` gets defined. However, attempting to load a package a second time resulted in this hook becoming undefined again. Now the hook remains defined so that extra loading attempts do not change the state of LaTeX (relevant only to package developers). *(github issue 198)*

Avoid spurious warning if `LY1` is made the default encoding

Making `LY1` the default encoding, as is done by some font support packages, gave a spurious warning even if `\rmdefault` was changed first. This was corrected. *(github issue 199)*

Ensure that `\textbackslash` remains robust

In the last release we made most document-level commands robust, but `\textbackslash` became fragile again whenever `\raggedright` or similar typesetting was used. This has been fixed. *(github issue 203)*

Make math delimiters robust in a different way

Making math delimiters robust caused an issue in some situations. This has been corrected. This also involved a correction to amsmath. *(github issue 251)*

Allow more write streams with `filecontents` in LuaTeX

Most TeX engines only support a maximum of sixteen concurrently open write streams, and when those have been used up, then `filecontents` or any other code trying to open one will fail. In LuaTeX more write streams are available and those can also now be utilised. *(github issue 238)*

Allow spaces in `filecontents` option list

Leaving spaces or newlines in the option list prevented the options from being correctly recognized. This has been corrected. *(github issue 256)*

New `reverselist` Lua callback type

A new callback type, `reverselist`, was added: `post_mlist_to_hlist_filter` and `post_linebreak_filter` are now of this type.

Changes to packages in the graphics category

Make color & graphics user-level commands robust

Some of the user-level commands in color, graphics and graphicx, such as `\textcolor` or `\includegraphics`, were still fragile so didn't work in moving arguments. All of these are now robust. *(github issue 208)*

Changes to packages in the tools category

Fixed column depth in boxed `multicols`

The `multicols` environment was setting `\maxdepth` when splitting boxes; but, due to the way the internal interfaces of LaTeX are designed, it should have used `\@maxdepth` instead. As a result, balanced boxed multicols sometimes ended up having different heights even if they had exactly the same content. *(github issue 190)*

Ensure that `multicols` does not lose text

The `multicols` environment needs a set of consecutively numbered boxes to collect column material. The way those got allocated could result in disaster if other packages allocated most boxes below box 255 (which TeX always uses for the output page). In the original implementation that problem was avoided because one could only allocate box numbers below 255, but nowadays the LaTeX allocation routine allows allocating box numbers both below and above 255. So the assumption that when asking for, say, 20 boxes you always get a consecutive sequence of 20 box register numbers became no longer true: some of the column material could end up in box 255, where it would get overwritten. This has now been corrected by allocating all necessary boxes with numbers above 255 whenever there aren't enough lower-numbered registers available. *(github issue 237)*

Allow spaces in `\hhline` arguments

The `\hhline` command, which allows the specification of rule segments in `tabular` environments, now allows (but ignores) spaces between its tokens: so `\hhline{: = : =}` is now allowed and is equivalent to `\hhline{:=:=}`. This matches similar token arguments in LaTeX such as the `[h t p]` argument on floats. A similar change has been made to the extended `\hhline` command in the colortbl package. *(github issue 242)*

LaTeX requirements on engine primitives

Since the finalization of ε-TeX in 1999, a number of additional 'utility' primitives have been added to pdfTeX. Several of these are broadly useful and have been required by expl3 for some time, most notably `\pdfstrcmp`. Over time, a common set of these 'post-ε-TeX' primitives have been incorporated into XeTeX and (u)p-TeX; they were already available in LuaTeX.

A number of these additional primitives are needed to support new or improved functionality in LaTeX. This is seen for example in the improved UTF-8 handling, which uses `\ifincsname`. The following primitive functionality (which in LuaTeX may be achieved using Lua code) will therefore be *required* by the LaTeX kernel and core packages from the start of 2021:

- `\expanded`
- `\ifincsname`
- `\ifpdfprimitive`
- `\pdfcreationdate`
- `\pdfelapsedtime`
- `\pdffiledump`
- `\pdffilemoddate`
- `\pdffilesize`
- `\pdflastxpos`
- `\pdflastypos`
- `\pdfmdfivesum`
- `\pdfnormaldeviate`
- `\pdfpageheight`
- `\pdfpagewidth`
- `\pdfprimitive`
- `\pdfrandomseed`
- `\pdfresettimer`
- `\pdfsavepos`
- `\pdfsetrandomseed`
- `\pdfshellescape`
- `\pdfstrcmp`
- `\pdfuniformdeviate`

For ease of reference, these primitives will be referred to as the 'pdfTeX utilities'. With the exception of `\expanded`, these have been present in pdfTeX since the release of version 1.40.0 in 2007; `\expanded` was added for TeX Live 2019. Similarly, the full set of these utility primitives has been available in XeTeX from the 2019 TeX Live release, and has always been available in LuaTeX (some by Lua emulation). The Japanese pTeX and upTeX gained all of the above (except `\ifincsname`) for TeX Live 2019 and will both have that primitive also from the 2020 release onward.

At the same time, engines which are fully Unicode-capable must provide the following three primitives:

- `\Uchar`
- `\Ucharcat`
- `\Umathcode`

Note that it has become standard practice to check for Unicode-aware engines by using the existence of the `\Umathcode` primitive. As such, this is already a requirement: engines lacking these primitives cannot use Unicode features of the LaTeX 2_ε kernel or expl3. Note also that upTeX can handle Unicode but it is not classed as a Unicode engine by the base LaTeX code.

References

[1] Frank Mittelbach: *The LaTeX release workflow and the LaTeX dev formats*. In: TUGboat, 40#2, 2019. https://latex-project.org/publications/

[2] LaTeX Project Team: *LaTeX 2_ε font selection*. https://latex-project.org/help/documentation/fntguide.pdf

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

LaTeX News

Issue 32, October 2020 (LaTeX release 2020-10-01)

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 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 of 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 2_ε 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 has already been 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 redefinitions 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 documentation 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 2_ε 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}; this limitation has now been lifted. *(github issue 124)*

Correct Unicode value of `\=y` ($\bar{y}$)

The Unicode slot for $\bar{y}$ was incorrectly pointing to the slot for $\bar{Y}$. 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 differently 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 trivial 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`, `\\` 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\textheight)`.

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 a 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 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 247)*

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 were carried out by redefining some commands, e.g., `\seriesdefault`. In 2019 we introduced various extensions and with it new methods of customizing 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 and 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 metafamilies (`rm`, `sf` and `tt`) via `\DeclareFontSeriesDefault`, it is no longer easy to answer the question "am I typesetting in a bold context?". To help with this problem a new command was provided:

`\IfFontSeriesContextTF{`⟨*context*⟩`}`
`{`⟨*true code*⟩`}{`⟨*false code*⟩`}`

The ⟨*context*⟩ 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 ⟨*true code*⟩ or ⟨*false code*⟩ 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 to get an error message that option `X` is not

known to the package. This obviously incorrect error was due to a 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 parameter allowing for setting it to a value such as `1em minus 1em`, i.e., so that 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 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 previously had only internal commands (`\@ifpackagelater` and `\@ifclasslater`) for testing package or class names, but nothing reasonable 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 problematic spaces after `\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 has been fixed and spaces directly following the command `\verb` or `\verb*` are now ignored as elsewhere. *(github issue 327)*

Provide a way to copy robust commands...

With the previous LaTeX 2_ε 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 2_ε 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 2_ε 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 2_ε 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 up 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 `\shown` 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 behavior for `\-`

LaTeX changes `\-` to add a discretionary hyphen even if `\hyphenchar` is set to −1. This change is not necessary under LuaTeX because there `\-` is not affected by `\hyphenchar` in the first place. Therefore this behavior 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.

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

(github issue 303)

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 XeTeX 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 `\addcontentsline` writes 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. *(github issue 370)*

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 ...` was 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 now be interpreted in pdfTEX and LuaTEX in the same way as graphicx's `\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)*

Files with multiple parts
Sometimes one has a graphics file, say, `file.svg`, and converts it to another format to include it in LATEX and ends up with a file named `file.svg.png`. In previous releases, if the user did `\includegraphics{file.svg}`, an error would be raised and the graphics inclusion would fail due to the unknown `.svg` extension. The graphics package now checks if the given extension is known, and if it doesn't, it tries appending the known extensions until it finds a graphics file with a valid extension, otherwise it falls back to the file as requested. *(github issue 355)*

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 of `\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 this, `\vrefformat`, `\Vrefformat`, `\vrefrangeformat`, and `\fullrefformat` have been added to all languages. *(github issue 352)*

xr: Support for spaces in filenames
The command `\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 uses their extended commands to define `\std@minus` and `\std@equal`. This avoids a package like unicode-math having to patch the code in the begin document hook to change the commands.

Use LuaTEX primitives where applicable
For a number of years lualatex-math patched `\frac`, `\genfrac` and the `subarray` environment to make use of new luaTEX primitives. This code has now been integrated into amsmath.

Changes to the babel package

Multilingual typesetting has evolved greatly in recent 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, while 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 defining and redefining 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 Southeast Asian and CJK scripts, nonstandard 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 a wide variety of languages, such as Russian, Arabic, Hindi, Thai, Japanese, Bangla, Amharic, Greek, and many others.

In addition, since these `ini` files 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

LaTeX News

Issue 33, June 2021 (LaTeX release 2021-06-01)

Contents

Introduction

The focus of the June 2021 release is to provide further important building blocks for the future production of reliable tagged PDF output (see [1]); these enhancements are discussed in the next two sections.

Subsequent sections describe quite a number of recent smaller enhancements and fixes. As usual, more detail on individual changes can be found in the `changes.txt` files in the distribution and, of course, in the documented sources [2].

Extending the hook concept to paragraphs

Largely triggered by the need for better control of paragraph text processing, in particular when producing tagged PDF output, we have changed LaTeX so that the kernel gains control both at the start and at the end of each paragraph. This is done in a manner that is (or should be) transparent to both packages and documents.

Besides the addition of internal control points for the exclusive use of the LaTeX kernel, we also implemented four public hooks that can be used in packages or documents (via the normal hook management declarations) to achieve special effects, etc. Until now, such enhancements required redefinitions of `\everypar` or `\par`, which led to the usual issues since such changes can easily conflict with changes made by other packages.

The documentation of these new "paragraph hooks", together with a few examples, is in `ltpara-doc.pdf` and,

for those who want to study it, the (quite interesting) code can be found in `ltpara-code.pdf`. Additionally, both of these files are included as part of the full kernel documentation in `source2e.pdf`.

Extending the hook concept to commands

Up to now, hook management covered hooks for only a few core areas, such as the hooks for the `\shipout` process or those in the `document` environment, as well as some "generic" hooks, both for file loading (helpful for patching such files) and for arbitrary environments (the hooks executed within `\begin` and `\end`). This concept of "generic hooks" has now been extended to provide `/before` and `/after` hooks for any (document-level) command—in theory at least.

In practice, these new generic `cmd` hooks, especially the `cmd/.../after`, hooks may fail with commands that are too complex to be automatically patched, breaking if the hook contains any code. These restrictions are documented in `ltcmdhooks-doc.pdf`. However, given that these hooks are mainly meant for developers who wish to provide better interoperability between different packages, and between packages and the LaTeX kernel, these restrictions are, we hope, of minor importance. Indeed, for commands where this mechanism can't be applied, one is in the same situation as before; and for all others there will be a noticeable improvement.

These hooks will be especially important for our current project to provide accessible and tagged PDF output [1] because we will eventually have to patch many third-party packages, and this must be done in controlled and standardized ways.

Other hook business

Shipping out a page while bypassing hooks

In the 2020 October release, several hooks were added to control the process of constructing and shipping out a page box: these support, for example, the addition of background or foreground material to some or all pages.

We have now added a command, called `\RawShipout`, which does not do any rebuilding of the page box and so does not run most of these hooks. When using this new command, essential internal book-keeping is still carried out, such as updating the `totalpages` counter and adding `shipout/firstpage` or `shipout/lastpage` material when appropriate.

A new Lua callback in ltshipout, for custom attributes

For use just before shipping out a page, there is now a LuaTeX callback `pre_shipout_filter` to contain final adjustments to the box being shipped out. This is particularly useful for LuaTeX packages which flag (using, for example, attributes or properties) elements on a page in order to apply effects (such as the insertion of "color commands") to these elements at shipout.

Improved handling of file names

File names with spaces, multiple dots or UTF-8 characters

In one of the recent LaTeX releases we improved the interface for specifying file names so that they can now safely contain spaces (as is common these days), more than one dot character, and also UTF-8 characters outside the ASCII range. In the past this was only possible by applying a special syntax in the case of spaces, while file names with several dots often failed, as did most UTF-8 characters.

Consequences for file names in `\include`: TeX has a built-in rule saying that you can normally leave out the extension if it is `.tex`. Thus `\input{file}` and `\input{file.tex}` both load `file.tex` (if it exists). While this is convenient most of the time, it is a little awkward in some scenarios (for example, when both `file` and `file.tex` exist) and also when you manually try to implement the rule.

LaTeX therefore had one special syntax for `\include` and `\includeonly`: they always expected that their arguments contain a file name[1] with no extension given, so that it had to be `.tex`. Thus, when you mistakenly wrote `\include{mychap.tex}` (for example, because you changed from `\input` to `\include`), LaTeX went ahead and looked for the file `mychap.tex.tex` for inclusion and tried to use the file `mychap.tex.aux` for internal (auxiliary) information. The reason was that `\include` had to construct both of these file names from the given argument and it didn't bother to do anything special with the supplied extension `.tex`.

With the new implementation this has changed: the extension `.tex` now gets removed/ignored if it was supplied. Thus `\include{mychap.tex}` now no longer looks for `mychap.tex.tex` but loads `mychap.tex` and uses `mychap.aux`. *(github issue 486)*

Normalization of robust commands in file names

The handling of file names has been modified so that `\string` is applied to normalize robust commands within the file name. Previously, for example, `\input{\sqrt{2}}` would cause LaTeX to loop indefinitely whereas with the new normalization it looks for the file named `sqrt {2}.tex` (and therefore very likely reports "file not found"). *(github issue 481)*

Fix for `filecontents` with UTF-8 chars in the file name

Since a few releases back, the `filecontents` environment has allowed UTF-8 characters in the file name. There was, however, a bug that would not allow *over*writing a file with UTF-8 characters in its name. This has been fixed and now `filecontents` allows any characters in the file name. *(github issue 415)*

[1] In the case of `\includeonly`, a comma-separated list of such names.

Updates to the font selection scheme

A new hook in \selectfont

After \selectfont has changed the font, we now run a hook (selectfont) so that packages can make final adjustments. This functionality was originally provided by the everysel package but our implementation is slightly different and uses the standard hook management. *(github issue 444)*

Change of font series/shape delayed until \selectfont

With the NFSS extensions introduced in 2020, the font series and shape settings can be influenced by changes to the font family. The settings of these two are now therefore delayed until \selectfont is executed; this avoids unnecessary or incorrect substitutions that may otherwise happen due to the order of declarations. *(github issue 444)*

Glyphs, characters & encodings

Improved copy & paste for pdfTeX documents

When compiling with pdfTeX, additional information (from the file glyphtounicode.tex) is now added automatically to the PDF file in order to improve copying from, and searching in, text.

In particular, this allows the most common ligatures to be copied as intended from all generated PDF files without the need to explicitly load the package cmap. *(github issue 465)*

Support for more Unicode characters

LaTeX is quite capable of typesetting characters such as "m̥", but until now it could not access some Unicode characters from the Latin Extended Additional block. This meant that, for example, there were no Unicode mappings for some characters that are used to write Sanskrit words in Latin transliteration (as seen in books about yoga, Buddhist philosophy, etc.). These characters have now been added so that they can be entered directly instead of using \d{m}, etc. *(github issue 484)*

More "dashes" in encodings OT1, T1 and TU

When pasting in text from external sources, one can encounter these three Unicode characters "2011 (non-breaking hyphen), "2012 (figure dash) and "2015 (horizontal bar), in addition to the more common "2013 (en-dash) and "2014 (em-dash). In the past, these first three produced an error message when used with pdfTeX (since they are not available in OT1 or T1 encoded fonts). They now typeset an approximation to the glyph: e.g., the "figure dash" is approximated by an en-dash.

With Unicode engines they either work (when the glyph is contained in the selected Unicode font) or they typeset nothing, producing a "Missing character" warning in the log file.

With all engines these characters can also now be accessed using the command names \textnonbreakinghyphen, \textfiguredash and \texthorizontalbar, respectively. *(github issue 404)*

Poor man's \textasteriskcentered

The \textasteriskcentered symbol, used as part of the set of footnote symbols in LaTeX, is assumed to be implemented by every font with the TS1 encoding (when pdfTeX is used) or with the TU encoding for the Unicode engines. That assumption is unfortunately not correct for all fonts since, for example, the stix2 fonts don't provide this glyph. A result is that one gets missing glyph messages when using \thanks, etc.

Therefore \textasteriskcentered now checks whether there is such a glyph and, if not, uses a normal "*", but slightly enlarged and lowered. This may not be perfect in all cases, but it is certainly better than no glyph showing up. *(github issue 502)*

The characters from textcomp are in the kernel

A couple of releases back, the functionality of the textcomp package was integrated into the LaTeX kernel. Thus it is no longer necessary to load this package in order to access glyphs such as \textcopyright, \texteuro or \textyen.

At this time the opportunity was also taken to bring some order to the chaos surrounding the question: "which glyphs from the TS1 encoding are available in a given font?". This was done using an approach based on font families and collections, with the differing glyph coverage of the 'text symbols' being indicated by assigning to a font family or collection a "sub-encoding number" that indicates which glyphs from the TS1 encoding are guaranteed to be available when using a font from that family or collection. This assignment ensures that LaTeX always errs on the side of caution, possibly claiming that a glyph is not available even when it in fact is.

A note on the history of "text symbols" and the TS1 encoding: The "text symbol encoding" (TS1) was originally designed at the Cork Conference as a companion to the T1 encoding. In it various symbols that are not subject to hyphenation got assembled and the textcomp package was developed to make them accessible. Unfortunately the TeX community was a bit too enthusiastic and included several symbols only available in a few TeX fonts and some, such as the capital accents, not available at all but developed as part of the reference font implementation.

In hindsight that was a very bad idea because it meant that other existing fonts (at the time) and later new fonts that got developed were unable to provide the full set of glyphs that made up the TS1 encoding. For existing free PostScript fonts people took the extra effort

and produced virtual fonts that faked (some) of the missing glyphs. But this was and is a time-consuming effort so it was done for only a few basic fonts. But even then, only some fonts included all glyphs from TS1 so the textcomp already back then contained a long list, dividing fonts into 5 categories according to which glyphs were implemented and which were missing.

When we recently integrated the functionality of the textcomp into the LaTeX kernel many new free fonts had appeared and unfortunately the chaos around the question "which glyphs of the TS1 encoding are implemented by which font" had increased with it. Not only did one find many new holes, it was next to impossible to order the set of fonts into a reasonable set of sub-encodings that are contained in each other in a single sequence.

