The latexrelease package*

The LaTeX3 Project

2015/06/22

1 Introduction

Prior to the 2015 release of LaTeX, essentially no changes had been made to the LaTeX format code for some years, with all improvements being instead added to the package fixltx2e.

While this worked at a technical level it meant that you had to explicitly opt-in to bug fixes and improvements, and the vast majority of documents did not benefit.

As described in LaTeX News 22, a new policy is being implemented in which improvements will now be added to the format by default, and this latexrelease package may be used to ensure stability where needed, either by making a new format use an older definition of some commands, or conversely may be used to supply the new definitions for use with an old format.

The basic use is:

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

After such a declaration the document will use definitions current in the January 2015 LaTeX, whether the actual format being used is older, or newer than that date. In the former case a copy of `latexrelease.sty` would need to be made available for use with the older format. This may be used, for example, to share a document between co-workers using different LaTeX releases, or to protect a document from being affected by system updates. As well as the definitions within the format itself, individual packages may use the commands defined here to adjust their definitions to the specified date as described below.

The bulk of this package, after some initial setup and option handling consists of a series of `\IncludeInRelease` commands which have been extracted from the main source files of the LaTeX format. These contain the old and new versions of any commands with modified definitions.

*This file has version number v1.0f, last revised 2015/06/22.

2 Package Options

- *yyyy/mm/dd* The package accepts any LaTeX format date as argument, although dates in the future for which the current release of this package has no information will generate a warning.
- `current` This is the default behaviour, it does not change the effective date of the format but does ensure that the `\IncludeInRelease` command is defined.
- `latest` sets the effective date of the format to the release date of this file, so in an older format applies all patches currently available.

3 Release Specific Code

The `\IncludeInRelease` mechanism allows the kernel developer to associate code with a specific date to choose different versions of definitions depending on the date specified as an option to the latexrelease package. Is also available for use by package authors (or even in a document if necessary).

`\IncludeInRelease` `{`⟨*code-date*⟩`}[`⟨*format-date*⟩`]{`⟨*label*⟩`}{`⟨*message*⟩`}`⟨*code*⟩`\EndIncludeInRelease`

`{`⟨***code-date***⟩`}` This date is associated with the `{`⟨*code*⟩`}` argument and will be compared to the requested date in the option to the latexrelease.

`[`⟨***format-date***⟩`]` This optional argument can be used to specify a format date with the code in addition to the mandatory `{`⟨*code-date*⟩`}` argument. This can be useful for package developers as described below.

`{`⟨***label***⟩`}` The `{`⟨*label*⟩`}` argument is an identifier (string) that within a given package must be a unique label for each related set of optional definitions. Per package at most one code block from all the `\IncludeInRelease` declarations with the same label will be executed.

`{`⟨***message***⟩`}` The `{`⟨*message*⟩`}` is an informative string that is used in messages. It has no other function.

⟨***code***⟩ Any TeX code after the `\IncludeInRelease` arguments up until the and the following `\EndIncludeInRelease` is to be conditionally included depending on the date of the format as described below.

The `\IncludeInRelease` declarations with a given label should be in reverse chronological order in the file. The one chosen will depend on this order, the effective format version and the date options, as described below.

If your package mypackage defines a `\widget` command but has one definition using the features available in the 2015 LaTeX release, and a different definition is required for older formats then you can use:

```
\IncludeInRelease{2015/01/01}{\widget}{Widget Definition}
 \def\widget{new version}%
\EndIncludeInRelease

\IncludeInRelease{0000/00/00}{\widget}{Widget Definition}
 \def\widget{old version}%
\EndIncludeInRelease
```

If a document using this package is used with a format with effective release date of 2015/01/01 or later the new code will be used, otherwise the old code will be used. Note the *effective release date* might be the original LaTeX release date as shown at the start of every LaTeX job, or it may be set by the latexrelease package, so for example a document author who wants to ensure the new version is used could use

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

If the document is used with a LaTeX format from 2014 or before, then latexrelease will not have been part of the original distribution, but it may be obtained from a later LaTeX release or from CTAN and distributed with the document, it will make an older LaTeX release act essentially like the 2015 release.

