The latexrelease package*

The LaTeX Project

2022/02/28

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.

Note that the latexrelease package is intended for use at the start of a *document*. Package and class code should not include this package as loading a package should not normally globally reset the effective version of LaTeX that is in force, so affecting all other packages used in the document.

*This file has version number v1.0o, last revised 2022/02/28.

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.

2 Package Options

- *yyyy/mm/dd* or *yyyy-nn-dd* The package accepts any possible 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. Dates earlier than 2015 will work but will roll back to some point in 2015 when the method was introduced. The `\requestedLaTeXdate` is set to the normalized date argument so that package rollback defaults to the specified date.

- `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. The `\requestedLaTeXdate` macro is reset to 0 so that package rollback does not use the implicit date.

- `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. The `\requestedLaTeXdate` macro is reset to 0 so that package rollback does not use the implicit date.

In all cases, when the package is loaded, the `\sourceLaTeXdate` is defined to be the numerical representation of `\fmtversion` before the rollback/forward actually happens, so it is possible to test from which was the original LaTeX version before latexrelease was loaded. This is particularly useful when some code in a package has to be redefined differently if rolling backwards in time or forwards.

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 today's 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]

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 Declaring entire modules

Sometimes a large chunk of code is added as a module to another larger code base. As example of that in the 2020-10-01 release LaTeX got a new hook management system, lthooks, which was added in one go and, as with all changes to the kernel, it was added to latexrelease. However rolling back from a future date to the 2020-10-01 release didn't work because latexrelease would try to define again all those commands, which would result in many "already defined" errors and similar issues.

To solve that problem, completely new modules can be defined in latexrelease using the commands:

> `\NewModuleRelease{`⟨*initial release date*⟩`}{`⟨*name*⟩`}{`⟨*message*⟩`}`
> ⟨*module code*⟩
> `\IncludeInRelease{0000/00/00}{`⟨*name*⟩`}{`⟨*message*⟩`}`
> ⟨*undefine module code*⟩
> `\EndModuleRelease`

With that setup, the module ⟨*name*⟩ will be declared to exist only in releases equal or later ⟨*initial release date*⟩.

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

If latexrelease is rolling backwards or forwards between dates after ⟨*initial release date*⟩, then all the ⟨*module code*⟩ is skipped, except when inside ⟨*IncludeInRelease*⟩ guards, in which case the code is applied or skipped as discussed above.

If rolling forward from a date before the module's ⟨*initial release date*⟩ to a date after that, then all the ⟨*module code*⟩ is executed to define the module, and `\IncludeInRelease` guards are executed accordingly, depending on the date declared and the target date.

If latexrelease is rolling back to a date before ⟨*release date*⟩, then the code in the `\IncludeInRelease` guard dated `0000/00/00` is executed instead to undefine the module. This guard *is not* ended by the usual `\EndIncludeInRelease`, but instead by `\EndModuleRelease`.

Finally, if rolling backwards or forwards between dates both before ⟨*initial release date*⟩, the entire code between ⟨*NewModuleRelease*⟩ and ⟨*EndModuleRelease*⟩ is entirely skipped.

4.1 Example

Here is an example usage of the structure described above, as it would be used in the LaTeX kernel, taking lthooks as example:

```
%<*2ekernel|latexrelease>
\ExplSyntaxOn
%<latexrelease>\NewModuleRelease{2020/10/01}{lthooks}%
%<latexrelease>                  {The~hook~management~system}
\NewDocumentCommand \NewHook { m }
  { \hook_new:n {#1} }
%<latexrelease>\IncludeInRelease{2021/06/01}{\AddToHook}{Long~argument}
\NewDocumentCommand \AddToHook { m o +m }
  { \hook_gput_code:nnn {#1} {#2} {#3} }
%<latexrelease>\EndIncludeInRelease
%<latexrelease>
%<latexrelease>\IncludeInRelease{2020/10/01}{\AddToHook}{Long~argument}
%<latexrelease>\NewDocumentCommand \AddToHook { m o m }
%<latexrelease>  { \hook_gput_code:nnn {#1} {#2} {#3} }
%<latexrelease>\EndIncludeInRelease
%<latexrelease>
%<latexrelease>\IncludeInRelease{0000/00/00}{lthooks}{Undefine~lthooks}
%<latexrelease>\cs_undefine:N \NewHook
%<latexrelease>\cs_undefine:N \AddToHook
%<latexrelease>\EndModuleRelease
\ExplSyntaxOff
%</2ekernel|latexrelease>
```

In the example above, `\NewHook` is declared only once, and unchanged in the next release (2021/06/01 in the example), so it has no `\IncludeInRelease` guards, and will only be defined if needed. `\AddToHook`, on the other hand, changed between the two releases (made up for the example; it didn't really happen) and has an `\IncludeInRelease` block for the current release (off docstrip guards, so it goes into the kernel too), and another for the previous release (in docstrip guards so it goes only into latexrelease).