In the end we decided on nine or ten sub-encodings with a reasonable number of fonts in each so that all fonts implemented all glyphs of the sub-encoding they got mapped to. Thus when typesetting with a font one could be sure that a command like \textcopyleft would either typeset the requested character (if the glyph was part of the sub-encoding the font belonged to) or it would raise an error, saying that the glyph is unavailable in that font. The mapping would ensure that LaTeX always errs on the side of caution, because it might claim a glyph is unavailable even though in fact it is.

For example, the old pcr (PostScript Courier) font (as well as most other older PS fonts) is mapped to sub-encoding 5 and therefore claims that \textasciigrave is unavailable even though in fact for Courier this is not true. If one uses such a font and this becomes an issue then there are a couple (suboptimal) possibilities. For one, one can alter the mapping of Courier and pretend that belongs to a fuller sub-encoding, e.g.

```
\DeclareEncodingSubset{TS1}{pcr}{2}
```

The downside is, that LaTeX then believes other glyphs that are in fact unavailable are also there, so that it is important to check that the final document doesn't have some missing glyphs.

An alternative is to pretend that \textasciigrave can always be taken from the TS1 encoding (no questions asked):

```
\DeclareTextSymbolDefault{\textasciigrave}{TS1}
```

Again there is a danger that this is not true when it is used with a different font and would then generate a missing glyph.

Finally, and possibly the best solution, if not impossible for other reasons, is to simply use a different font, for example, to use the TeX Gyre Cursor font (a reimplementation of Courier with a much more complete glyph set).

New or improved commands

Adjusting itemize labels with \labelitemfont

The command \labelitemfont was introduced already with the LaTeX release 2020-02-02, but back then we forgot to describe it, so we do this now. Its purpose is to resolve some bad formatting issues with the itemize environment and also to make it easier to adjust the layout when necessary. What could happen in the past was that the itemize labels (e.g., the •) would sometimes react to surrounding font changes and could then suddenly change shape, for example to •.

This new command \labelitemfont, which defaults to \normalfont, can be used to provide additional control in the typesetting of each label. Thus by choosing different settings other effects can be achieved. Here are two examples:

```
\renewcommand\labelitemfont
   {\normalfont\fontfamily{lmss}\selectfont}
\renewcommand\labelitemfont
   {\rmfamily\normalshape}
```

The first definition will take the symbols from the font Latin Modern Sans, so that you get ▪, –, ∗ and · ; while the second variant freezes the font family and shape, but leaves the series as a variable quantity, so that an itemize in a bold context would show bolder symbols. Making \labelitemfont empty would give you back the buggy old behavior. *(github issue 497)*

Producing several marks for one footnote

It is sometimes necessary to reference the same footnote several times: i.e., to produce several footnote marks using the same number or symbol. This is now easily possible by placing a \label within the referenced \footnote and referencing this label by using the new command \footref. This means that footnote marks can be generated to refer to arbitrary footnotes (including those in minipages).

This \footref command has previously been available, but only when using certain classes or the footmisc package. *(github issue 482)*

Allow \nocite in the preamble

A natural place for \nocite{*} would be the preamble of the document, but for historical reasons LaTeX issued an error message if it was placed there. This command is now allowed in the preamble. *(github issue 424)*

Made \\ generally robust

In 2018 most LaTeX user-level commands were made robust, including the \\ command. However, \\ gets redefined in various environments and not all these cases were caught: such as, in particular, its use as the row delimiter in tabular structures. This has been corrected so that \\ should now be robust in all circumstances.

This change also fixed one anomaly present in the past: in a tabular preamble of the form

```
{l>{\raggedright}p{10cm}r}
```

a \\ in the second column would have the definition used within \raggedright and so it would not indicate the (premature) end of the tabular. Thus, for example,

```
a & b1 \\ b2 & c \\
```

was interpreted as a single row of the tabular (as intended), whereas

```
a &    \\ b2 & c \\
```

resulted in two rows! This happened because the \\ directly following the & got interpreted while it still had the "end the row" meaning and not yet the "start a new line within the second column" meaning.

With \\ now being robust, the special scanning mode initiated by the & ends immediately when this command is seen: the second column is therefore then started, which results in the \\ being interpreted as being within that column and hence as having its expected, within-column, meaning.

We have restored consistency here: now both of the above lines produce a single tabular row. As before, you can put \raggedright\arraybackslash in the tabular's preamble for a column to ensure that \\ is always interpreted as a tabular row separator when used in that column. And you can use \tabularnewline to explicitly ask for a new table row, even when \\ has a different meaning within the current column. *(github issue 548)*

Allow extra space between name and address in letter class

The \opening command in the letter class expects the name and address to be separated by \\, but it didn't allow the use of an optional argument to add some extra space after the name. The code has now been slightly altered to allow this. *(github issue 427)*

Additions to \tracingall

In July 2020 David Jones suggested an extension to TEX engines, that added the possibility to set \tracinglostchars=3 in order to generate an error message in case some character is missing from a font. In previous years, a warning about a missing character was silently printed to the .log file (if \tracinglostchars $>$ 0) and to the terminal (if $>$ 1). This extension was added for TEX Live and MiKTEX (except in Knuth's TEX, of course), so that with \tracinglostchars $>$ 2 you now also get an error message for each missing glyph.

Later, in January 2021, Petr Olšák suggested yet another extension: a new primitive parameter \tracingstacklevels that, when both it and \tracingmacros are positive, will add to the tracing information for each macro a visual indication (using dots) of its nesting level in the macro expansion stack.

These changes have both now been added to LATEX's debugging macros \tracingall and \tracingnone, so that these two new extensions are activated/deactivated as appropriate, so long as the TEX engine supports them. An example document demonstrating these parameters is in the linked GitHub issue. *(github issue 524)*

Code improvements

Execute \par at the end of \marginpar

Previously, LATEX ended a \marginpar without ever explicitly calling \par. This command is now explicitly added because it is essential to the correct working of the paragraph hooks.

Another case where this issue caused problems was the lineno package, where the last line was not numbered if the \marginpar ended without an explicit \par. *(github issue 489)*

Execute \AtEndDocument hook in vertical mode

Until now \end{document} executed the code from the \AtEndDocument hook as its first action. This meant that this hook was executed in horizontal mode if the user left no empty line after the last paragraph. As a result, one could get a spurious space added when, for example, that code contained a \write statement. This was fixed and now \enddocument first issues a \par to ensure that it always goes into vertical mode. *(github issue 385)*

Color groups made permanent

The use of color in certain LATEX constructs, especially boxes, needs an extra layer of grouping to ensure that the color setting does not *escape* and continue outside the box when it shouldn't. To support this, the LATEX kernel defines a number of commands, e.g., \color@begingroup to be used in such places.

Until now, these commands were initially set as no-ops and only the color packages redefined them to become real groups; this methodology complicates the coding as one has to account for a group being present or not (depending on what is loaded in the document). The kernel therefore now permanently adds these "color groups". *(github issue 488)*

Provide the raw option list to key/value option handlers

Before any further processing of the option list, the original (un-normalized, "raw" and unchanged) list of package or class options is now saved, as \@raw@opt@...; this list is not used by the standard option processing code but it is now available for use by extended class/package processing systems. Note that, for compatibility reasons, the standard option processing code has not been changed.

One aspect of this change does affect the standard processing: any tokens to the right of an = sign are removed from consideration when constructing the

"unused option list". For example, in this release `clip=true` and `clip=false` both contribute `clip` to the list of options that have been used. *(github issue 85)*

New for latexrelease: `\NewModuleRelease`
To explain the need for this new feature, we shall consider the following example: in the 2020-10-01 release, LaTeX's new hook management system was added to the kernel (see [3]) and, as with all changes to the kernel, it was added to latexrelease; this made it possible to roll back to a date where this module didn't yet exist, or to roll forward from an older LaTeX release to get the hook management system (by loading the latexrelease package). However, this method of rolling back from a later release to the 2020-10-01 release didn't quite work because it would try to define all the commands from lthooks again; and this would of course result in the expected errors from commands defined with `\newcommand` or (as in lthooks) `\cs_new:Npn`.

To solve such issues, we now provide `\NewModuleRelease` so that completely new modules can be defined using the facilities of latexrelease in such a way that, when rolling back or forward, the system will know whether the code of the new module has to be read or completely ignored. More details on this can be found in the latexrelease documentation (get this with `texdoc latexrelease`). *(github issue 479)*

Small fix for rolling back prior to 2020-02-02
Whereas the latexrelease package can usually emulate an older LaTeX kernel without much problem, rolling back to before the 2020-02-02 release didn't work properly: this is because the management of the `\ExplSyntaxOn/Off` status for packages (after an expl3-based package is loaded) cannot be removed by the rollback without messing up the catcodes. This has been fixed so that rollback is now more careful not to leave `\ExplSyntaxOn` after a package ends. *(github issue 504)*

Changes to packages in the graphics category

Removed warning when loading graphics files
A previous release sometimes mistakenly caused a (false) warning message to appear when using a generic graphics rule to find and load a graphics file with an unknown extension. This warning would incorrectly say that the file was not found, whereas the file would in fact be correctly loaded. The warning now doesn't show up in that case. *(github issue 516)*

Fixed loading of gzipped PostScript files
A previous release mistakenly changed the file searching mechanism so that compressed PostScript graphics files would raise an error when being loaded with `\includegraphics`. This has been fixed so that gzipped graphics files now load correctly. *(github issue 519)*

Changes to packages in the tools category

layout: Added language options
This package now recognizes `japanese` and `romanian` as language options. *(github issues 353 and 529)*

array and longtable: Make `\\` generally robust
The fix for this issue was also applied to these packages; see above. *(github issue 548)*

longtable: General bug fix update
This is a minor update to the longtable package that fixes several reported bugs: notably the possibility of incorrect page breaks when floats appear on the page where a `longtable` starts. As this may affect page breaking in existing documents, a rollback to longtable 4.13 (`longtable-2020-01-07.sty`) is supported. *(gnats issue tools/2914 3396 3512)* *(github issue 133 137 183 464 561)*

trace: Additions to `\traceon`
The `\tracingstacklevels` and `\tracinglostchars` extensions to `\tracingall` (see above) were also added to `\traceon` in the trace package, so its users can also benefit from these new debugging possibilities. *(github issue 524)*

bm: Better support for commands with optional arguments
Some uses of optional arguments in `\bm` stopped being supported (in 2004) when `\kernel@ifnextchar` was used internally by the format instead of `\@ifnextchar`. This update handles both versions of this command and restores the original behavior.

In addition, package options for guiding the use of "poor man's bold" in fallback situations were added. *(github issue 554)*

Changes to packages in the amsmath category

The fix for issue 548 was also applied in amsmath; see above. *(github issue 548)*

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 32.* https://latex-project.org/news/latex2e-news/ltnews32.pdf

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

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 font metafamilies 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 metafamilies 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

LaTeX News

Issue 35, June 2022 (LaTeX release 2022-06-01)

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 `ltmeta` (part of `source2e.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 2_ε) 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`.

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\obeyedline=\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

⟨*version*⟩ – ⟨*date*⟩

as the heading (instead of just ⟨*version*⟩), 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 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 `\specials` 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.

LaTeX News

Issue 36, November 2022 (LaTeX release 2022-11-01)

Contents

Introduction

The 2022-11 release of LaTeX is largely a consolidation release where we made a number of minor improvements to fix some bugs or improve one or the other interface.

The only really important functionality that was added is described in the next section: the ability to easily define document-level commands and environments that accept a key/value list in one of its (usually optional) arguments, including the ability to determine if the argument does in fact contain such a key/value list or just a single "classical" value.

For the "Tagged LaTeX Project" this functionality is very important because many document-level commands will need to accept such key/value lists, for example, to specify alternative text or overwrite default tagging if that becomes necessary in a document.

Auto-detecting key/value arguments

To allow extension of the core LaTeX syntax, ltcmd now supports a `=...` modifier when grabbing arguments. This modifier instructs LaTeX that the argument should be passed to the underlying code as a set of key/values. If the argument does not "look like" a set of key/values, it will be converted into a single key/value pair, with the argument to `=` specifying the name of that key. For example, the `\caption` command could be defined as

```
\DeclareDocumentCommand\caption
        {s ={short-text}+O{#3} +m}
        {...}
```

which would mean that if the optional argument does *not* contain key/value data, it will be converted to a single key/value pair with the key name `short-text`.

Arguments which begin with `=,` are always interpreted as key/values even if they do not contain further `=` signs. Any `=` signs enclosed within `$...$` or `\(...\)`, i.e. in inline math mode, are ignored, meaning that only `=` outside of math mode will generally cause interpretation as key/value material.

In case the argument contains a "textual" `=` sign that is mistaken as a key/value indicator you can hide it using a brace group as you would do in other places, e.g.,

```
\caption[{Use of = signs}]
        {Use of = signs in optional arguments}
```

However, because `=` signs in math mode are already ignored, this should seldom be necessary.

A note for font package developers

Encoding subsets for TS1 encoded fonts

The text companion encoding TS1 is unfortunately not very faithfully supported in fonts that are not close cousins to the Computer Modern fonts. It was therefore necessary to provide the notion of "sub-encodings" on a per font basis. These sub-encodings are declared for a

font family with the help of a `\DeclareEncodingSubset` declaration, see [5] for details.

Maintainers of font bundles that include `TS1` encoded font files should add an appropriate declaration into the corresponding `ts1`*family*`.fd` file, because otherwise the default subencoding is assumed, which is probably disabling too many glyphs that are actually available in the font.[1] *(github issue 905)*

New or improved commands

Better language handling for case-changing commands

The commands `\MakeUppercase`, `\MakeLowercase` and `\MakeTitlecase` now automatically detect the locale currently in use when `babel` is loaded. This allows automatic adjustment of letter mappings where appropriate. They also accept a leading optional argument. This accepts a key–value list of control settings. At present, there is one key available: `locale`, which can also be accessed via the alias `lang`. This is intended to allow local setting of the language, which can be done using a BCP-47 descriptor. For example, this could be used to force Turkish case changing in otherwise English input

```
\MakeUppercase[lang = tr]{Ragıp Hulûsi Özdem}
```

yields RAGIP HULÛSİ ÖZDEM.

Code improvements

Support for slanted small caps in the EC fonts

For some time LaTeX has supported the combination of the shapes small caps and italic/slanted. The EC fonts contain slanted small caps fonts but using them required the loading of an external package. Suitable font definitions have now been added to `t1cmd.fd` and so from now on

```
\usepackage[T1]{fontenc}
...
 \textsc{\textsl{Slanted Small Caps}};
 \textsc{\textit{Italic Small Caps}};
 \bfseries
 \textsc{\textsl{Bold Slanted Small Caps}};
 \textsc{\textit{Bold Italic Small Caps}}.
```

will give the expected result: *Slanted Small Caps*; *Italic Small Caps*; ***Bold Slanted Small Caps***; ***Bold Italic Small Caps***.

Given that the Computer Modern fonts in `T1` do not have real italic small caps but only slanted small caps, the latter is substituted for the former. This is why both work in the above, but there is no difference between the two (and you get a substitution warning for the `\textit\textsc` shape combination). *(github issue 782)*

EC sans serif at small sizes

The EC (T1 encoded Computer Modern) sans serif fonts have errors at small sizes: the medium weight is bolder and wider than the bold extended. This makes them unusable at these small sizes. The default `.fd` file has therefore been adjusted to use a scaled down 8pt font instead. *(github issue 879)*

Improve font series handling with incorrect `.fd` files

By convention, the font series value is supposed to contain no `m`, unless you refer to the "medium" series (which is represented by a single `m`). For example, one should write `c` for "medium weight, condensed width" and not `mc`. This was one of the many space-conserving methods necessary in the early days of LaTeX 2$_\varepsilon$.

Some older `.fd` files do not obey that convention but use `mc`, `bm`, etc., in their declarations. As a result, some font selection scheme functionality was not working when confronted with such `.fd` files. We have therefore augmented `\DeclareSymbolFont` and `\SetSymbolFont` to strip any surplus `m` from their series argument so that they do not unnecessarily trigger font substitutions. Regardless of this support such `.fd` files should get fixed by their maintainers. *(github issue 918)*

Detect nested `minipage` environments

Nesting of `minipage` environments is only partially supported in LaTeX and can lead to incorrect output, such as overfull boxes or footnotes appearing in the wrong place; see [1, p. 106]. However, until now there was no warning if that happened. This has been changed and the environment now warns if you nest it in another `minipage` environment that already contains footnotes. *(github issue 168)*

Robust commands in package options

With the standard key-based option handler added in the last release, or with contributed packages offering similar features, users may expect to be able to use a package option such as `[font=\bfseries]`. Previously this failed with internal errors as the option list was expanded via `\edef`. This has now been changed to use the existing command `\protected@edef` so that any LaTeX robust command should be safe to pass to a key value option. *(github issue 932)*

Improve l3docstrip integration into docstrip

In 2020 we merged l3docstrip.tex into docstrip.tex to support the `%<@@=`⟨*module*⟩`>` syntax of expl3; see [2]. However, this support was incomplete, because it didn't cover docstrip lines of the form `%<+...>` or `%<-...>`. This was never noticed until now, because usually `%<*...>` blocks are used. Now all lines in a `.dtx` file are subject to the `@@` replacement approach. *(github issue 903)*

[1] The LaTeX format contains declarations for many font families already. This was done in 2020 to quickstart the use of the symbols in the kernel, but it is really the wrong place for such declarations. Thus, for new fonts the declarations should be placed into the corresponding `.fd` files.

LuaTeX callback efficiency improvement

The mechanism for providing the `pre/post_mlist_to_hlist_filter` callbacks in LuaTeX has been improved to make it more reusable and to avoid overhead if these callbacks are not used. *(github issue 830)*

Rule-based ordering for LuaTeX callback handlers

In LuaLaTeX the callback handlers used to be called in the order in which they were registered in, but this was often rather fragile. It depends a lot on the load order and any attempts to enforce a different order required unregistering and reregistering the handlers to be reordered. Additionally, even if some ordering constraints where enforced that way, another package loaded later could accidentally overwrite it.

To improve this, we now order the callback handlers based on ordering rules similar to the hook rules.

When registering a callback which should run before or after another callback, `luatexbase.declare_callback_rule` can now be used to record this ordering constraint. For example

```
luatexbase.add_to_callback
 ('pre_shaping_filter', my_handler, 'my_name')
luatexbase.declare_callback_rule
 ('pre_shaping_filter',
            'my_name', 'before', 'other_name')
```

will ensure that `my_handler` will always be called before the handler registered as `other_name`.

This also means that the order in which callbacks are registered no longer implicitly defines an order. Code which relied on this implicit order should now define the order rules explicitly.

