LaTeX News

Issue 38, November 2023 (LaTeX release 2023-11-01)

Contents

News from the "LaTeX Tagged PDF" project 1
Approaching an important milestone 1
A GitHub repository dedicated to the project . 1

Hooks, sockets and plugs 1

Document properties and cross-referencing 2

New or improved commands 3
Testing for the LaTeX3 programming layer version: `\IfExplAtLeastTF` 3

Code improvements 3
Support for tabs in `\verb*` and `verbatim*` . . 3
Improved argument checking for box commands 3
Aligning status of tilde with other active characters 3
In the programming layer 4

***Removed* kernel commands** 4

Changes to packages in the tools category 4
longtable: correct p-column definition 4

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/`