3.1 Intermediate Package Releases

The above example works well for testing against the latex format but is not always ideal for controlling code by the release date of the *package*. Suppose LaTeX is not updated but in March you update the mypackage package and modify the definition of `\widget`. You could code the package as:

```
\IncludeInRelease{2015/03/01}{\widget}{Widget Definition}
 \def\widget{even newer improved March version}%
\EndIncludeInRelease

\IncludeInRelease{2015/01/01}{\widget}{Widget Definition}
 \def\widget{new version}%
\EndIncludeInRelease

\IncludeInRelease{0000/00/00}{\widget}{Widget Definition}
 \def\widget{old version}%
\EndIncludeInRelease
```

This would work and allow a document author to choose a date such as

```
\RequirePackage[2015/03/01]{latexrelease}
\documentclass{article}
\usepackage{mypackage}
```

To use the latest version, however it would have disadvantage that until the next release of LaTeX, by default, if the document does not use latexrelease to specify a date, the new improved code will not be selected as the effective date will be 2015/01/01 and so the first code block will be skipped.

For this reason `\IncludeInRelease` has an optional argument that specifies an alternative date to use if a date option has not been specified to latexrelease.

```
\IncludeInRelease{2015/03/01}[2015/01/01]{\widget}{Widget Definition}
 \def\widget{even newer improved March version}%
\EndIncludeInRelease

\IncludeInRelease{2015/01/01}{\widget}{Widget Definition}
 \def\widget{new version}%
```

```
\EndIncludeInRelease

\IncludeInRelease{0000/00/00}{\widget}{Widget Definition}
 \def\widget{old version}%
\EndIncludeInRelease
```

Now, by default on a 2015/01/01 LaTeX format, the first code block will compare the format date to the optional argument 2015/01/01 and so will execute the *even newer improved* version. The remaining blocks using the `\widget` label argument will all then be skipped.

If on the other hand the document requests an explicit release date using latexrelease then this date will be used to decide what code block to include.

3.2 Using `\IncludeInRelease` in Packages

If `\IncludeInRelease` is used within a package then all such conditional code needs to be within such declarations, e.g., it is not possible in the above example to have the "current" definition of `\widget` somewhere in the main code and only the two older definitions inside `\IncludeInRelease` declarations. If you would do this then one of those `\IncludeInRelease` declarations would be included overwriting the even newer code in the main part of the package. As a result your package may get fragmented over time with various `\IncludeInRelease` declarations sprinkled throughout your code or you have to interrupt the reading flow by putting those declarations together but not necessarily in the place where they belong.

To avoid this issue you can use the following coding strategy: place the current `\widget` definition in the main code where it correctly belongs.

```
  ...
\def\widget {even newer improved March version}
\def\@widget{newly added helper command no defined in older releases}
  ...
```

Then, near the end of your package place the following:

```
\IncludeInRelease{2015/03/01}[2015/01/01]{\widget}{Widget Definition}
\EndIncludeInRelease

\IncludeInRelease{2015/01/01}{\widget}{Widget Definition}
 \def\widget{new version}%
 \let\@widget\@undefined % this doesn't exist in earlier releases
\EndIncludeInRelease

\IncludeInRelease{0000/00/00}{\widget}{Widget Definition}
 \def\widget{old version}%
\EndIncludeInRelease
```

This way the empty code block hides the other `\IncludeInRelease` declarations unless there is an explicit request with a date 2015/01/01 or earlier.

Now if you make a further change to `\widget` in the future you simply copy the current definition into the empty block and add a new empty declaration with todays date and the current format date. This way your main code stays readable and the old versions accumulate at the end of the package.[1]

[1] Of course there may be some cases in which the old code has to be in a specific place within

The only other "extra effort" necessary when using this approach is that it may be advisable to undo new definitions in the code block for the previous release, e.g., in the above example we undefined `\@widget` as that isn't available in the 2015/01/01 release but was defined in the main code. If all your conditional code is within `\IncludeInRelease` declarations that wouldn't been necessary as the new code only gets defined if that release is chosen.