Bug fixes

Prevent TeX from losing a `\smash`

When TeX is typesetting a fraction, it will rebox the material in either the numerator or denominator, depending on which is wider. If the repackaged part consists of a single box, that box gets new dimensions and if it was built using a `\smash` that effect vanishes (because a smash is nothing other than zeroing some box dimension, which now got undone). For example, in the line

```
\frac{1}{2} = \frac{1}{\smash{2^X}}
            \neq \frac{100}{\smash{2^X}}
```

the 2 in the denominators was not always at the same vertical position, because the second `\smash` was ignored due to reboxing:

$$\frac{1}{2} = \frac{1}{2^X} \neq \frac{100}{2^X}$$

The differences are subtle but noticeable. This is now corrected and the `\smash` is always honored. Thus now you get this output:

$$\frac{1}{2} = \frac{1}{2^X} \neq \frac{100}{2^X}$$

(github issue 517)

Resolve an issue with `\mathchoice` and `localalphabets`

The code for keeping a number of math alphabets local (introduced in 2021; see [3]) used `\aftergroup` to do some cleanup actions after a formula had finished. Unfortunately, `\aftergroup` can't be used inside the arguments of the `\mathchoice` primitive and as a result one got low-level errors if the freezing happened in such a place. The implementation was therefore revised to avoid the `\aftergroup` approach altogether. *(github issue 921)*

Reporting of unused global options when using key/value processing

Using the new key/value option processor did not properly report any unused global options when it was used in handling class options. This has now been corrected. *(github issue 938)*

Changes to packages in the graphics category

Fix a `\mathcolor` bug

The `\mathcolor` command introduced in [4] needs to scan for following sub- and superscripts, but if it did so at the end of an alignment cell, e.g., in a `array` environment, the `&` was evaluated too early, causing some internal errors. This is now properly guarded for. *(github issue 901)*

Changes to packages in the tools category

array: Correctly identify single-line m-cells

Cells in m-columns that contain only a single line are supposed to behave like single-line p-cells and align at the same baseline. To test for the condition, array used to compare the height of the cell to the height of the strut used for the table rows. However, the height of that strut depends on the setting of `\arraystretch` and if you made this negative (or very large) the test came out wrong. Therefore, we now test against the height of a normal strut to ensure that single-line cells are correctly identified as such (unless their content is truly very tall, in which case aligning is pointless anyway). *(github issue 766)*

References

[1] Leslie Lamport. LaTeX: A Document Preparation System: User's Guide and Reference Manual. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

[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] LaTeX Project Team: *LaTeX 2ε news 35*.
https://latex-project.org/news/latex2e-news/ltnews35.pdf

[5] LaTeX Project Team: *LaTeX 2ε font selection*.
https://latex-project.org/help/documentation/

LaTeX News

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

Contents

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

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

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

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

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

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

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

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

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

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

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

New or improved commands

Extending hooks to take arguments

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

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

To declare a hook with arguments, use

```
\NewHookWithArguments {<hook>} {<num-args>}
```

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

```
\UseHookWithArguments {<hook>} {2} {<arg1>} {<arg2>}
```

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

```
\AddToHookWithArguments {<hook>} [<label>] {<code>}
```

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

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

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

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

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

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

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

Providing copy and show functions for environments

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

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

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

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

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

(github issue 963)

`\IfFileAtLeastTF`

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

(github issue 1015)

`\DeclareLowercaseMapping`, `\DeclareTitlecaseMapping` and `\DeclareUppercaseMapping`

The move from a case-changing approach using `\lccode` and `\uccode` data to one where information is stored by a kernel-managed structure left a gap in the ability of

the user to *tune* the case changing outcomes. This has now been addressed by the addition of three commands

- `\DeclareLowercaseMapping`
- `\DeclareTitlecaseMapping`
- `\DeclareUppercaseMapping`

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

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

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

`\BCPdata`

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

Improve `\samepage`

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

Groups in `\MakeUppercase`

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

Extension of the `\label` command

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

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

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

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

Code improvements

Performance in checking file existence

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

doc: Handle `\␣` correctly in the index

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

doc: Support the upquote package

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

```
\verb/‘prog ’my input’‘/
```

is rendered as ‘`prog ’my input’‘` and not as `` `prog 'my input'` `` as preferred by many programmers.

This can be adjusted, for example, with the upquote package, which results in the second output. However, for historical reasons doc had its own definition of

\verb and verbatim and as a consequence the two packages did not cooperate. This has now been fixed and loading upquote together with doc has the desired effect. *(github issue 953)*

Default definition for \do

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

New key for filecontents

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

A further hook for shipping out pages

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

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

Displaying release information in the .log

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

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

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

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

Bug fixes

Incompatibility between doc and unicode-math

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

A fix for \hspace

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

Ensure that \cs is defined in ltxdoc

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

Improve spacing at top of minipages

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

With the new paragraph handling introduced in 2021 it is now finally possible to detect and avoid this problem and apply appropriate counter measures so that from now on the spacing will always be correct. *(github issue 989)*

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

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

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

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

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

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

Documentation improvements

Updates to the guides
When LaTeX 2$_\varepsilon$ was released, the team provided documentation for both document authors and package/class developers in the two files `usrguide` and `clsguide`. Over time, the team have augmented these documents as new methods have been added to the kernel. However, they retained their structure as assuming familiarity with LaTeX 2.09. This meant that for new users, there was material which is no longer relevant, and less clarity than desirable regarding the approaches that are recommended today.

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

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

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

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

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

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

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

Since the publication of the last edition (2004), a lot has happened in the LaTeX world and thus a complete rewrite was necessary. All chapters have been thoroughly revised, and in many cases significantly extended, to describe new important functionality and features. More than 5,000 add-on packages have been analyzed in detail, out of which roughly 10% have been chosen for inclusion in *The LaTeX Companion*. All important aspects of these packages are described to provide the user once

again with a satisfying one-stop–shop experience for the decade to come.

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

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

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

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

Changes to packages in the tools category

multicol: Better support for CJK languages

The default minimum depth of each column in a `multicols` corresponds to the depth of a "p" in the current font. This helps to get some uniformity if rules are used between the columns and makes sense for Latin-based languages. Until now it was hard-wired, but for CJK (Chinese/Japanese/Korean) languages it is better to use a zero depth, because there all characters have the same height and depth. And even with Latin-based languages one might want to use the depth of a `\strut` or that of a parenthesis. So we now offer a way to adjust this while maintaining backward compatibility: redefine `\multicolmindepthstring` to hold whatever you want to get measured for its depth (the width is not relevant). *(github issue 698)*

multicol: Fix handling of nested environments

If `multicols` environments have been nested into each other (the inner one boxed) it could fail if the boxed environment appeared near a page break. The problem was that the output routine was called while the `\hsize` was still altered to fit the column width of the inner `multicols` — thereby messing up the placement of columns of the page. This has now been fixed. *(github issue 1002)*

References

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

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

LaTeX News

Issue 38, November 2023 (LaTeX release 2023-11-01)

Contents

News from the "LaTeX Tagged PDF" project

The multi-year project to automatically tag LaTeX documents in order to make them accessible [3] is progressing steadily (at this point in time mainly as experimental `latex-lab` code).

Just recently we added support for automatic tagging of tabular structures including environments from tabularx and longtable. The code is still in its early stages and lacks configuration possibilities—these will be added in the future.

Approaching an important milestone

Nevertheless, with this new addition we are more or less able to automatically tag any document that confines itself to the commands and environments described in Leslie Lamport's *LaTeX Manual* [1] by simply adding a single configuration line at the top.

In addition, a number of extension packages that go beyond Lamport are already supported, most importantly perhaps amsmath (providing extended math capabilities) and hyperref (enhancing LaTeX with interactive hyperlinking features). Also already supported are some of the major bibliography support packages such as natbib and biblatex.

For now activation is done through the line

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

The math and the tabular support are not yet incorporated into `phase-III` but need their own activation, so that we can better experiment with additions and code adjustments.

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

A GitHub repository dedicated to the project

We have also started a new GitHub repository mainly intended for reporting issues, and offering a platform for discussions. For example, there is one discussion on ways to extend the LaTeX `tabular` syntax to allow describing the logical structure of tables (e.g., which cells are header cells, etc.).

Having all issues and discussions related to the project in a single place instead of being spread across multiple repositories such as latex2e, latex3, tagpdf, hyperref, pdfresources, etc., helps people to find information easily and report any issue related to the project without needing to know in which code repository the problematic code resides.

You find this repository at https://github.com/latex3/tagging-project and the mentioned discussion on `tabular` syntax at https://github.com/latex3/tagging-project/discussions/1.

Your feedback is important and reporting what doesn't yet work is beneficial to all users, so we hope to see you there and thank you for any contribution, whether it is an issue or a post on a discussion thread.

Hooks, sockets and plugs

In previous releases of LaTeX we introduced the general concept of hooks (both specific and generic ones). These are places in the code into which different packages (or the user in the document preamble) can safely add their own code to extend the functionality of existing commands and environments without the need to overwrite or patch them in incompatible ways. An important feature of such hooks is that the code chunks added by different packages can be ordered by rules, if necessary, thereby avoiding problems arising from

differences in package loading order. See LaTeX News issues 32–34 [2] for more information.

However, sometimes you need a kind of "hook" into which only a single chunk of code is placed at any time.[1] For example, there is code that implements footnote placement in relation to bottom floats (above or below them). But at any time in the document only one such placement code can be in force. Or consider the extra code needed for making LaTeX documents accessible (e.g., adding tags to the PDF output). Such code is either there (perhaps in alternative versions) or not at all, but it cannot have code from other packages added at the same point interfering with the algorithm.

For these use cases we now introduce the concept of sockets and plugs. A socket is a place in the code into which one can put a plug (a chunk of code with a name) after which the socket is in use; to put in a different plug, the former one has to be taken out first.[2] A socket may or may not have inputs that can then be used by the plugs. While this is technically not much different to putting a command in the code and at some point alter its definition, the advantage is that this offers a consistent interface, allows for status information, supports tracing, etc.

You declare a new socket and possibly some plugs for it with

`\NewSocket{`⟨*socket name*⟩`}{`⟨*# of inputs*⟩`}`
`\NewSocketPlug{`⟨*socket name*⟩`}{`⟨*plug name*⟩`}{`⟨*code*⟩`}`

For example, after the declaration `\NewSocket{foo}{0}` you can immediately use this socket in your code with `\UseSocket{foo}`. The `\NewSocket` declaration automatically defines a simple plug with the name `noop` for the socket and assigns it to the socket (plugs it in), thus your `\UseSocket` sits idle doing nothing[3] until you assign it a different plug, which is done with `\AssignSocketPlug`. This takes the current plug out and puts the new one in. All the declarations and commands are also available in the LaTeX3 programming layer as `\socket_new:nn`, `\socket_new_plug:nnn`, etc.

With this concept we can, for example, add tagging support for the "LaTeX Tagged PDF" project to various packages without altering their behavior if the tagging code is inactive. Activating one or the other form of tagging then just means to assign named plugs to the different sockets.

This is just a brief introduction to the mechanism; for more detailed documentation see `texdoc ltsockets-doc`.

[1] While this is in theory possible to model with the existing hook mechanism, it is inefficient and cumbersome.

[2] Think of electric outlets and plugging something into them.

[3] Sockets with one input also define an `identity` plug and initially assign that to the socket—this means that their input is simply returned without processing.

Document properties and cross-referencing

Traditional LaTeX uses `\label{`⟨*key*⟩`}` to record the values of two "local" properties of the document: the textual representations of the *current page number* and the *current* `\ref` *value* set by `\refstepcounter` declarations [1, p. 209]. (These declarations are issued, for example, by sectioning commands, by numbered environments like `equation`, and by `\item` in an `enumerate` or similar environment.)

These two recorded values can then be accessed and typeset (from anywhere in the next run of the document) by use of the (non-expandable) commands `\ref` and `\pageref` using the *key* that was specified as the argument to `\label` when recording these values. This supported basic cross-referencing (within a document), using these recorded values to provide both page-related and counter-related information (such as the page xvii or the subsection number 4.5.2).[4]

Over the years LaTeX packages have appeared that extend this basic "label-ref system" in various ways. For example, the refcount package made a small but significant change to the functions used to access recorded values, by making them expandable. And the smart-ref package supports the storage of a larger collection of counter values so that, for example, a cross-reference can refer to the relevant chapter together with an equation tag. The cleveref package stores (by means of a second, internal "logical label") extra information such as the name of the counter. The hyperref package adds the `\autoref` command, which tries to retrieve the name of a counter from the *logical name* used for a link target. The tikzmarks library records information about *labelled positions* on the page when using tikz. Finally, the zref package implements many related ideas, including a general idea of properties and lists of properties, with methods to record, and subsequently access, the value of any declared property.

Starting with this release, the LaTeX kernel provides handling of general document properties as a core functionality with standard interfaces. This is based on concepts introduced by the zref package but with some differences in detail, particularly in the implementation. It supports the declaration of new properties, and the recording of the values of any list of properties. These values are retrieved expandably.

To set up a new property that is the current chapter number, for example, here is the declaration to use.

```
\NewProperty{chapter}{now}{?}{\thechapter}
```

[4] In the Spring 2023 release of LaTeX, the `\label` command was extended to record, in addition, both a title (such as the text used in a section head) and the *logical name* used for an associated link target provided these have been set by packages such as nameref or hyperref.

The second argument means that the property value will be recorded immediately ("now"), and not "during the next `\shipout`". The third argument sets a default to be used when, for example, an unknown label is supplied. The final argument contains the code that will, as part of the recording process, be expanded to obtain the value to record for this property.

Then, to record the value of this new property, together with others, use this command.

```
\RecordProperties{mylabel}
                 {chapter,page,label}
```

This records the current values for the properties `chapter`, `page`, and `label`, using `mylabel` as the label, or *key*, for the record.

To *reference* (i.e., retrieve) this recorded value for use in a cross-reference to this chapter, use the `\RefProperty` command with two arguments: the label, or *key*, and the property.

```
\RefProperty{mylabel}{chapter}
```

The LaTeX kernel itself contains declarations for some generally useful properties, including these:

`label` the textual representation of the *current* `\ref` *value*, see above;

`page` the textual representation of the page number for the page currently under construction;

`title` the title, if set by, e.g., nameref;

`target` the logical name of the associated link target, if set by, e.g., hyperref;

`pagetarget` the logical name of the target added by hyperref at the origin of each shipped out page;

`pagenum` the value of the LaTeX counter `page` in Arabic numerals;

`abspage` the absolute page number of the page under construction, i.e., one more than the number of pages shipped out so far (thus it starts at 1 and is increased by 1 whenever a page is shipped out);

`counter` the name of the counter that produced the *current* `\ref` *value*, i.e., the counter that was stepped in the most recent `\refstepcounter` within the current scope;

`xpos, ypos` the position on the shipped out page as set by the most recent `\pdfsavepos`: recording these properties should be done as soon as possible after saving the position.

Both LaTeX 2$_\varepsilon$ commands (using camel-case names) and LaTeX3 programming layer commands are provided. For a more complete documentation, see `texdoc ltproperties-doc`.

New or improved commands

Testing for the LaTeX3 programming layer version: `\IfExplAtLeastTF`

The integration of expl3 (the LaTeX3 programming layer) into the kernel means that programmers can use all of the features available without needing to load it explicitly. However, as expl3 is upgraded separately from LaTeX 2$_\varepsilon$ and is not a separate package, its version is different from that of LaTeX 2$_\varepsilon$ and cannot be tested using `\IfPackageAtLeastTF`. To date, low-level methods have therefore been needed to check for the availability of features in expl3. We have now added `\IfExplAtLeastTF` as a test working in the same way as `\IfPackageAtLeastTF` but focused on the pre-loaded programming layer. Programmers can check the date of expl3 they are using in the `.log`, as it appears both at the start and end in the format

```
L3 programming layer <YYYY-MM-DD>
```

just after the line which identifies the format (`LaTeX2e`, etc.). *(github issue 1004)*

Code improvements

Support for tabs in `\verb*` and `verbatim*`

LaTeX converts a single tab to a single space, which is then treated like a "real" space in typesetting. The same has been true to date inside `\verb`, but was done in a way that meant that they remained as normal spaces even in `\verb*`, etc. We have now adjusted the code so that tabs are retained within the argument to `\verb` and `\verb*`, and the `verbatim` and `verbatim*` environments, independently from spaces, and are set up to print in the same way spaces do. This means that they now generate visible spaces inside `\verb*` and `verbatim*`, and their behavior can be adjusted if required to be different from that of spaces. *(github issue 1085)*

Improved argument checking for box commands

Previously if an alignment option had an unexpected value, such as `\makebox[4cm][x]{text}`, no warning was given but the box content was silently discarded. This will now produce a warning and act like the default `c` alignment. `\framebox` and `\parbox` have a similar change. *(github issue 1072)*

Aligning status of tilde with other active characters

Some time ago we revised the definition of active characters in pdfTeX to allow the full range of UTF-8 codepoints to be used in for example labels, file names, etc. However, `~` was not changed at that point as it is active independent of the engine in use. This has now been corrected: the definition of `~` is an engine-protected one which gives the string version of the character if used inside a csname.

In the programming layer

In the programming layer (expl3), we have revised the behavior of the titlecasing function to enable this to either titlecase only the first word of the input, or to titlecase every word. This should be transparent at the document level but will be useful for programmers.

We have also added the ability to define variables and functions inside `\fpeval` (at the expl3 level this is `\fp_eval:n`). This allows programmers to create non-standard functions that can then be used inside `\fpeval`. For example, this could be used to create a new function `dinner`:

```
\ExplSyntaxOn
\fp_new_variable:n{duck}
\fp_new_function:n{dinner}
\fp_set_function:nnn{dinner}{duck}
                    {duck - 0.25 * duck}
\fp_set_variable:nn{duck}{1}
$\fp_eval:n{duck}
 >\fp_eval:n{dinner(duck)}
  \fp_set_variable:nn{duck}{dinner(duck)}
 >\fp_eval:n{dinner(duck)}
  \fp_set_variable:nn{duck}{dinner(duck)}
 >\fp_eval:n{dinner(duck)}
  \fp_set_variable:nn{duck}{dinner(duck)}
 >\fp_eval:n{dinner(duck)}
$
\ExplSyntaxOff
```

The computation above would then generate

$$1 > 0.75 > 0.5625 > 0.421875 > 0.31640625$$

Users will be able to access added functions without needing to use the expl3 layer. It is possible that a future release of LaTeX will add the ability to create and set floating point variables at the document level: this will be examined based on feedback on the utility of the programming layer change.

Removed *kernel commands*

It is very rare that commands are removed from the LaTeX kernel. However, in this release we have elected to remove `\GetDocumentCommandArgSpec`, `\GetDocumentEnvironmentArgSpec`, `\ShowDocumentCommandArgSpec` and `\ShowDocumentEnvironmentArgSpec` from the kernel. These commands have been moved back to the "stub" xparse provided in l3packages. The reason for this change is that the removed commands exposed implementation details. They were essentially debugging tools which with hindsight should not have been made available directly in the kernel.

Changes to packages in the tools category

longtable: correct p-column definition

In general the longtable implementation follows the array usage but the package didn't take over a change made 1992 in array which adjusted the handling of the strut inserted at the begin of p-columns. As a consequence there are a number of inconsistencies in the output of p-columns between tabular and longtable. This has been corrected; longtable now uses for the strut the same definition as array. *(github issue 1128)*

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

[2] LaTeX Project Team. *LaTeX 2ε news 1–38*. https://latex-project.org/news/latex2e-news/ltnews.pdf

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

LaTeX News

Issue 39, June 2024 (LaTeX release 2024-06-01)

Contents

Introduction

The LaTeX Project team remains strongly focused on producing automatically tagged PDF output for accessibility and reuse. At the beginning of 2024 the ISO PDF/UA-2 and the WTPDF (well-tagged PDF) standards were released and we are glad to be able to report that it is now possible to use LaTeX to automatically produce documents that conform to these new standards.[1] A sample collection of such documents ranging from classical texts, such as the Bible, to recent technical papers submitted to arXiv.org can be found at https://github.com/latex3/tagging-project/discussions/72.

In February Ulrike and Frank presented the current project status during the 5th International Workshop on "Digitization and E-Inclusion in Mathematics and Science 2024" (DEIMS 2024) at Nihon University, Tokyo, Japan; see [8].

News from the "LaTeX Tagged PDF" project

In the previous LaTeX News [7] we announced some prototype support for tagged tabulars. Some of the necessary code has now been moved from `latex-lab` to the corresponding packages (using sockets and plugs) and to the LaTeX kernel (for those parts that are also necessary for other aspects of tagging).

The kernel code specific to tagging is implemented in the file `lttagging.dtx`. For now it contains `\UseTaggingSocket`, a special invocation command for

[1] At the present time we are still in a trial/prototype phase in which only a limited set of document classes and packages are supported. Over the next releases we expect to gradually lift these restrictions and eventually provide the full functionality as part of the core distribution, rather than through `latex-lab` modules.

sockets that are specific to tagging. This enables us to also provide `\SuspendTagging` and `\ResumeTagging`, i.e., a very efficient way to temporarily disable the whole tagging process. This is, for example, necessary if some code is doing trial typesetting. In that case the trials should not generate tagging structures—only the finally-chosen version should. Thus, tabularx, for example, stops the tagging while doing its trials to figure out the correct column widths to use, and then re-enables tagging when the table is finally typeset.

Over time, `lttagging.dtx` will hold more general tagging code as appropriate. For now it is only documented as part of `source2e.pdf` but long term we will provide a separate guide for tagging, which will then also include the information currently found in various other places, e.g., `tagpdf.pdf`.

We also added support for a few missing commands described in Leslie Lamport's *LaTeX Manual* [1]: If `phase-III` is used the `\marginpar` command will be properly tagged (depending on the PDF version) as an `Aside` or a `Note` structure. In the standard classes `\maketitle` will be tagged if the additional testphase module `title` is used.

The `math` module has been extended and now includes options to attach MathML files to the structures. First tests with a PDF reader and screen reader that support associated files look very promising. Examples of PDF files tagged with the new method can be found at https://github.com/latex3/tagging-project/discussions/72.

At last various small bugs and problems reported at https://github.com/latex3/tagging-project have been fixed. Such feedback is very valuable, so we hope to see you there and thank you for any contribution, whether it is an issue or a post on a discussion thread.

Enhancements to the new mark mechanism

In June 2022 we introduced a new mark mechanism [2, p. 76] that allows keeping track of multiple independent marks. It also properly supports top marks, something that wasn't reliably possible with LaTeX before.

There was, however, one limitation: to retrieve the marks from the page data it was necessary to `\vsplit` that data artificially so that TeX would produce split marks that the mechanism could then use. Unfortunately, TeX gets very upset if it finds infinite negative glue (e.g., from `\vss`) within this data. This is not totally surprising because such glue would allow splitting off any amount of material as such glue would hide its size. TeX therefore responds with an error message if it find such glue while doing a `\vsplit` operation (and it does so even if a later glue item cancels the infinite glue).

To account for this, the code in 2022 attempted to detect this situation beforehand and if so did not do any splitting but, of course, it would then also not extract any mark information.

In this release the approach has been changed and we always do a `\vsplit` operation and thus always get the right mark data extracted. While it is not possible to avoid upsetting TeX in case we have infinite negative glue present, it is possible to hide this (more or less) from the user.[2] With the new code TeX will neither stop nor show anything on the terminal. What we can't do, though, is avoid an error being written to the log file, but to make it clear that this error is harmless and should be ignored we have arranged the code so that the error message, if it is issued, takes the following format:

```
! Infinite glue shrinkage found in box being split.
<argument> Infinite shrink error above ignored !
l. ... }
```

Not perfect (especially the somewhat unmotivated `<argument>`), but you can only do so much when error messages and their texts are hard-wired in the engine.

So why all this? There are two reasons: we do not lose marks in edge cases any more, and perhaps more importantly we are now also reliably able to extract marks from arbitrarily boxed data, something that wasn't possible at all before. This is necessary, for example, to support extended marks in `multicols` environments or extract them from floats, marginpars, etc.

Details about the implementation can be found in `texdoc ltmarks-code` or in the shorter `texdoc ltmarks-doc` (which only describes the general concepts and the command interfaces).

Providing xtemplate in the format

In LaTeX News 32, we described the move of one long-term experimental idea into the kernel: the package xparse, which was integrated as ltcmd. With this edition, we move another long-term development idea to stable status: *templates*.

In this context, templates are a mechanism to abstract out various elements of a document (such as "sectioning") in such a way that different implementations can be interchanged, and design decisions can be implemented efficiently and controllably.

In contrast to ltcmd, which provides a mechanism that many document authors will exploit routinely, templates are a more specialised tool. We anticipate that they will be used by a small number of programmers, providing

[2] A note to l3build users that make use of its testing capabilities: the new mechanism temporarily changes `\interactionmode` and, for implementation reasons in TeX, that results in extra newlines in the `.log` file, so instead of seeing `[1] [2]` you will see each on separate lines. This means that test files might show differences of that nature, once the code is active, and must therefore be regenerated as necessary.

generic ideas that will then be used within document classes. Most document authors will therefore likely directly encounter templates only rarely. We anticipate though that they will be *using* templates provided by the team or others.

The template system requires three separate ideas

- Template *type*: the "thing" we are using templates for, such as "sectioning" or "enumerated-list"
- A template: a combination of code and keys that can be used to implement a type. Here for example we might have "standard-LaTeX-sectioning" as a template for "sectioning"
- One or more *instances*: a specific use case of a template where (some) keys are set to known values. We might for example see "LaTeX-section", "LaTeX-subsection", etc.

As part of the move from the experimental xtemplate to kernel integration, the team have revisited the commands provided. The stable set now comprises

- `\NewTemplateType`
- `\DeclareTemplateInterface`
- `\DeclareTemplateCode`
- `\DeclareTemplateCopy`
- `\EditTemplateDefault`
- `\UseTemplate`
- `\DeclareInstance`
- `\DeclareInstanceCopy`
- `\EditInstance`
- `\UseInstance`
- `\IfInstanceExistsTF` and variants

To support existing package authors, we have released an updated version of xtemplate which will work smoothly with the new kernel-level code. The existing commands provided in xtemplate will continue to work, but we encourage programmers to move to the set above.

New or improved commands

doc: Provide `\ProvideDocElement`

In addition to `\NewDocElement` and `\RenewDocElement` we now also offer a `\ProvideDocElement` declaration that does nothing unless the doc element could be declared with `\NewDocElement`. This can be useful if documentation files are processed both individually and combined.

doc: Better support for upquote

In LaTeX News 37 [6] we wrote that support for the upquote package was added to the doc package, but back then this was added only for `\verb` and the `verbatim` environments. However, in a typical `.dtx` file, most of the code will be in the body of some `macrocode` or `macrocode*` environments, and neither of these was affected by adding upquote. We have now updated doc so that upquote alters the quote characters in these environments as well. *(github issue 1230)*

ifthen: Guard against active characters in comparisons

The `\ifthenelse` command now ensures that `<`, `=` and `>` are safe in numeric tests, even if they have been made active (typically by babel language shorthands). *(github issue 756)*

New conditionals: `\IfClassAtLeastT` and friends

Around 2020 we added a number of conditionals with CamelCase names, i.e., `\IfClassAtLeastTF`, `\IfClassLoadedTF`, `\IfClassLoadedWithOptionsTF`, `\IfFormatAtLeastTF`, `\IfPackageAtLeastTF`, `\IfPackageLoadedTF`, and `\IfPackageLoadedWithOptionsTF` to help arranging conditional code that depends on the release of a particular class, package or format. However, we only provided the `TF` commands and not also the `T` and `F` variants. This has now been changed.

In 2023 we introduced `\IfFileAtLeastTF` but we did not also provide `\IfFileLoadedTF` at the same time. This conditional and its `T` and `F` variants have now also been added. Remember that one can only test for files that contain a `\ProvidesFile` line. We did the same for the conditionals `\IfLabelExistsTF` and `\IfPropertyExistsTF`, also introduced in 2023.[3] *(github issues 1222 1262)*

Code improvements

Load packages only at the top level

Classes and packages must be loaded only by using the commands `\documentclass` and `\usepackage` or the class interface commands such as `\LoadClass` or `\RequirePackageWithOptions`; moreover, all of these must always be used at the top level, and not inside a group of any type (for example, within an environment). Previously LaTeX did not check this, which would often lead to low level errors later on if package declarations were reverted when a group ended. LaTeX now checks the group level and an error is thrown if the class or package is loaded in a group. *(github issue 1185)*

[3] By mistake they were initially introduced under the names `\IfLabelExistTF` and `\IfPropertyExistTF`; we corrected that at the same time. This is a breaking change, but the commands have been used so far only in kernel code.

Keep track of lost glyphs

A while ago we changed the LaTeX default value for `\tracinglostchars` from `1` to `2` so that missing glyphs generate at least a warning, but we forgot to make the same change to `\tracingnone`. Thus, when issuing that command LaTeX stopped generating warnings about missing glyphs. This has now been corrected. *(github issue 549)*

Improve fontenc error message

If the fontenc package is asked to load a font encoding for which it doesn't find a suitable `.def` file then it generates an error message indicating that the encoding name might be misspelled. That is, of course, one of the possible causes, but another one is that the installation is missing a necessary support package, e.g., that no support for Cyrillic fonts has been installed. The error message text has therefore been extended to explain the issue more generally. *(github issue 1102)*

Warn if counter names are problematic

In the past it was possible to declare, for example, `\newcounter{index}` with the side-effect that this defines `\theindex`, even though LaTeX has a `theindex` environment that then got clobbered by the declaration. This has now been changed: if `\the`⟨*counter*⟩ is already defined it is not altered, but instead a warning message is displayed. *(github issue 823)*

Extended information in `\listfiles`

The `\listfiles` command provides useful information when finding issues related to variation in package versions. However, this has to date relied on the information in the `\ProvidesPackage` line, or similar: that can be misleading if for example a file has been edited locally. We have now extended `\listfiles` to take an optional argument which can include the MD5 hash and size of each file in the `.log`. Thus for example you can use

```
\listfiles[hashes,sizes]
```

to get both the file sizes and file hashes in the `.log` as well as the standard release information. *(github issue 945)*

Optimize creation of simple document commands

Creating document commands using declarations such as `\NewDocumentCommand`, etc., provides a very flexible way of grabbing arguments. When the document command only takes simple mandatory arguments, this has to-date added an overhead that could be avoided. We have now refined the internal code path such that "simple" document commands avoid almost any overhead at point-of-use, making the results essentially as efficient as using `\newcommand` for low-level TeX constructs. Note that as `\NewDocumentCommand` makes engine-robust commands, the direct equivalent to `\newcommand` is `\NewExpandableDocumentCommand`. *(github issue 1189)*

Handling of end-of-lines in `+v` arguments of `\NewDocumentCommand` and friends

The `+v` argument type provided by declarations such as `\NewDocumentCommand`, etc., allows grabbing of multiple lines of text in a verbatim-like argument. Almost always, the result of this grabbing will be used in a typesetting context. Previously, the end-of-line characters were stored literally as category code 12 ("other") `^^M` tokens. However, these are difficult to work with in general. We have now revised this behavior, such that end-of-line characters are converted to the `\obeyedline` command when parsed by `+v`-type arguments. This change may require adjustments to the source of some documents, but the enhanced ability of users and programmers to exploit the `+v`-type argument means we believe it is necessary.

Declaring appropriate sub-encodings for TS1 symbol fonts

In 2020 we incorporated support for the `TS1` symbol encoding directly into the kernel and in this way removed the need to load the textcomp package [3] to make commands such as `\texteuro` available.

There is, however, a big problem with this `TS1` symbol encoding: only very few fonts provide every glyph that is supposed to be part of `TS1`. This means that changing font families might result in certain symbols becoming unavailable. This can be a major disaster if, for example, the symbol `\texteuro` (€) or `\textohm` (Ω) no longer gets printed in your document, just because you altered the text font family.

To mitigate this problem, in 2020 we also introduced the declaration `\DeclareEncodingSubset`. This declaration is supposed to be used in font definition files for the `TS1` encoding to specify which subset (we have defined 10 common ones) a specific font implements. If such a declaration is used then missing symbols are automatically taken from a fallback font.

While this is not perfect, it is the best you can do other than painstakingly checking that your document uses only glyphs that the font supports and, if necessary, switching to a different font or avoiding the missing symbols. See also the discussion in [4].

To jumpstart the process we also added declarations to the LaTeX kernel for most of the fonts found in TeX Live at the time—with the assumption that such declarations would over time be superseded by declarations in the `.fd` files. Unfortunately, this hasn't happened yet (or not often) and so many of the initial declarations went stale: several fonts got new glyphs added to them (so their sub-encoding should have been changed but didn't); others (mainly due to license issues) changed the family name and thus our declarations became useless and the renamed fonts (now without a declaration) ended up in the default sub-encoding that offers only a few glyphs; yet others such as CharisSIL (which triggered

the GitHub issue) were simply not around at the time.

We have, therefore, again attempted to provide the (currently) correct declarations, but it is obvious that this is not a workable process. As we do not maintain the fonts we do not have the information that something has changed, and to regularly check the ever growing font support bundles is simply not possible. It is therefore very important that maintainers of font packages not only provide `.fd` files but also add such a declaration to every `TS1...fd` font definition file that they distribute.

To simplify this process, we now provide a simple LaTeX file (`checkencodingsubset.tex`) for determining the correct (safe) sub-encoding. If run, it asks for a font family and then outputs its findings, for example, for `AlgolRevived-TLF` you will get:

```
----------------------------------------
Testing font family AlgolRevived-TLF
(currently TS1-sub-encoding 9)
----------------------------------------
Some glyphs are missing from sub-encoding 8:
   ==> \textcelsius (137) is missing
   ==> \texttwosuperior (178) is missing
   ==> \textthreesuperior (179) is missing
   ==> \textonesuperior (185) is missing
Some glyphs are missing from sub-encoding 7:
   ==> \texteuro (191) is missing
All glyphs between sub-encoding 6 and 7 exist
All glyphs between sub-encoding 5 and 6 exist
All glyphs between sub-encoding 4 and 5 exist
Some glyphs are missing from sub-encoding 3:
   ==> \textwon (142) is missing
All glyphs between sub-encoding 2 and 3 exist
Some glyphs are missing from sub-encoding 1:
   ==> \textmho (77) is missing
   ==> \textpertenthousand (152) is missing
All glyphs between sub-encoding 0 and 1 exist
All glyphs in core exist
----------------------------------------
TS1 encoding subset for AlgolRevived-TLF (ok)
Use sub-encoding 9
----------------------------------------
```

This output is meant for human consumption, e.g., you see which glyphs are missing and why a certain sub-encoding is suggested, but it is not that hard to use it in a script and extract the suggested sub-encoding by grepping for the line starting with `Use sub-encoding`.

Of course, this check will only work if the missing glyphs are really missing: some fonts placed "tofu"[4] into such slots and in this case it looks to TeX as if the glyph is provided. For example, for the old Palatino fonts (family `ppl`) it would report

```
----------------------------------------
TS1 encoding subset for ppl (bad)
Use sub-encoding 0 (not 5)
----------------------------------------
```

thus it claims that all glyphs are provided, while in reality more than twenty are missing and sub-encoding 5, as declared in the kernel, is in fact correct. *(github issue 1257)*

[4] Little squares to indicate a missing symbol.

Behavior when loading textcomp without options

When incorporating the textcomp package into the LaTeX kernel, in the February 2020 release [3], the default type of its package messages was changed from package info (`Package textcomp Info`) to LaTeX kernel info (`LaTeX Info`). But if textcomp was loaded without options, the message type got restored to package info. This restoration has now been canceled.

Note that loading textcomp with one of the options `error`, `warn`, or `info` still changes the message type to an error, warning, or info message from the textcomp package. *(github issue 1333)*

Rollback improvements

When requesting a rollback of the LaTeX kernel and/or packages, several packages produced the error "Suspicious rollback date" because their rollback section contained only data about recent releases even if the package, such as array, was available since the first release of LaTeX 2ε in 1994. We now suppress this error and load the first release that is still part of the distribution (and hope for the best). This change was implemented for the packages amsmath, array, doc, graphics, longtable, multicol, showkeys, textcomp, and varioref. *(github issue 1333)*

Documentation improvements

Further updates to the guides

We reported about the updated versions of `usrguide` and `clsguide` in LaTeX News 37 [6]. We have now revised `fntguide` as well to reflect the changes and macros added to the kernel over the last years of development. Note that the file name hasn't changed and there is no `fntguide-historic`.

Bug fixes

Fix inconsistent expansion of the package option list

LaTeX applies one-step expansion to the raw option list of packages and classes, so that constructions such as

```
\def\myoptions{opt1,opt2}
\usepackage[\myoptions]{foo}
```

are supported. But if a package declares its options using the new key/value approach [5] and it gets loaded a second time, then its raw option list will not be expanded and so an error might be raised. This has now been corrected. *(github issue 1298)*

Fix logic for first mark (page region)
In the new mark mechanism introduced in June 2022 [5] the result of `\FirstMark` on a two-column page was incorrect if the first column contained no marks. In that case it should have returned the first mark of the second column but didn't. This has now been corrected.

Documents using `\leftmark` are not affected, because that command is still using the old mechanism for now. *(github issue 1359)*

Struts at the end of footnotes or p columns
To produce consistent spacing in footnotes and tabular p-cells LaTeX adds a strut at the beginning and end of the content. This assumed, however, that the content of the footnote or tabular cell ended in horizontal mode and so, until now, these struts were unconditionally added; as a result, if this content ended with vertical material then this strut started a new paragraph consisting of a single line with just the strut in it. This has finally been corrected and now the placement logic for the strut changes when vertical mode is detected.