4 fixltx2e

As noted above, prior to the 2015 LaTeX release updates to the LaTeX kernel were not made in the format source files but were made available in the fixltx2e package. That package is no longer needed but we generate a small package from this source that just makes a warning message but otherwise does nothing.

5 Implementation

We require at least a somewhat sane version of LaTeX 2_ε. Earlier ones where really quite different from one another.

```
1 ⟨*latexrelease⟩
2 \NeedsTeXFormat{LaTeX2e}[1996/06/01]
```

6 Setup

`\IncludeInRelease`
`\EndIncludeInRelease`

```
3 \DeclareOption*{%
4   \def\@IncludeInRelease#1[#2]{\@IncludeInRele@se{#1}}%
5   \let\requestedpatchdate\CurrentOption}
6 \DeclareOption{latest}{%
7   \let\requestedpatchdate\latexreleaseversion}
8 \DeclareOption{current}{%
9   \let\requestedpatchdate\fmtversion}
10 \ExecuteOptions{current}
11 \ProcessOptions\relax
```

Sanity check options, it allows some non-legal dates but always ensures `requestedLaTeXdate` gets set to a number. Generate an error if there are any non digit tokens remaining after removing the `//`.

```
12 \def\reserved@a{%
13 \edef\requestedLaTeXdate{\the\count@}%
14 \reserved@b}
15 \def\reserved@b#1\\{%
16 \def\reserved@b{#1}%
17 \ifx\reserved@b\@empty\else
18 \PackageError{latexrelease}%
19              {Unexpected option \requestedpatchdate}%
```

the package as other code depends on it (e.g., if you `\let` something to it). In that case you have to place the code variations in the right place in your package rather than accumulating them at the very end.

```
            {The option must be of the form yyyy/mm/dd}%
\fi}
\afterassignment\reserved@a
\count@\expandafter
  \@parse@version\expandafter0\requestedpatchdate//00\@nil\\
```

less precautions needed for `\fmtversion`

```
\edef\currentLaTeXdate{%
   \expandafter\@parse@version\fmtversion//00\@nil}
\ifnum\requestedLaTeXdate=\currentLaTeXdate
\PackageWarningNoLine{latexrelease}{%
  Current format date selected, no patches applied.}
\expandafter\endinput
\fi
```

A newer version of latexrelease should have been distributed with the later format.

```
\ifnum\currentLaTeXdate
  >\expandafter\@parse@version\latexreleaseversion//00\@nil
\PackageWarningNoLine{latexrelease}{%
The current package is for an older LaTeX format:\MessageBreak
LaTeX \fmtversion\space\MessageBreak
Obtain a newer version of this package!}
\expandafter\endinput
\fi
```

can't patch into the future, could make this an error but it has some uses to control package updates so allow for now.

```
\ifnum\requestedLaTeXdate
  >\expandafter\@parse@version\latexreleaseversion//00\@nil
\PackageWarningNoLine{latexrelease}{%
The current package is for LaTeX \latexreleaseversion:\MessageBreak
It has no patches beyond that date\MessageBreak
There may be an updated version\MessageBreak
of this package available from CTAN}
\expandafter\endinput
\fi
```

Update the format version to the requested date.

```
\let\fmtversion\requestedpatchdate
\let\currentLaTeXdate\requestedLaTeXdate
```

7 Individual Changes

The code for each change will be inserted at this point, extracted from the kernel source files.

⟨/latexrelease⟩

8 fixltx2e

Generate a stub fixltx2e package:

⟨*fixltx2e⟩

```
\NeedsTeXFormat{LaTeX2e}
\PackageWarningNoLine{fixltx2e}{%
fixltx2e is not required with releases after 2015\MessageBreak
All fixes are now in the LaTeX kernel.\MessageBreak
See the latexrelease package for details}
```
⟨/fixltx2e⟩