(First seen in a bug report for footmisc in combination with bigfoot)

Fix a "missing `\item`" rollback error
If LaTeX is rolled back to a date between 2023/06/01 (inclusive) and 2024/06/01 (exclusive), any list-based environment would raise an error:

```
! LaTeX Error: Something's wrong--perhaps
a missing \item.
```

This has now been corrected as a hotfix in patch level 2, by enhancing the 2023/06/01 version rollback code of the new paragraph mechanism. *(github issue 1386)*

Changes to packages in the amsmath category

amsmath: Correct equation tag placement
If there is not enough space to place an equation tag on the same line as the equation then amsmath calculates a suitable offset and it places the tag above (or below) the equation. In the case of the `gather` environment this offset was not reset at the end, with the result that it also got applied to any following environment, resulting in incorrect spacing in certain situations. This has now been corrected. *(github issue 1289)*

Changes to packages in the tools category

array, longtable, tabularx: Support tagging
These three packages have been extended so they can now, on request, produce tagged tabular. This is done by adding a number of sockets (see [7]) that, by default, do nothing; but when tagged PDF is requested they get equipped with appropriate plugs.

In the previous LaTeX release this was handled in `latex-lab`, by patching the packages when tagging was requested.

array: No `\unskip` in math cells
Math cells in the standard `array` environment of the kernel are not subject to space removal at the right end of the cell, i.e., explicit spaces from `\hspace` or `\␣`, etc. are honored (normal spaces are automatically ignored in math). In the array package all spaces got removed by calling `\unskip` unconditionally, regardless of the type of cell. This difference in behavior has now been removed by correcting the processing of math cells in array. *(github issue 1323)*

verbatim: `\verb` showed visible spaces
A recent change in the kernel was not reflected in the verbatim package, with the result that `\verb` showed visible spaces (␣) after the package was loaded. This has already been corrected in a hotfix for the November 2023 release. *(github issue 1160)*

verbatim: Support tabs in `\verbatiminput*`
Mimicking the November 2023 kernel update that allowed `\verb*` to mark tabs as spaces, the verbatim package has now been updated so that `\verbatiminput*` also marks tabs as spaces. *(github issue 1245)*

multicol: `\columnbreak` interferes with mark mechanism
The multicol package has to keep track of marks (from `\markright` or `\markboth`) as part of its output routine code and can't rely on LaTeX handling that automatically. It does so by artificially splitting page data with `\vsplit` to extract the mark data. With the introduction of `\columnbreak` that code failed sometimes, because it was not seeing any mark that followed such a forced column break.

This has now been corrected, but there is further work to do, because as of now multicol does not yet handle marks using the new mark mechanism—see the discussion at the beginning of the newsletter. *(github issue 1130)*

showkeys: Allow `\newline` in amsthm to work
Previously showkeys added an extra box layer which disabled the `\newline` of amsthm theorem styles. This extra box has now been avoided. *(github issue 1123)*

xr: Support links and properties
The xr package implements a system for eXternal References. The xr-hyper package (in the hyperref bundle) extended this to also support links to external documents. Using last year's extension of the `\label` command, which unified the label syntax of LaTeX and hyperref, it became possible to merge the two packages and thus make xr-hyper obsolete. With this change it is also possible to refer to properties that are stored in external documents using `\RecordProperties`. *(github issue 1180)*

Changes to files in the cyrillic category

Correct definition of \k

Ages ago, the encoding-specific definitions for various accent commands were changed to guard against altering some parameter values non-locally by mistake. For some reason the definition for `\k` in the Cyrillic encodings `T2A`, `T2B`, and `T2C` didn't get this treatment. This oversight has now been corrected. *(github issue 1148)*

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

[2] LaTeX Project Team. *LaTeX 2ε news 1–39*. June, 2024. https://latex-project.org/news/latex2e-news/ltnews.pdf

[3] LaTeX Project Team. *LaTeX 2ε news 31*. February, 2020. https://latex-project.org/news/latex2e-news/ltnews31.pdf

[4] LaTeX Project Team. *LaTeX 2ε news 33*. June 2021. https://latex-project.org/news/latex2e-news/ltnews33.pdf

[5] LaTeX Project Team. *LaTeX 2ε news 35*. June 2022. https://latex-project.org/news/latex2e-news/ltnews35.pdf

[6] LaTeX Project Team. *LaTeX 2ε news 37*. June 2023. https://latex-project.org/news/latex2e-news/ltnews37.pdf

[7] LaTeX Project Team. *LaTeX 2ε news 38*. November 2023. https://latex-project.org/news/latex2e-news/ltnews38.pdf

[8] Frank Mittelbach and Ulrike Fischer. *Enhancing LaTeX to automatically produce tagged and accessible PDF*. *TUGboat* 45:1, 2024. https://latex-project.org/publications/indexbyyear/2024/

LaTeX News

Issue 40, November 2024 (LaTeX release 2024-11-01)

Contents

Thirty years of LaTeX 2ε

In summer 1994, i.e., thirty years ago, LaTeX 2ε saw its first public release. Back then it was meant to be an intermediate version (hence the ε) on the way to a major new version (the mythical LaTeX3) that we expected to take a couple of more years to reach maturity. It took much more than that in the end—nominally, LaTeX 2ε is still with us today.

However, under the hood, LaTeX 2ε changed a lot throughout these thirty years, as one can see, for example, when looking through the forty newsletters [2] that accompanied the LaTeX releases that happened in the meantime.

During the first two decades, the LaTeX kernel was kept largely stable with only minimal bug fix activities. During that period additional functionality was mostly provided through new or extended packages that could be loaded in the document preamble. This included many of the ideas targeted for LaTeX3, e.g., expl3 (LaTeX3 programming language), xparse (new document command interface), xtemplate (a configuration mechanism), and many others.

Initially, this approach worked well and provided good backward compatibility; however, over time it became apparent that keeping all developments confined to packages was more and more problematical. Features or bug fixes that should have been generally available, i.e., part of the kernel, were only available in packages, so a lot of dependencies between packages were introduced and resulted in convoluted code that was difficult to manage. For example, hyperref had to rewrite a lot of kernel (and package) macros, so the code and behavior of other packages had to change depending on whether or not hyperref was loaded or not.

Thus, in 2015 the LaTeX team decided to change the policy and (re)start active kernel development, see [3]. To ensure continuous backward compatibility we introduced at the same time the latexrelease package that enables users to roll back changes to the LaTeX kernel to an earlier release, in case this is necessary to successfully rerun a document produced at that time.

As a consequence of this policy change the last decade saw a larger number of enhancements and corrections that were made part of the LaTeX kernel. Overall, we can confidently say that the new approach has worked well and enabled us to modernize LaTeX and ensure that it remains relevant without compromising one of

the cornerstones of LaTeX: its outstanding ability to reprocess old documents written many years ago.

Being able to update and modernize the kernel sources allowed us to embark in 2019 on the multi-year "LaTeX Tagged PDF" project with the goal of automatically providing accessible PDF documents with LaTeX. While there are several more project phases to complete, the milestones already reached allow users to generate PDF/UA compliant documents if the input is restricted to a (growing) subset of packages and document classes; see next section and previous newsletters.

A big change happened with the 2020-02-02 release as part of the project activity, albeit somewhat obfuscated by us as "Improved load-times for expl3". While technically correct, what it really meant is that we had finally integrated the programming layer of LaTeX3, i.e., the ideas originally sketched out around 1992. Or saying it differently: with that date the original ideas for LaTeX3 became a reality as part of the standard LaTeX kernel.

With the programming layer available under the hood we were then able to provide new concepts and extensions as part of LaTeX, e.g., the hook management system, a new mark mechanism, core functionality for tagging and PDF resource management, a consistent key/value interface, and more recently the socket and plug mechanism.

More will follow while we continue to work on modernizing LaTeX and bringing the Tagged PDF project to a truly successful completion—so stay tuned and watch this space for future announcements in the next newsletters.

News from the "LaTeX Tagged PDF" project

Engine support: An important update

As detailed below, work is progressing on the Tagged PDF project. There are many drivers for this work, including legal changes in many places which will increasingly require well-tagged PDFs including full support for mathematics. As part of the work on this, we are looking at the technical abilities of the TeX engines.

With XeTeX, it is impossible to reliably produce tagged PDFs due to engine limitations. The increasing importance of tagged PDFs means that this requires a move away from XeTeX. We will continue to address issues with XeTeX support in team-maintained LaTeX code on a best-effort basis. No *new* functionality will be added for XeTeX by the LaTeX team. It is likely that over time functionality may become more restricted, and users are urged to migrate XeTeX documents to LuaTeX.

For pdfTeX, tagging is available and we are able to support mathematics by including relevant TeX source or by using externally-generated MathML. Only LuaTeX is capable of *automatic* generation of MathML as part of a LaTeX run. Thus pdfTeX continues to be supported for existing material, but for new documents, moving to LuaTeX is recommended.

We cannot make statements about the support for other engines such as (u)pTeX, as we don't use these programs nor have in depth knowledge of their functionalities. To the best of our knowledge, core LaTeX works well with these engines, but if and to what extent tagging can be supported will remain to be seen. If relevant information becomes available to us we will provide an update in future editions of the LaTeX newsletter.

Tagging support for external packages

At https://latex3.github.io/tagging-project/tagging-status/ we show the status of many LaTeX packages and classes with respect to PDF tagging. We also started to improve tagging support in external packages. If the `firstaid` key is used in addition to the `phase-III` key, basic commands of several packages, including amsthm and fancyvrb, can now be used.

Improved table tagging

The tagging of tabulars has been extended: it is now possible to tag row headers and to create cells that span more than one row.

The interface to this functionality is not finalized but can be accessed in the current release by specifying the row and columns to be treated as headers. For example

```
\tagpdfsetup{
 table/header-rows={1,2},
 table/header-columns={1} }
```

would specify that in the following tables the first two rows and first column of each row should be tagged as heading entries.

Similarly you may add a RowSpan attribute to tag a cell that spans two rows using:

```
\tagpdfsetup{table/multirow={2}}
```

Automatic MathML tagging

When LuaLaTeX is being used, and the luamml package is available, and if the document uses the unicode-math package, then the math module will automatically convert each math formula to MathML and use it to attach MathML associated files (or MathML Structure elements) to the tagged PDF. This new feature can be disabled with `\tagpdfsetup{math/mathml/luamml/load=false}`. More options to configure MathML tagging can be found in the documentation of latex-lab-math.

Change behavior of tagging sockets with two arguments

When calling tagging sockets with two arguments using `\UseTaggingSocket` when tagging is suspended, previous versions of LaTeX 2ε dropped both arguments. This behavior has been changed to drop the first argument and preserve the second one instead, thereby allowing tagging sockets to be used to wrap existing content which should still appear in a non-tagging context.

Since no tagging sockets currently provided by LaTeX use two arguments we do not expect this change to affect any existing documents, but if a custom tagging socket has been defined outside of the kernel it might need to be adapted to be compatible with the new behavior.

(github issue 1500)

Changes to the LaTeX kernel

Handling paragraph continuation

Already LaTeX 2.09 offered some automation to detect whether or not text after a list or some other display environment is meant to be a continuation of the current paragraph or should start a new one. The document-level syntax for this is that a blank line after such an environment signals to LaTeX that it should start a new paragraph; whilst no blank line signals that there should be no new paragraph and the text should be considered a continuation.

Unfortunately, there were a number of cases where the original 2.09 approach failed, e.g., with

```
{⟨local customizations⟩
\begin{equation} a<b \end{equation}}
⟨some text⟩
```

the ⟨*some text*⟩ incorrectly started a new paragraph. Bug reports about this behavior can be traced back to the time LaTeX 2ε was developed, e.g., one test file from 1992 has a note that the above case was unfortunately not resolvable despite some improvements made back then. The main cause of the issue (as you probably guessed) is that the mechanism failed whenever the environment was executed within a group (`{...}`, `\begingroup`/`\endgroup`, or `\bgroup`/`\egroup` pair) that was closed before the next blank line was reached.

While most of the time this could be visually corrected by adding some explicit `\noindent`, the situation got worse when we tried to implement tagged PDFs resulting in incorrect structures or worse.

We therefore made a new attempt to resolve this problem in every situation and this new solution is rolled out in the current release.

Avoid bogus "no item" error

The commands `\addvspace` and `\addpenalty` generated the famous error message "Something's wrong—perhaps a missing `\item`" when they were encountered outside vertical mode. Most of the time this error was bogus and if not, then it was generated several times rather than once.

Once upon a time (in LaTeX 2.09) it was necessary that these commands were used only in vertical mode, but with LaTeX 2ε in 1994, we changed the internals but simply overlooked that this error message then had become useless. In this release, i.e., 30 years too late, we have finally lifted the ban and from now on this error should only show up if there is indeed a missing `\item`.

(github issue 1460)

Switch to T1 as default encoding in documents using `\DocumentMetadata`

As it is well known, the font encoding `OT1` supports only 128 characters and has various problems and quirks notably for languages different to English. Nevertheless `OT1` is the default encoding in LaTeX and this cannot be easily changed without affecting many documents as the `T1` version of the fonts have slightly different metrics.

The introduction of the `\DocumentMetadata` command, which announces *new* code and changes that can also affect the layout gives us now the opportunity to make this step. So with this version a use of `\DocumentMetadata` with (pdf)LaTeX will setup `T1` as default font encoding.[1] To ensure that scalable fonts are used, the package `cm-super` has to be installed. Users who want to revert to the `OT1` encoding in their document can do so with `\usepackage[OT1]{fontenc}`.

Code improvements

Avoiding key–value option clashes between classes and packages

In LaTeX News 35 [5] we introduced key–value option processing to the kernel. Following the standard for LaTeX 2ε options, keyval options given to the `\documentclass` line were treated as global and so parsed by every package. However, with keyvals, the likelihood of a name clash between a class-specific option and one used by a package is much higher than it is with simple strings. We have therefore refined the mechanism in the current release.

When a class uses the kernel keyval processor, any options it recognizes are recorded and any packages using the keyval processor will then *skip* these "global" options. To allow for the case where a class directly uses an option which should be global (for example `draft`), a new key property `.pass-to-packages` has been added. This can then be set to indicate that this key is not to be skipped. For example

```
\DeclareKeys{
  draft .if = {ifl@cls@draft},
  draft .pass-to-packages = true,
```

[1] The Unicode engines will continue to use `TU` as the encoding.

```
  mode   .store = \cls@mode
}
```

in a class would create two options, `draft` and `mode`. The `draft` option will be treated in the normal way by packages using keyvals, but they will ignore the `mode` option: it is effectively marked as "private" to the class. *(github issue 1279)*

Improvement to XeTeX \showhyphens

When using `\showhyphens` with XeTeX, missing character warnings would be generated for any character not in Latin Modern. This has been corrected and the warnings are suppressed. *(github issue 1380)*

Improved error raised for empty hook name

When using the hook management, both hook and label names (if specified) should be non-empty. Before, empty hook and empty label names both raised the same label-specific error:

```
! LaTeX hooks Error: Empty code label on line ..
  Using 'top-level' instead.
```

This has now been improved. Now an empty hook name generates

```
! LaTeX hooks Error: Empty hook name on line ..
```

(github issue 1423)

Provide counter representations for link targets

To create unique target names for links the package hyperref uses a special counter representation `\theH`⟨*counter*⟩. To ensure that this counter representation exists, hyperref redefined the commands `\@definecounter`, `\@addtoreset` and `\refstepcounter`. This counter representation is also needed for the Tagged PDF project and so these augmented command definitions have now been incorporated into the kernel. Thus from now on every `\newcounter{`⟨*counter*⟩`}` will define not only `\the`⟨*counter*⟩ but also `\theH`⟨*counter*⟩.

Extending \refstepcounter

For many years, the package hyperref had been redefining `\refstepcounter` and adding code that creates link targets. The kernel definition has now been extended with socket interfaces that will allow hyperref to avoid the redefinitions. The new interfaces are also used by the Tagged PDF code that needs target names to resolve references between structures.

Bug fixes

Fix wrong file type in a rollback warning

When LaTeX is rolled back to date ⟨*date1*⟩ and a class or package with minimum date requirement ⟨*date2*⟩ is to be loaded, a rollback warning is raised if ⟨*date2*⟩ is later than ⟨*date1*⟩:

```
LaTeX Warning: Suspicious rollback/min-date
               date given.
  A minimal date of YYYY-MM-DD has been
  specified for package '<pkgname>'.
  But this is in conflict with a rollback
  request to YYYY-MM-DD.
```

In some cases this message showed a wrong file type, i.e., `document class '<pkgname>'` or `package '<clsname>'`. This has now been corrected. *(github issue 870)*

Fix existence check of document environments

`\NewDocumentEnvironment` and friends define (or redefine) a document environment using the space-trimmed ⟨*envname*⟩, but the existence check for ⟨*envname*⟩ was done without space trimming. Thus when the user-specified ⟨*envname*⟩ consists of leading and/or trailing space(s), it may lead to erroneously silent environment declaration. For example, in

```
\NewDocumentEnvironment{myenv}{}{begin}{end}
\NewDocumentEnvironment{␣myenv␣}{}{begin}{end}
```

the first line defines a new environment `myenv` but the second line would check existence for `␣myenv␣` (which is not yet defined), then redefine `myenv` environment without raising any errors. This has now been corrected. *(github issue 1399)*

Handling of global keys with spaces

If the global (class) options contained spaces around key names, `\ProcessKeyOptions` would fail to remove known keys from the list of unused global options and `\OptionNotUsed` would mistakenly add space-surrounded key names to that list. The first issue was corrected as a hotfix in patch level 1 of the November 2023 release (but unfortunately not mentioned in [6]) and the second in the current release. *(github issue 1238)*

File list entries for rolled back packages/classes

When the rollback mechanism for packages and classes was introduced in 2018 [4], loading of the selected historic release was not recorded in the file list used by `\listfiles`. This has now been corrected so that the extended usage [7]

```
\listfiles[hashes,sizes]
```

now gives more complete and less confusing info. *(github issue 1413)*

doc: \PrintDescribeMacro in preamble

In doc version 2 it was possible to alter the definition of `\PrintDescribeMacro` and similar commands in preamble. In version 3 this stopped working because they were reset at the end of the preamble. This has now been implemented differently and changes in the preamble are possible again. *(github issue 1000)*

Avoid low-level error if \ShowHooks is used late
If \ShowHooks was used to examine a package hook after the package was loaded, a low level error resulted. This has now been corrected. *(github issue 1513)*

Avoid code duplication in rollback
When the kernel uses \AddToHook in a region that might be rolled back (which happens in a few places) and a document requests a rollback, then we have the situation that the hook already contains code to which we added the same (or slightly different) code during the rollback; this results in code duplication or, worse, in errors. This has now been corrected by dropping any such code chunk (if there is one) prior to adding the rollback code. *(github issue 1407)*

Passing template keys using \KeyValue
With the move of the template code to the kernel, internal functions were reviewed to improve efficiency. However, there was an oversight in how passing key values from one setting to another was implemented, such that using \KeyValue could result in an infinite loop. This has now been fixed. *(github issue 1486)*

Changes to packages in the amsmath category

Extend support for \dots
The implementation of \dots in amsmath has the feature that it selects different dots depending on the symbol that follows: e.g., dots between commas would normally be on the baseline, while dots between binary or relational symbols would be raised. However, when symbols such as \cong were protected from expansion in moving arguments (so that they worked in places such as headings) it had the unfortunate side-effect that the \dots magic stopped working for them. This has now been corrected. *(github issue 1265)*

Changes to packages in the tools category

Modification to generation of the .tex from fileerr
The fileerr extraction has been modified to write rename-to-empty-base.tex rather than .tex to comply with an expected security change in TeX Live 2025. The build.lua file for the tools has been modified to rename rename-to-empty-base.tex to .tex after unpacking. However if using docstrip directly rather than using l3build or the unpacked zip file from CTAN, the user must now rename the file and install as .tex. *(github issue 1412)*

array: Improve >{...} specifier
If the argument of >{...} ended with a command accepting a trailing optional argument, e.g., defined for example with \NewDocumentCommand\foo{o}{...}, one could get low-level parsing errors. This has now been corrected. *(github issue 1468)*

array: Tagging support for \cline
In the last release we added tagging support for array, longtable and other tabular packages, but we overlooked that the kernel definition for \cline also needs modification because the rule generated by the command needs to be tagged as an artifact. Furthermore, the processing of a \cline looks to the algorithm as if another row is added (which is technically what happens), thus it was also necessary to decrement the internal row counter to get a correct row count. This has now been corrected in array, which is automatically loaded for tagging, so that all these packages are now fully compatible with the tagging code if it is turned on. *(github tagging issue 134)*

longtable: Extend caption type
The longtable package has been extended and now provides the command \LTcaptype (stemming from the ltcaption package) to change the counter and caption type used by the \caption command from longtable. So with \renewcommand\LTcaptype{figure}, a longtable will step the figure counter instead of the table counter and produce an entry in the list of figures. An empty definition, \renewcommand\LTcaptype{}, will suppress increasing of the counter. This makes it easy to define an unnumbered variant of longtable:

```
\newenvironment{longtable*}
  {\renewcommand\LTcaptype{}\longtable}
  {\endlongtable}
```

longtable: Prevent \pagegoal exceeding maximum value
An internal guard has been added to avoid TeX errors if \pagegoal is increased beyond the maximum value for a TeX dimension. *(github issue 1495)*

Changes to l3build

To support third-party developers testing their code against pre-release LaTeX, a new switch --dev has been added to l3build. This allows the developer to run

```
l3build check
```

to run their test suite against the current release of LaTeX and

```
l3build check --dev
```

to run exactly the same tests using the development release of LaTeX.

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

[2] LaTeX Project Team. *LaTeX 2ε news 1–39*. June 2024. https://latex-project.org/news/latex2e-news/ltnews.pdf

[3] LaTeX Project Team. *LaTeX 2ε news 22*. January 2015. https://latex-project.org/news/latex2e-news/ltnews22.pdf

[4] LaTeX Project Team. *LaTeX 2ε news 28*. April 2018. https://latex-project.org/news/latex2e-news/ltnews28.pdf

[5] LaTeX Project Team. *LaTeX 2ε news 35*. June 2022. https://latex-project.org/news/latex2e-news/ltnews35.pdf

[6] LaTeX Project Team. *LaTeX 2ε news 38*. November 2023. https://latex-project.org/news/latex2e-news/ltnews38.pdf

[7] LaTeX Project Team. *LaTeX 2ε news 39*. June 2024. https://latex-project.org/news/latex2e-news/ltnews39.pdf

LaTeX News

Issue 41, June 2025 (LaTeX release 2025-06-01)

Contents

Introduction

We are continuing to work on the support of tagged PDF output and on wider kernel development, much of which is needed to make this happen. Probably the most notable changes in this latest release of the kernel are those in the output routine and in the mark mechanism: these are described in the first two sections.

Work also continues apace on further aspects of the tagging project, where some highlights are: new sockets, better graphics tagging, improved math mode support and the promotion of PDF 2.0.

In addition to this tagging-focussed work, we have also made advances in these areas: a new argument type for use in `\NewDocumentEnvironment` and friends which will make working with verbatim content a *lot* easier; further improvements to case changing; and, of course, several bug fixes.

Finally, we have highlighted a few of the recent changes in the L3 programming layer, where development work continues in parallel with other improvements to the LaTeX kernel; and we are integrating more, formerly "experimental", ideas into this core programming system.

A configurable output routine

For nearly 40 years LaTeX's output routine (the mechanism to paginate the document and attach footnotes, floats and headers & footers) was a largely hardwired algorithm with a limited number of configuration possibilities. Packages that attempted to alter any aspect of this process had to overwrite the internals, which led

to the usual problems: incompatibilities and out-of-date code whenever something was changed in LaTeX.

To improve this, and to support the production of accessible PDF documents, we have started to refactor the output routine by adding a number of hooks and sockets. This means that packages needing to adjust the output routine can do so safely, avoiding the dangers previously associated with such activities.

For packages that need to hook into the output routine we have implemented the following hooks:

`build/page/before`, `build/page/after` These two hooks enable packages to prepend or append code to the processing of each page in the output routine. They are implemented as mirrored hooks. Technically, they are executed at the start and the end, respectively, of the internal LaTeX 2_ε `\@outputpage` command. Currently, a number of packages change this command by adding code in exactly these two places—so they can now instead simply add code to these hooks.

`build/page/reset` Packages that set up special conventions for the main text (such as catcode changes, etc.) can use this hook to undo these changes within the output routine, so that they aren't applied to unrelated material such as the text for running headers or footers.

`build/column/before`, `build/column/after` These two hooks enable packages to prepend or append code to the column processing in the output routine. They are implemented as mirrored hooks. Technically, they are executed at the start and the end, respectively, of the internal LaTeX 2_ε `\@makecol` command. A number of packages alter `\@makecol` to place code in exactly these two places—they can now instead simply add their code to these hooks.

We have also added a number of sockets: for configuring the algorithm and also to support tagging. Two of these sockets are of interest for use in class files and also in the document preamble.

The first and most complex of these is the socket `build/column/outputbox`, which controls how the column text, the column floats (top and bottom) and the footnotes are combined in a column: i.e., their order and spacing.

Thus in order to change the layout, all one now has to do is to assign a suitable plug to this socket, like this:

`\AssignSocketPlug{build/column/outputbox}`
`{`⟨*plug-name*⟩`}`

For this socket we have provided the following plugs:

`space-footnotes-floats` After the galley text there is a vertical `\vfil` followed by the footnotes, followed by the bottom floats, if any.

`footnotes-space-floats` As before but the `\vfil` is between the footnotes and the floats.

`floats-space-footnotes` The floats come directly after the text, followed by a `\vfil` and then the footnotes at the bottom.

`space-floats-footnotes` Both floats and footnotes are pushed to the bottom, the footnotes coming last.[1]

`floats-footnotes` All excess space is distributed across the existing glue on the page: e.g., within the text galley, the separation between blocks, etc. The order is text, floats, footnotes.

`footnotes-floats` Like the previous plug, but floats and footnotes are swapped. This is the LaTeX default for newer documents, i.e., this plug is assigned to the socket when `\DocumentMetadata` is used.

`footnotes-floats-legacy` Like the previous plug, but LaTeX's bottom skip bug is not corrected: i.e., in ragged bottom designs where footnotes are supposed to be directly attached to the text, they suddenly appear at the bottom of the page when the page ends with a `\newpage` or `\clearpage`. While this is clearly a bug, it has been like this since the days of LaTeX 2.09; thus, for compatibility, we continue to support this behavior. This plug is assigned to the socket when `\DocumentMetadata` is *not* used.

By default the separation between the last line of text and the footnotes (`\skip\footins`) is not measured from the baseline of the last text line, but from its bottom. This goes back to plain TeX where it is done in this way. Similarly, `\textfloatsep` is added between text and bottom floats, not starting from the baseline of the last text line. Typographically speaking this is suboptimal, because it means that with `\flushbottom` in effect, the position of the last text line, when it is followed by footnotes or floats, depends on whether or not that line contains characters with descenders.

For this reason there is now also a socket `build/column/baselineattach` with a plug `on`: this causes the attachment of footnotes/floats to be measured from the baseline of the last text line. To mimic the behavior of old documents, this socket is, by default, assigned the plug `off`. For documents using `\DocumentMetadata`, the plug `on` will probably become the default here.

There are more configuration possibilities, mainly for class developers to use: documentation of these can be found in [4, §54 `ltoutput.dtx`].

Replacement for the legacy mark mechanism

LaTeX's legacy mechanism supported only two classes of marks, left and right marks, and setting the left mark

[1] There are two more permutations but neither of them has ever been requested; so these two are not set up by default—doing that in a class would be trivial though.

(with \markboth) always altered the state of the right mark as well, i.e., they were far from independent. For generating running headers with "chapter titles" on the left and "section titles" on the right, they work reasonably well but without much flexibility: e.g., \leftmark always generated the last "left"-mark on the page, while \rightmark always generated the first "right"-mark.

A few releases ago [2, p. 76] we therefore introduced a new mark mechanism for LaTeX, one that supports any number of truly independent mark classes. This mechanism also offers the ability to query the mark status at the top of the page, something that wasn't previously available at all.

Up to now these two mechanisms coexisted, with completely separate implementations; but we have now retired the legacy code and reimplemented its public interfaces using the new concepts. Thus the old commands (\markboth, \markright, \leftmark and \rightmark) remain supported, but internally these commands all use \InsertMark, etc.

Existing document classes, and documents using the legacy interfaces, will therefore continue to work without any modifications; but they now use a single underlying implementation. Also, new documents can benefit from the additional flexibility, e.g., by being able to display not only the first right-mark (\rightmark or \FirstMark{2e-right}) but also, or alternatively, the last right-mark (\LastMark{2e-right}) or the top right-mark (\TopMark{2e-right}), etc.

More information concerning all of this extended functionality can be found in [3].

News from the Tagged PDF project

In the Tagged PDF project we have now reached a state where, within certain limits, it is possible to generate accessible PDF output that conforms to PDF/UA for arbitrarily complex documents as long as they only use (a growing number of) compatible packages and classes.

The focus for this release was on adding special sockets for tagging support, on improving the tagging of math formulas, and on extending the tagging support for various types of graphics.

New Metadata keys, to activate tagging

Up to now users had to activate tagging by loading modules from latex-lab with the help of the testphase key. Further configuration of the tagging then had to be done by using the \tagpdfsetup command. We now offer Metadata keys for this that do not use "test" in their names, reflecting the fact that producing tagged PDF documents has become "production-ready".[2]

[2] To be fully precise, this is true provided only compatible packages and classes are used: these are listed at https://latex3.github.io/tagging-project/tagging-status/.

The tagging key allows for the activation and deactivation of the tagging support. It accepts the three values on, off and draft. When this key is used it loads the tagpdf package and all the modules that we currently recommend should be loaded.[3] The list of loaded modules will be adjusted as needed as the project progresses. For reference, it is also written to the log. Setting tagging=off loads the same set of modules and then deactivates the tagging commands in the class/before hook; and tagging=draft leaves the tagging commands active, so as to preserve warnings and errors in the tagging, but it deactivates the writing of the structure tree at the end of the compilation. This can save time when drafting a long document.

The tagging-setup key allows configuration of the tagging. It accepts as values all the keys that can be used in \tagpdfsetup, such as the math/setup key described below. It knows about both the key modules, which allows overwriting of the set of loaded modules, and the key extra-modules, which allows loading of experimental modules that are not yet in latest. The tagging-setup key implies tagging=on so that, if this key is used, then it is not necessary to also set the tagging key unless you want to turn tagging off, or to set it to draft.

With these new Metadata keys a standard setup might look like this:

```
\DocumentMetadata{
   pdfstandard={UA-2,A-4f},
   tagging=on,
   tagging-setup=
      {math/setup=mathml-SE,
       extra-modules=verbatim-alt}
 }
```

New value latest for testphase key

With the new keys for enabling tagging the use of the testphase key is now of minor importance and mainly of interest for developers and for backwards compatibility.

With this release it also supports the value latest. This will load all modules that we currently recommend should be loaded, so that it is not necessary to specify a long list of individual modules. The list of loaded modules will be adjusted as needed when the project progresses. For reference, it is also written to the log.

Sockets for tagging support

A lot of the tagging support in packages is handled through the socket-and-plug mechanism that was introduced in LaTeX 2023-11-01 [2, p. 93]. Sockets offer an easily used interface for package developers to invoke variable code at pre-specified places: code that then

[3] This set of modules can also be loaded with the key testphase=latest.

can be changed from outside the package by assigning a different plug to alter the processing.

For the tagging support, a specialized set of sockets is available: their plugs are invoked by using the `\UseTaggingSocket` command, instead of the normal `\UseSocket` command. This allows tagging to be turned off or on at high speed by the commands `\SuspendTagging` and `\ResumeTagging`, without the need to individually reassign plugs to each of the many tagging sockets [2, p. 97]. This is very useful when there is a need to typeset material several times during trials.

In the current release we now also offer three dedicated declaration commands for these "tagging sockets": this works better than directly using the underlying general socket interface. These new commands also better support the special conventions used for "tagging sockets". They are: `\NewTaggingSocket`, `\NewTaggingSocketPlug` and `\AssignTaggingSocketPlug`.

Setting the version to PDF 2.0

Creating a PDF 2.0 version is considered essential for any document that has substantial mathematical content. This is because only this PDF version supports the straightforward use of tags from the MathML namespace.

When `\DocumentMetadata` is used, LaTeX will therefore, by default, set PDF 2.0 as the PDF version. A different PDF version can, if required, be set by explicit use of the `pdfversion` key.

Setting up math tagging

With the LuaTeX engine there are now various options for the production of accessible math which are described in full detail in `latex-lab-math.pdf`. To simplify the setup, a new key `math/setup` can be used in `\tagpdfsetup` (or in `tagging-setup` as shown above) that accepts a comma list with the values `mathml-SE` (add MathML structure elements), `mathml-AF` (attach MathML associated files) or `tex-AF` (attach the TeX sources).

The use of $$...$$ for math displays

Use of the plain TeX method `$$...$$` in LaTeX, to mark up a display math formula, is not officially supported because it produces a fixed visual result that is not receptive to style changes such as the `fleqn` option. Instead, the recommended way is to use `\[ ... \]` or the `displaymath` environment. However, since many authors have used this input method in their documents, we are doing our best to support it for the production of accessible PDFs; but users should be aware that it has some limitations.

However, these accommodations for tagged PDF clash with the direct use of `$$` in environment definitions for special math environments (such as those defined in amsmath). The kernel therefore now contains the two commands `\dollardollar@begin` and `\dollardollar@end`. These new commands must be used by packages and classes to specify where inside an environment the displayed math formula starts and ends: their use is essential in order to make the package or a class compatible with tagging, and to allow its use when producing accessible documents. No more explicit `$$`s in code, please!

Package and class developers can prepare code to meet this new requirement by adding these two commands:

```
\providecommand\dollardollar@begin{$$}
\providecommand\dollardollar@end{$$}
```

and replacing every occurrence of `$$` with the appropriate start or end command.

Adding these `\providecommand` lines to classes and packages doesn't hurt but ensures that they will work with older LaTeX kernels.

Fixing the spacing after display math

When LaTeX produces accessible (tagged) PDF it has to add structure data in the PDF to mark (i.e., tag) individual elements. If the pdfTeX engine is used this has to be done with the help of `\pdfliteral`s, which are whatsit nodes like `\special` or `\write`. This means that they should be added only in places where these extra nodes do not affect the spacing—TeX can't, for example, look backwards past such a whatsit node, so consecutive spaces, that are normally collapsed into one, suddenly both appear when such a node separates them.

The situation is especially complicated in displayed math because there TeX adds penalties and spaces using low-level procedures that are not directly accessible from the macro level. Moreover, the PDF tagging structures have to be added somewhere in the middle of this processing: this is needed to ensure that the formula and these PDF structures do not get separated by a page break. Because of this it is necessary to use some fairly complex methods (essentially, we disable TeX's mechanisms and reprogram them on the macro level) to get the structure data in the right places.

Our first attempt to do this was slightly faulty and, in some cases, resulted in the addition of an incorrect `\parskip` space; this has now been corrected. The implementation that achieves this is a rather "interesting" study in obfuscated TeX coding—it is described in `latex-lab-math.pdf` for the interested.

When using LuaTeX the situation is much better because the necessary extra structures can be added at a later stage, after the formula has been typeset.

(github tagging issue 762)

Local changes to spacing around math displays

Due to TeX's low-level handling of display math, it is very difficult to add the code needed for tagging

within such display math formulas whilst ensuring that such code always stays on the same page as the formula. This is because such code must be placed after the end of the display, but before the TEX engine adds a `\postdisplaypenalty` to the page. However, there is no way to add code in the middle of this low-level TEX processing, which is why we have to resort to complex gymnastics as already hinted at: we set `\postdisplaypenalty` locally to 10000 and also make sure that `\belowdisplayskip` when used by TEX is negative. Then we let TEX do its job and afterwards regain control via `\aftergroup` and insert the tagging code. Finally, we add the real `\postdisplaypenalty` and make a space correction.

With our first implementation of this approach it was not possible for a user to add an explicit `\postdisplaypenalty` or `\belowdisplayskip` setting inside the formula. In this release we have slightly altered our algorithm to make such user adjustments possible again. *(github tagging issue 809)*

Extended support for pictures

The tagging of graphics has been reimplemented and now uses tagging sockets (see above). Document authors can choose between four tagging flavors on a per-graphic basis: as illustrative figures, as artifacts (i.e., decorations), as replacements for symbols, and if applicable as normal text (for example, "todo notes"). To this effect the options of `\includegraphics` and the environments `picture` and `tikzpicture` have been extended and now accept keys such as `alt` (for the description text of illustrative figures), `actualtext` (to set the symbol), and `artifact`. The code supports graphics produced using the tikz packages and "todo notes" from the todonotes package. The extended documentation in `latex-lab-graphics.pdf` lists the full set of options and also describes what authors of other graphic packages can do to make their packages tagging aware.

New or improved commands

Socket-and-plug conditionals

It is sometimes necessary, or helpful, to know whether a particular socket or plug exists (or whether a plug is assigned to a certain socket) and, based on such information, to take different actions. With the current release we added conditionals, such as `\IfSocketExistsTF`, to support such scenarios. Corresponding L3 programming layer conditionals are also provided. *(github issue 1577)*

Accessing the current counter

Counter commands such as `\alph`, `\stepcounter`, can now use the argument `*` to denote the *current counter* (in the sense used by `\label`). This is compatible with the use by the enumitem package of `\alph*` in item labels; and it is now generally available. Not all commands accept `*`; for example, `\counterwithin` and `\counterwithout` still require counter names as before. *(github issue 1632)*

Collecting environment bodies verbatim

The mechanisms provided with `\NewDocumentCommand`, etc., offer a powerful way to specify a range of types of document command and environment syntax. This includes the ability to collect the entire body of an environment, for cases where treating it as a standard argument is useful. It is also possible to use this mechanism to define arguments which grab their content verbatim. To date, however, it was not possible to combine these two ideas.

In this release a new specifier, `c`, has been introduced for use in `\NewDocumentEnvironment` and friends: this collects the body of an environment in a verbatim-like way. As with the existing `+v` specification, each separate line is marked by the special `\obeyedline` marker, which by default issues a normal paragraph. Thus, this new specifier is usable both for typesetting and for collecting file contents (the letter `c` indicates "collect code"). Thus, we may use[4]

```
\NewDocumentEnvironment
  {MyVerbatim}{!O{\ttfamily} c}
  {\begin{flushright}#1 #2\end{flushright}}
  {}
\begin{MyVerbatim}[\ttfamily\itshape]
  % Some code is shown here
  $y = mx + c$
\end{MyVerbatim}
```

to obtain

```
% Some code is shown here
$y = mx + c$
```

Code improvements

Refinement of `\MakeTitlecase`

We introduced `\MakeTitlecase` as a late addition to the June 2022 release, making use of the improved case code in the L3 programming layer. Compared to upper and lowercasing, making text titlecased is even trickier to get right: it can be applied either to the whole text, or on a word-by-word basis.

A subtle issue was reported concerning the L3 programming layer (https://github.com/latex3/latex3/issues/1316); this is related to how we deal with the case changing of "words" but it also shows up when you titlecase some text stored in a command.

[4] Proper support for Tagged PDF needs additional code which is not shown here.

We have looked again at how to implement `\MakeTitlecase` in order to make it as predictable as possible, and we have made a change in this release. The command no longer tries to lowercase text before applying titlecasing, and it therefore gives correct results for text stored in commands.

We have also added an additional key to the optional argument to `\MakeTitlecase` which allows the user to decide if the case change gets applied only to the first word (the default) or to all the words.

Tab character as a special character

In LaTeX News 38 [2, p. 95], we described a change to `\verb`, etc., that makes the tab character equivalent to a space; we have now completed this work by adding the tab character to the list of characters covered by `\dospecials`. This allows tab to be used, for example, in a `v` specification document command without the need for additional steps.

Refinement of `v` specification category codes

Work on verbatim argument handling has highlighted that it is problematic to store all characters as "other" (category code 12) when using a `v` specification in `\NewDocumentCommand`, etc. We have therefore now revised this so that characters of category code letter retain their original category code.

Logging declarations of commands and symbols

For thirty years the documentation claimed that `\DeclareTextSymbol`, `\DeclareTextCommand` and friends all log their changes. However, in contrast to their math counterparts, they never in fact did so. This behavior has now finally been corrected. *(github issue 1242)*

Improved management of the NFSS font series

LaTeX's font selection mechanism (NFSS) supports 9 weight levels, from ultra-light (`ul`) to ultra-bold (`ub`), and also 9 width levels, from ultra-condensed (`uc`) to ultra-expanded (`ux`). In the February 2020 release this mechanism was extended, so that requests to set the weight or the width attributes of a font series are combined in a sensible way [2, p. 52]: for example, if you typeset a paragraph in a condensed face using `\fontseries{c}\selectfont` and then you use `\textbf` inside the paragraph, a bold condensed face is selected. The combination of such values is done by consulting a simple lookup table whose entries are defined by using the command `\DeclareFontSeriesChangeRule`.

Until now, this lookup table was missing some entries, especially with regard to rarely used width values. In such cases, the series values were not combined as expected. This has been fixed (thanks to Maurice Hansen) by adding numerous `\DeclareFontSeriesChangeRule` entries so that, when combining these font series values, the full range of weights (from `ul` to `ub`) and widths (from `uc` to `ux`) is now supported. *(github issue 1583)*

Supporting the `ssc` and `sw` font shapes

The `ssc` font shape (spaced small capitals) is supported in LaTeX through the commands `\sscshape` and `\textssc`. However, until this release there were no font shape change rules defined for this, admittedly seldom available, shape; so

```
\sscshape\itshape
```

changed unconditionally to `it` (italics) rather than to `sscit` (spaced small italic capitals). Thanks to Michael Ummels, the missing declarations have now been added, so shape changes in font families that support spaced small capitals work properly. At the same time we took the opportunity to improve the fallbacks for the `sw` (swash) shapes, which are accessible through the commands `\swshape` or `\textsw`. If an `sw` combination is not available, the rules now try to replace `sw` with `it` rather than falling back to `n`. *(github issue 1581)*

Improving the handling of `\label`, `\index`, and `\glossary`

In standard LaTeX, the three commands `\label`, `\index`, and `\glossary` take exactly one mandatory argument, e.g., `\index{`⟨*entry*⟩`}`. In some extension packages, for example index or cleveref, these are all augmented to accept an optional argument and, in the case of `\index`, also a star form. These extensions conflicted with LaTeX's way of disabling these commands within the table of contents and within running headers because they were, in these places, redefined to expect just a mandatory argument and then do nothing. We have now changed this behavior, so that the redefinitions in these places now accept this extended syntax. *(github issue 311)*

Tracing lost characters

In LaTeX News 33 [2, p. 63] we announced that `\tracingall` changes `\tracinglostchars` to an error condition. This change has been reverted and `\tracingall` and `\tracingnone` no longer alter `\tracinglostchars`, so its current setting is retained.

The default value used in LaTeX is set so that lost character information is written as a warning to both the log and the terminal. Users may wish to change this into an error, in which case `\tracinglostchars` should be set to 5 (not 3) as this works in all engines. *(github issue 1687)*

Always use the extended pool of registers

As the kernel has grown, the use of registers has risen to the point where rolling back to the classical register allocation approach (using only 256 registers) is no longer viable. We have therefore adjusted the rollback code so that even when requesting a pre-2015 LaTeX, the extended pool remains in use.

A version of `\input` for expansion contexts

The LaTeX definition of `\input` cannot be used in places where TeX is performing expansion: the classic example is at the start of a tabular cell. There are a number of reasons for this: the key ones are that LaTeX's `\input` records which files are read, and provides pre- and post-file hooks. To support the need to carry out file input in expansion contexts, we have now added `\expandableinput`; this skips recording the file name and does not apply any file hooks, but otherwise behaves like `\input`. In particular, it still uses `\input@path` when doing file lookup (contrasting with the behavior of the TeX primitive, which remains internally available for programmers as `\@@input`). *(github issue 514)*

Bug fixes

Avoid problems with page breaks in the middle of `verbatim`-like environments

If a page break occurs in the middle of an environment that sets up special `\catcode` settings, such as a `verbatim` environment, then these settings will remain active when the output routine is building the page. This is normally harmless, because the material contained in the page had been previously tokenized, so that the `\catcode` changes do not matter. However, in certain circumstances tokenization can happen during this page processing: for example, if processing the header involves reading in a file; or if there is a command that uses `\scantokens` so that it retokenizes some material using the verbatim settings.

This has been fixed and LaTeX now explicitly resets the `\catcode` values to their default settings when entering the output routine. Furthermore, packages that make changes to the tokenization beyond what is done by `verbatim` can use the newly introduced hook `build/page/reset` to add their own resets to the output routine processing. This hook is evaluated after LaTeX has done its reset, so it is also possible, if necessary, to overwrite LaTeX's default behavior. *(github issue 600)*

Fix for nested use of `localmathalphabets`

In 2021 we introduced a method to overcome the problem that classic TeX engines (but not the Unicode engines) have only a very limited number of math alphabets available (so they easily got used up by loading math font packages, even if their symbols got used only occasionally). The idea was to avoid allocating all math alphabets globally, but instead to allow a number of them (defined by counter `localmathalphabets`) to vary from one formula to the next. This means that different formulas can make use of different alphabets, so the chances are much higher that the processing of a complex document succeeds. See [2, p. 69] for details.

Unfortunately, the approach we took back then failed in some cases of nested formulas, with the result that the wrong glyphs were used. This has now been corrected. *(github issues 1101 1028)*

docstrip: Error if an `.ins` file is problematic

If the file to be generated had the same name as a preamble declared with `\declarepreamble` then the preamble definition was overwritten, because the macro used to store it got reused to denote the output stream. The same problem happened with postambles declared with `\declarepostamble`. This situation is now detected and an error message is issued. To circumvent the issue, simply use a different macro name for the preamble or postamble. *(github issue 1150)*

Prevent a `cmd` hook from defining an undefined command

Using `\AddToHook{cmd/FOO/...}` when the command `\FOO` was undefined resulted in this command becoming `\relax`. Thus, if used, it no longer raised an "Undefined control sequence" error, but silently did nothing. This behavior has been corrected, and, if the command `\FOO` does not get defined later, e.g., in a package, it now raises an error when it is used in the document. *(github issue 1591)*

Process global options just once per package

In 2022, we introduced key–value (keyval) option processing in the kernel [2, p. 77]. This also added the idea that keys could have scope: load-only, preamble-only and general use. However, we overlooked that an option given globally (in the optional argument to `\documentclass`) would be repeatedly processed and could therefore lead to spurious warnings. This has now been corrected so that now each global option is seen, by the keyval-based option handling system, exactly once per package. *(github issue 1619)*

Make `\label`, `\index`, and `\glossary` truly invisible in running headers

LaTeX has had this bug since its initial implementation: whilst it correctly ignored any `\label`, `\index`, or `\glossary` command that appears in a mark, it neglected correct handling of the spaces around the command. As a result, one could end up with two spaces in the running header where only one should be present. This was detected as part of working on issue 311 and has now been corrected. *(github issue 1638)*

Fully expand the arguments of the declarations `\counterwithin` and `\counterwithout`

The arguments of the commands `\counterwithin` and `\counterwithout` are two counter names that are used to reset (or not reset) one counter when the other is stepped. They also redefine the representation of that counter, e.g., `\counterwithin{section}{chapter}` would lead to:

```
\renewcommand\thesection
   {\thechapter.\arabic{section}}
```

However, if one of these counters was not named explicitly, as in this example:

```
\newcommand\sectioncounter{section}
\counterwithin{\sectioncounter}{chapter}
```

then we ended up with

```
\renewcommand\thesection
   {\thechapter.\arabic{\sectioncounter}}
```

which could lead to strange results if `\sectioncounter` got changed later on. This has been corrected: these arguments now get fully expanded when the declaration is made. *(github issue 1675)*

Correction in the float placement algorithm

When floats are added to the current or next page, LaTeX makes several tests in order to find an area that can receive the float. One of these tests calculates how much space is already used on the page and how much additional space is needed to place the float in a particular area. This means that it looks not only at the height of the float but also at the values from `\intextsep` (for `h` floats) or `\textfloatsep` and `\floatsep` (for `t` and `b` floats). The resulting space requirement was then stored in an internal variable and compared to the space still available on the page. If the test failed, the algorithm tried the next area.

Unfortunately, the code was reusing the value in that internal variable as the starting point for the next test, without removing the added space for the float separation (`\intextsep`, `\floatsep`, or `\textfloatsep`). Thus the comparison was being made with the wrong value (i.e., too high); therefore the test may have incorrectly concluded that a float doesn't fit, even when it would in fact have fit. This has now been corrected. *(github issue 1645)*

Correct `\CheckEncodingSubset`

In [2, p. 83], and again in [2, p. 100], we suggested that font maintainers should place an appropriate `\DeclareEncodingSubset` declaration in each `ts1`⟨*family*⟩`.fd` file, so that this is tied to the font definition and so will be available whenever a font family is explicitly selected by `\fontfamily{`⟨*name*⟩`}` instead of using a font support package. Unfortunately, however, this method could result in incorrect selection of glyphs if the font encoding subset setting was evaluated before the `.fd` file was loaded (as subset 9 would then be assumed). This has been corrected: `\CheckEncodingSubset` now first loads the `.fd` file when this is necessary. *(github issue 1669)*

Ensuring late `\write` commands aren't lost

If a non-`\immediate` `\write` command is used after the final page has been shipped out then no write will happen because the system waits for a `\shipout` that will never happen. After the last page has been shipped out, we therefore force all further `\write` calls to be `\immediate`: this ensures that they get written even though we are not going to ship out any more pages. This change of behavior is implemented just before the `enddocument/afterlastpage` hook because this hook may contain such `\write` commands. *(github issue 1689)*

Documentation

Clarifying the handling of spaces by `\textcolor`

In contrast to other `\text`-commands such as `\textbf` or `\textrm`, the command `\textcolor` gobbles spaces at the start of its argument. Thus, for example, `Hello\textcolor{red}{␣World}` will produce the output HelloWorld. There are technical as well as compatibility reasons for this, so the behavior will not change. This is now correctly documented. *(github issue 1474)*

Changes to packages in the amsmath category

`\numberwithin` now aliased to `\counterwithin`

The amsmath package offers a `\numberwithin` declaration to specify that a counter should be reset whenever some other counter is stepped. This is a restricted version of the more general kernel command `\counterwithin` which was introduced in the LaTeX kernel in 2018 and extended in 2021 [2, p. 72]. With the current release we have made `\numberwithin` an alias for the more powerful `\counterwithin` and we suggest that the latter command is used in new documents. *(github issue 1673)*

amsmath: Correct equation tag placement

If there is not enough space to place an equation tag on the same line as the equation then amsmath calculates a suitable offset placement for the tag, above (or below) the equation. In the case of the `gather` environment this offset was not reset correctly, so that it also got applied to these tags in any following environment, which gave incorrect placement in certain situations. The fix for this, implemented in 2024/06, was not entirely correct; so this has been changed to do such resetting at the start of every displayed math environment. *(github issue 1289)*

Changes to packages in the graphics category

More accessibility keys in graphicx

The `\includegraphics` command now accepts `actualtext` and `artifact` keys, which by default do nothing but are used by the tagging code to provide an ActualText string or a boolean flag to indicate that the graphic is an artifact. *(github issue 1552)*

Changes to packages in the tools category

multicol: Full support for extended marks

In 2022 we introduced a new mark mechanism for LaTeX [2, p. 76]. However, the initial implementation covered only the standard output routine of LaTeX. As a result the extended marks were not available within columns produced with the multicol package (where they would be especially useful). This limitation has finally been lifted so that the new mechanism is now fully supported by all of our packages. *(github issue 1421)*

array: Improve preamble code for p, m and b

When the preamble of a `tabular` or `array` is being built, the arguments to `p`, `m`, or `b` columns all get expanded several times. This is normally harmless because that argument usually contains just an explicit dimension. However, in a case such as `p{\fpeval{15}pt}` these expansions resulted in an error; this happened because `\fpeval` was expanded a few times, but not often enough to result in a single number. This has now been corrected: these arguments are not expanded at all. This allows for such edge cases and also for the extensions available with the calc package, such as `p{\widthof{AAAAAA}}`. *(github issue 1585)*

array: Fix handling of empty p-cells

If an `\arraystretch` greater than `1` is used, table rows are spread apart by placing suitable struts (invisible rules) into each row, or in case of p-cells into each cell. If such a cell was empty the placement of the strut was not correct so that the cell appeared to be larger than it should have been. This has now been corrected. *(github issue 1730)*

varioref: How to make `\reftextfaceafter`, etc. empty

In the case that one wants to make a command such as `\reftextfaceafter` produce truly nothing, one has to get rid of the space that is automatically placed in front of the command by `\vref`. This can be done by simply defining the command to remove it, e.g.,

```
\renewcommand\reftextfaceafter{\unskip}
```

The varioref package does not test if such strings are empty, because that would require a lot of tests each time `\vref` is used, and it would nearly always find that the text is not empty. However, as shown above, the solution for this uncommon case is simple, and it is now explicitly documented in the package documentation. *(github issue 1622)*

Changes to files in the L3 programming layer

Work on the L3 programming layer continues in parallel with development of the rest of the LaTeX kernel. Of note for developers is that we have integrated more code into the main l3kernel bundle, and therefore into the functionality available automatically in LaTeX. Most notably, l3benchmark, which provides tools for checking code performance, is now part of l3kernel.

We have also extended the color module to recognize the Oklab and Oklch color models; thanks to Markus Kurtz for contributing this code. The Oklab color space (https://bottosson.github.io/posts/oklab) is a perceptual color space which is supported by CSS and so also by modern web browsers; Oklch expresses the Oklab color space in cylindrical form.

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

[2] LaTeX Project Team. *LaTeX 2ε news 1–41*. June 2025. https://latex-project.org/news/latex2e-news/ltnews.pdf

[3] Frank Mittelbach, LaTeX Project Team. *The `ltmarks.dtx` code*. June 2025. https://latex-project.org/help/documentation/ltmarks-doc.pdf

[4] LaTeX Project Team. *The LaTeX 2ε Sources*. June 2025. https://latex-project.org/help/documentation/source2e.pdf

LaTeX News

Issue 42, November 2025 (LaTeX release 2025-11-01)

Contents

Introduction

Fans of Douglas Adams will know that this must be a special issue of the LaTeX news; unfortunately it doesn't give us the "Answer to the Ultimate Question of Life, the Universe, and Everything" yet—we still need to address a few more issues to reach that point.

However, with this release we have made further progress in generating tagged and accessible PDF documents from LaTeX. To indicate this, we move away from calling it a prototype solution as by now it can be used in production workflows—and indeed already has been—as long as one restricts the documents to already-supported packages. This does not mean that latex-lab code (where the tagging support currently resides) is no longer under development, but that the user-facing side of the project is by now fairly stable and usable.

Outside of the tagging project we have added new functionality to the font selection support as well as code improvements and additional functionality in a number of other places, for example, supporting value expansion of key values at the time of declaration, which is useful when specifying template instances and in similar places.

As usual, there have been a few bugs to take care of (odd ones for sure this time) and we also decided to finally retire a few packages from the tools collection. More precisely, we suggest that they should no longer be used in new documents as there are better possibilities available by now.

News from the Tagged PDF project

Expanding the \DocumentMetadata command

In 2022 we introduced \DocumentMetadata with a twofold purpose: to provide a dedicated place for document-wide settings and metadata, and to act as a trigger command to identify documents that want to load new code. The latter allows the use of the new, extended interfaces essential for the tagging project, but also useful without tagging.

Initially, using \DocumentMetadata with an empty argument loaded only the PDF management code and a new hyperref driver was used. Since November 2024 \DocumentMetadata changes the default encoding from OT1 to T1; and since June 2025 it also changes the default PDF version from 1.7 to 2.0.

Additional code from latex-lab (needed, e.g., for the tagging project) had to be loaded explicitly by using

the `testphase` or the new `tagging` key in the argument of `\DocumentMetadata`. Whilst this allowed for the selective loading and testing of the new code, it also produced problems for classes and packages adapting their code for the tagging project since it was difficult to test which parts of the latex-lab code were active.

In this release we therefore extend `\DocumentMetadata` even further: it will now load directly all the code that one would get when using the `tagging=off` or the `testphase=latest` key.

The values `phase-I`, `phase-II`, `phase-III` of the `testphase` key will no longer load different code variants but only activate tagging. Extra modules not yet incorporated in the `latest` set of modules can still be loaded by using the `testphase` key.

For documents that want to load the PDF management but do not want the new tagging support code we provide a dedicated package. Such documents should replace

```
\DocumentMetadata{pdfversion=1.7,
  pdfstandard=a-3b}
```

by

```
\RequirePackage{pdfmanagement}
\SetKeys[document/metadata]
        {pdfversion=1.7,pdfstandard=a-3b}
```

Checking the compatibility with the tagging support code

We maintain a database showing the compatibility of classes and packages with the tagging support code. This data can be viewed online [7]. We have now exported a part of the data into a small package latex-tagging-status and added a key `check-tagging-status` to `\DocumentMetadata`. When used, the status of the packages and the class used by the document will be shown at the end of the log file.

This status is only a rough overview and a debugging aid, not a final report! Using packages that are classified as incompatible or partially incompatible does not mean that the tagging is necessarily broken. For example, hyperref is partially incompatible as form fields are not properly tagged (this requires the use of the l3pdffield package), but in documents without form fields it is unproblematic. In case of partially-compatible or incompatible packages the full table should be checked as it often contains an explanation of what is not yet working.

The package latex-tagging-status will be regularly updated to reflect changes in packages and the status database. Erroneous messages should be reported at the tagging project github repository [8]. It is also possible to create a pull request to update or correct the data.

Requiring or testing for the tagging support code

Classes or packages that are written only for the new code loaded by `\DocumentMetadata` can use the new command `\NeedsDocumentMetadata` at the start of the class or package file. It will produce a suitable error message if the tagging support code has not been loaded.

Classes and packages that want to support both legacy documents and newer documents using `\DocumentMetadata` can now use `\IfDocumentMetadataTF` to test whether the new code has been loaded—eventually in combination with a test of the date of the format. To test whether the PDF management has been loaded, the test `\IfPDFManagementActiveTF` is provided.

Moving paragraph tagging into sockets

Paragraphs in LaTeX can be nested, e.g., you can have a paragraph containing a display quote, which in turn consists of more than one (sub)paragraph, followed by some more text which all belongs to the same outer paragraph.

To model such "semantic paragraphs" LaTeX uses a structure named `text-unit`[1] and uses `text` (role mapped to `P`) only for (portions of) the paragraph text.

This is semantically clear and allows processors who care to identify the complete paragraphs by looking for `text-unit` tags. But we received a request for an option to disable the tagging of the "semantic paragraphs", so with this release we moved the relevant tagging code into sockets. The "semantic paragraphs" can now be disabled by assigning the `noop` plug to these sockets:

```
\AssignTaggingSocketPlug{para/semantic/begin}
  {noop}
\AssignTaggingSocketPlug{para/semantic/end}
  {noop}
```

Hooks for `\includegraphics` keys

The three key definitions `alt`, `actualtext` and `artifact` used by `\includegraphics` now contain hooks, named `Gin/alt`, `Gin/actualtext`, and `Gin/artifact`.[2] The first two are hooks with two arguments and get as first argument the purified (with `\text_purify:n`) value of the key which is also used in the PDF, and as second argument the raw value. The hooks are processed even if tagging is not activated. With them it is possible, for example, to store the alternative text:

```
\AddToHookWithArguments{Gin/alt}
  {\gdef\myalttext{#2}}
\includegraphics[alt=Hello World]
  {example-image}
The alt text of the graphic was \myalttext.
```

Symbolic structure names

The names of structure elements tags may be taken from the standard PDF namespaces like `Sect`, `H1` or `Figure` but they can also use alternative names, provided the

[1] The name is under review and is likely to change in the future.

[2] `Gin` refers to the family name used by keys in the graphicx package.

latter are role-mapped to a standard name. The second approach is useful for three reasons:

- It looks nicer, if, e.g., a bible uses tag names such as `Testament`, `Book` or `Chapter` instead of `Sect`.
- It is possible to formulate additional constraints on such structures in a schema and thus ensure that there is no `Testament` inside a `Book`, something that cannot be done if `Sect` is used everywhere.
- We can provide a uniform LaTeX set of names for tags.

Currently it is difficult for document authors to change tag names, as the tagging support code uses either a fixed name or some ad hoc internal variable. We therefore added three commands that offer an interface to declare, use and reassign *symbolic structure names*. `\NewStructureName` takes one argument and declares a *symbolic structure name*. The expandable command `\UseStructureName` takes one argument and allows using the name in a `\tagstructbegin` command. `\AssignStructureRole` allows assigning a role to the symbolic structure name.

In the coming months the various tag names in the tagging code will be replaced by such symbolic names. Once the process is finished, document and class authors will have a flexible tool to set up the tag names of their documents.

Normalizing key names for block environments

The display block environments, such as `itemize`, `center`, `verbatim`, etc., were all reimplemented a while back to become tagging-aware. That was also the first time we used the template/instance mechanism to offer consistent layout configuration possibilities (heading commands will be next to use that approach). Doing this meant experimenting with different setups to see what works best. However, as a side effect of these trials and rewrites we ended up with a rather inconsistent set of key names across the different templates, so after the dust had settled it was about time to take a look at the complete set and standardize the key names as much as possible. This task has been largely completed, though some changes are still likely while we develop more templates covering other areas.

These changes are basically transparent for users who are just interested in producing tagged and accessible documents out of the box. However, for people who have started to customize the layout of the environments, using for example `\EditInstance`, the key name changes need to be reflected accordingly.

Contexts in typesetting

Sometimes document elements should change their layout depending on where they are used; for example, lists might use less vertical space when used inside a footnote or a float. To allow for such designs in a consistent and easy way we introduce the concept of "named contexts": while typesetting the document LaTeX keeps track of a current "primary context" and a current "secondary context". They are changed automatically when certain commands, such as `\footnote`, or environments, such as floats, etc., are typeset.

For the primary context, LaTeX distinguishes by default between typesetting material in the main galley (context name is ⟨*empty*⟩), or in a `footnote`, `marginal`, `float`, `caption`, `header`, or `footer`.

The "secondary context" is by default used to identify typesetting in different font sizes and therefore knows about `tiny`, `scriptsize`, `footnotesize`, `small`, `large`, `Large`, `LARGE`, `huge`, `Huge`, and ⟨*empty*⟩ (denoting typesetting in `\normalsize`).

In theory it would be possible for commands and environments to query the current context and then alter their behavior; however, that would require comparatively complex coding. Instead, the main usage for the context is with template instances that are used to define layouts. If a template instance is used via `\UseInstance{`⟨*type*⟩`}{`⟨*inst-name*⟩`}` then this normally results in calling up an instance of type ⟨*type*⟩ with the name ⟨*inst-name*⟩.[3]

However, when the "primary context" and/or the "secondary context" is non-empty then `\UseInstance` searches for an instance that is especially tailored to the current context. This works as follows:

- The string `:`⟨*primary context*⟩`:`⟨*secondary context*⟩ is appended to ⟨*inst-name*⟩ and if that instance exists it is used.[4]
- If not, then ⟨*inst-name*⟩`:`⟨*primary context*⟩ is tried.
- If that doesn't exist either, then ⟨*inst-name*⟩ is used as usual.

This means it becomes trivial to alter the behavior of instances if they appear in a special typesetting context. For example, if `itemize-1` is the instance name for first-level itemize lists then one can define another instance named `itemize-1:footnote` to describe a special layout used in footnotes. More details on this can be found in `latex-lab-context.pdf` or by using `texdoc␣latex-lab-context` on the command line.

At this point in time the mechanism is still rather experimental; i.e., we provide and use it in latex-lab to gain experience and we also encourage developers to experiment with it and provide feedback. Details of the implementation are likely to change though.

[3] Such instances are defined with a `\DeclareInstance` or `\DeclareInstanceCopy` declaration; see the documentation in the file `lttemplates-doc.pdf`.

[4] Note that this means that if the ⟨*primary context*⟩ is empty we effectively append `::`⟨*secondary context*⟩.

MathML intent attributes

Two new commands, `\MathMLintent` and `\MathMLarg` are added. They are defined in the format as no-ops so they may be added to command definitions in packages. If luamml is enabled to generate MathML, these commands allow *intent* and *arg* attributes to be specified. A definition such as

```
\newcommand\abs[1]{%
   \MathMLintent
     {absolute-value($x)}%
     {{\lvert\MathMLarg{x}{#1}\rvert}}%
}
```

would cause `\abs{y}` to generate

```
<mrow intent="absolute-value($x)">
  <mo>|</mo><mi arg="x">y</mi><mo>|</mo>
</mrow>
```

which will allow Assistive Technology (AT) to correctly read the ambiguous notation $|y|$ as "the absolute value of y" or some similar reading depending on the chosen language.

Correctly handle tagging of math in tabular cells

Mathematical content in tabular cells was not correctly tagged when a MathML representation was automatically generated by LuaTeX. Also tabular preambles of the form `>{$}c<{$}` or `>{\(}c<{\)}` failed. This has been corrected. *(github tagging issues 973 983)*

New or improved commands

Support separate font families for script fonts

In TeX's math processing separate fonts can be selected for text, script and scriptscript sizes. LaTeX's NFSS traditionally uses the same font family at different sizes, handling adjustments needed for making fonts appear better in a script location through the use of optical sizes. This works great for traditional TeX fonts, but for OpenType fonts it leads to issues. OpenType math assumes the font in a script location has a separate feature set and therefore receives specific adjustments.

To support this without relying on heuristics based on the font size, a new command `\DeclareMathScriptfontMapping` has been added. It takes three pairs of encoding/family arguments to indicate that when the first pair is used as the math main font, the second and the third should be used as the script and scriptscript font, respectively. *(github issue 1707)*

Programming support for LaTeX's font metafamilies

LaTeX knows three main document font families: `\rmfamily` for the document's serifed font family, `\sffamily` for its sans serif font family, and `\ttfamily` for its monospaced font family. In addition, other font families can be used by the user or in a document class or package by explicitly loading them through `\fontfamily{`⟨*name*⟩`}\selectfont`.

In some cases it is helpful to know which of the three metafamilies (if any) is currently used for typesetting, and this information is now made available for programmers in `\@currentmetafamily`. It returns either `rm`, `sf`, `tt`, or `??` (in case none of the metafamilies is currently used).

As a small application of this, the LaTeX kernel now also contains `\@restoremetafamily`. If the current metafamily is ⟨*name*⟩ it executes `\`⟨*name*⟩`family`, e.g., `\sffamily`, and that then executes the hook ⟨*name*⟩`family` besides other re-initializations. This can be useful if that hook contains conditional code and the condition has changed, therefore requiring re-initialization.

Recovering the argument specifier for document commands

In LaTeX News 38 [4] we explained that we had *removed* `\GetDocumentCommandArgSpec` since we felt that it was only required for debugging. However, there are some special use cases where access to the argument specification is useful: see, for example, https://github.com/latex3/latex3/pull/1799. We have therefore looked again at this area and added a *code* interface `\cmd_arg_spec:N` for accessing the argument specification. The use of a code-level rather than design-level name here reflects the fact that this is a very specialized use case, mainly of interest to package authors.

Code improvements

Ensure that commands without arguments are not `\long`

In its original implementation `\newcommand` or `\renewcommand` always defined commands using `\long\def` even if the commands had no arguments, i.e., in situations where the concept of `\long` made no sense whatsoever.

The issue with that behavior is that commands differing only in their `\long` status are nevertheless considered different when compared with `\ifx`, even if there are no arguments to which the `\long` would apply. Thus, after `\renewcommand\rmdefault{lmr}` and `\def\test{lmr}` the test `\ifx\test\rmdefault` would be *false*, but it would be *true* if `\rmdefault` had been defined using `\def` (as many class files do). This made comparing commands without arguments rather difficult. We have therefore changed `\newcommand` and friends so that commands without arguments are always defined without using the unnecessary `\long` prefix.

Going forward, this will simplify package and kernel code as the code can reliably assume that such macros are not `\long` regardless of whether they are defined by `\renewcommand` or `\def`.

There is a small chance that this is a breaking change for some package code (though we don't know of any case). For instance, if the code was deliberately checking against `\long\def` only—in that case, the test now needs to be made against the definition without `\long` (or against both, which is what the NFSS implementation of the kernel did in the past). *(github issue 571)*

Avoid strange warnings about font substitutions

A font series value such as `sbc` contains both the weight (`sb`, i.e. "semibold") and the width (`c`, i.e. "condensed") of the font. If you want to reset only one of the two to "medium" and keep the other, you can use `\fontseries{m?}` or `\fontseries{?m}`: The former switches `sbc` to `c`, the latter switches `sbc` to `sb`. However, if the resulting series does not exist, you got strange warnings in the past, e.g.:

```
LaTeX Font Warning:
Font shape `OT1/cmss/c/n' undefined
using `OT1/cmss/m?/n' instead on input line 7.
LaTeX Font Warning:
Font shape `OT1/cmss/m?/n' undefined
using `OT1/cmss/m/n' instead on input line 7.
```

This has now been corrected so that you get a single, more meaningful warning:

```
LaTeX Font Warning:
Font shape `OT1/cmss/c/n' undefined
using `OT1/cmss/m/n' instead on input line 7.
```

If the `m` series does not exist either, you will still get strange warnings, but this should affect very few fonts. The source file was also tidied up a little on this occasion. *(github issue 1727)*

Improved handling of infinite shrinkage errors

In the June 2024 release [5] we described the improved mark mechanism and the problems we had when working around TEX's "infinite shrinkage error". By now, the engines have added a new primitive `\ignoreprimitiveerror` which can be used to turn this error into a warning, when, for example, you do only a trial splitting of a box. This noticeably improves the output in the `.log` file from

```
! Infinite glue shrinkage found in box being split.
<argument> Infinite shrink error above ignored !
l. ...   }
The box you are \vsplitting contains some
infinitely shrinkable glue, e.g., `\vss' or
`\vskip 0pt minus 1fil'. Such glue doesn't belong
there; but you can safely proceed, since the
offensive shrinkability has been made finite.
```

to a simple

```
ignored error: Infinite glue shrinkage found in
box being split
```

Perhaps even more important, the return code from the TEX run stays at 0 (unless there are real errors); so in workflows that want to test whether a TEX run ended without errors, you don't get a bogus result because there is no longer an ignored error. *(github issue 1750)*

Allow multiple family names in `\ProcessKeyOptions`

The ability to process key–value options was introduced into the kernel in the June 2022 release [3], with the command `\ProcessKeyOptions` carrying out the option assignment. In the original version, this takes an optional argument which can select one key family (namespace) for options. We have now extended this to take a comma-separated list of possible families. *(github issue 1756)*

Control of value expansion in keys

Normally, key–value input is treated "as is", with no expansion of either key names or values. However, there are occasions when expansion of selected values is useful. We have now extended the key handling for templates (`\DeclareInstance`, etc.) and for keys created using the L3 programming layer to allow selective expansion. In both cases, the syntax uses a trailing colon and a single letter specifier: these letters are those used in `\ExpandArgs` or the L3 programming layer. For example, to use the values of the LATEX 2_ε variable `\@itemdepth`, one could have settings

```
key-a:c =  @itemdepth ,
key-b:v =  @itemdepth
```

This facility will *automatically* be available in any package setup macro using the L3 programming layer, for example siunitx. *(github issue 1801)*

Support word exclusion in case changing

Work on improving automatic case changing over previous releases has continued. We have now added the ability to 'register' words for exclusion from case changing, using `\DeclareLowercaseExclusions`, `\DeclareTitlecaseExclusions` and `\DeclareUppercaseExclusions`.

Automatic insertion of `\par` tokens

Since 2022 the major TEX engines have provided a parameter, `\partokencontext`, that controls whether a `\par` token is added when TEX is in horizontal mode at the end of `\vbox` and in similar contexts. This gives more control than the classical behavior where the internal *end paragraph* routine is invoked with no explicit token being added.

This allows the paragraph hooks to detect the end of paragraph even in contexts such as at the end of a `\vbox`, where traditionally package code has had to be modified to add an explicit `\par`. This is expected to improve compatibility of existing packages with the tagging code.

LaTeX now sets this parameter to 2 by default, to enable automatic insertion of `\par` in these contexts. *(github issue 1864)*

Improved access to generic hooks

The code to add generic hooks such as `\AddToHook{cmd/somecmd/before}{...}` has been improved so that it is more likely to succeed in cases where the command has been defined using *expl3* syntax. Previously, attempts to add hooks to commands would fail if the original definition used `~` in an `\ExplSyntaxOn` context. *(github issue 1099)*

Bug fixes

Support active characters correctly with `\DeclareRobustCommand`

The mechanism used by `\DeclareRobustCommand` creates an internal command which has a space added to the name of the document one: so `\foo␣` for a command `\foo`. That fails if applied to an active character: unlike normal commands, these have to be exactly one character long. Due to the way the implementation works, to date this would result in redefining `\␣` every time `\DeclareRobustCommand` was used with an active character. This has now been corrected: robust active characters are now created using the engines' `\protected` mechanism and do not use an internal auxiliary. They still work in file names and labels to give the character itself. *(github issue 345)*

Avoid a "Corrupted NFSS tables" error

When a character with an accent is typeset, say "ä" or "é", it might be the case that it doesn't exist in the font but has to be constructed from the base character and a standalone accent. If that accent is also not available in the font then LaTeX attempts to find it in a different font, typically one in a different encoding, e.g., `OT1`. Unfortunately, when that involved font substitutions it resulted in a loop generating the mentioned error. This has now been corrected by adding necessary `\DeclareFontSubstitution` statements. *(github issue 1709)*

Changes to packages in the tools category

Updating the status of some components

The tools bundle contains a range of packages with different usage profiles. Some of these were necessary in the transition from LaTeX 2.09 to LaTeX 2_ε, while others continue to be very widely used in current documents (for example, array). We have therefore marked a small number of packages in tools as *retained only for historical and stability reasons*, and, where relevant, pointed to more up-to-date alternatives; the list is:

- enumerate: use enumitem instead
- rawfonts: retained as part of LaTeX 2.09 support
- somedefs: retained as part of LaTeX 2.09 support
- theorem: use amsthm instead
- verbatim: use fancyvrb instead

Update to handling page marks in longtable

The longtable package has been updated to correctly adjust the new LaTeX mark structures as each page is output. *(github issue 1814)*

Update to bm

The bm package has been extended to accept commands defined via `\chardef`—for example `\#`. *(github issue 1867)*

Changes to files in the firstaid category

First aid for AMS classes

The AMS classes still use the old mark mechanism (replaced by a new one in 2024 and finally retired in the June 2025 release) overwriting some but not all of its code. In some cases this now causes problems, so we have added a first aid for now. *(github issue 1887)*

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

[2] LaTeX Project Team. *LaTeX 2_ε News 1–42*. November 2025. https://latex-project.org/news/latex2e-news/ltnews.pdf

[3] LaTeX Project Team. *LaTeX 2_ε News 35*. June 2022. https://latex-project.org/news/latex2e-news/ltnews35.pdf

[4] LaTeX Project Team. *LaTeX 2_ε News 38*. November 2023. https://latex-project.org/news/latex2e-news/ltnews38.pdf

[5] LaTeX Project Team. *LaTeX 2_ε News 39*. June 2024. https://latex-project.org/news/latex2e-news/ltnews39.pdf

[6] LaTeX Project Team. *LaTeX 2_ε News 41*. June 2025. https://latex-project.org/news/latex2e-news/ltnews41.pdf

[7] LaTeX Project Team. *Tagging Status of LaTeX Packages and Classes*. November 2025. https://latex3.github.io/tagging-project/tagging-status

[8] LaTeX Project Team. *The LaTeX Tagged PDF repository*. November 2025. https://github.com/latex3/tagging-project/issues