Note that in the example above, `\ExplSyntaxOn` and `\ExplSyntaxOff` were added *outside* the module code because, as discussed above, sometimes the code

outside `\IncludeInRelease` guards may be skipped, but not the code inside them, and in that case the catcodes would be wrong when defining the code.

5 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.

6 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]
```

6.1 Setup

`\sourceLaTeXdate` Store the original LaTeX format version as a number in the format `YYYYMMDD` . This macro has to be defined conditionally, so that it isn't changed in case `latexrelease.sty` is reloaded, but it can't be defined in the kernel only, otherwise `latexrelease.sty` wouldn't work in older LaTeX due to the missing macro.

```
3 \@ifundefined{sourceLaTeXdate}{%
4   \edef\sourceLaTeXdate{%
5     \expandafter\@parse@version\fmtversion//00\@nil}}{}%
```

`\IncludeInRelease` `\EndIncludeInRelease` These are defined in `ltvers.dtx`.

```
6 \DeclareOption*{%
7   \def\@IncludeInRelease#1[#2]{\@IncludeInRele@se{#1}}%
8   \let\requestedpatchdate\CurrentOption}
9 \DeclareOption{latest}{%
10  \let\requestedpatchdate\latexreleaseversion
11  \AtEndOfPackage{\def\requestedLaTeXdate{0}}}
12 \DeclareOption{current}{%
13  \let\requestedpatchdate\fmtversion
14  \AtEndOfPackage{\def\requestedLaTeXdate{0}}}
15 \let\requestedpatchdate\fmtversion
16 \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 `//`.

```
17 \def\reserved@a{%
18 \edef\requestedLaTeXdate{\the\count@}%
19 \reserved@b}
20 \def\reserved@b#1\\{%
21 \def\reserved@b{#1}%
22 \ifx\reserved@b\@empty\else
23 \PackageError{latexrelease}%
```

```
              {Unexpected option \requestedpatchdate}%
              {The option must be of the form yyyy/mm/dd or 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 \latexreleaseversion\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
```

6.2 Ignoring _new errors when rolling back

Enforce `\ExplSyntaxOn` and `\ExplSyntaxOff` to be `\relax` in latexrelease if they are not yet defined. They are later restored to be undefined if needed.

```
\csname ExplSyntaxOn\endcsname
\csname ExplSyntaxOff\endcsname
```

Define a set of changes here, but we'll only use them later to make sure they are applied after expl3 is loaded. If loading from a rather old format, we don't have `\ExplSyntaxOn` yet.

```
\begingroup
  \endlinechar=-1
```

```
  \catcode95=11 % _
  \catcode58=11 % :
  \catcode126=10 % ~
  \catcode32=09 % <space>
  \xdef\latexrelease@postltexpl{\unexpanded{%
⟨@@=latexrelease⟩
```

First we'll define a `\declarecommand` that does `\renewcommand` if the command being defined already exists, and `\newcommand` otherwise.

```
\cs_gset_protected:Npn \@@_declare_command:w
  { \@star@or@long \@@_declare_command:Nw }
\cs_gset_protected:Npn \@@_declare_command:Nw #1
  { \cs_if_exist:NTF #1 { \renew@command } { \new@command } #1 }
```

Then define a version of `\e@alloc` that checks if the control sequence being defined already exists, and if so, checks if its meaning is the same as the one that would be defined with the call to `\e@alloc`. If both tests pass, nothing is defined to save a register. This version also takes care of setting `\allocationnumber` to the value it would have after the register is allocated.

```
\cs_gset_protected:Npn \@@_e@alloc:NnnnnN #1 #2 #3 #4 #5 #6
  {
    \cs_if_free:NTF #6
      { \use:n }
      {
        \exp_after:wN \@@_e@alloc:N
          \token_to_meaning:N #6 \scan_stop: {#2} #6
      }
        { \@@_e@alloc #1 {#2} {#3} {#4} {#5} #6 }
  }
```

Walk through the meaning of the control sequence token by token, looking for the register allocation number.

```
\cs_gset_protected:Npn \@@_e@alloc:N #1
  {
    \if_int_compare:w 0 < 0
        \if_int_compare:w 10 < 9#1 ~ 1 \fi:
        \if_charcode:w " #1 1 \fi: \exp_stop_f:
      \tex_afterassignment:D \@@_e@alloc:w
      \@tempcnta #1
      \use_i:nnn
    \fi:
    \use:n
      {
        \if_meaning:w \scan_stop: #1
          \exp_after:wN \use_iv:nnnn
        \fi:
        \@@_e@alloc:N
      }
  }
```

When found, check if it is the exact same register as it would be allocated, and if it is, set `\allocationnumber` accordingly and exit, otherwise undefine the register and allocate from scratch.

```
\cs_gset_protected:Npn \@@_e@alloc:w #1 \scan_stop: #2 #3
  {
```

```
    #2 \@@_tmp:w = \@tempcnta
    \token_if_eq_meaning:NNTF #3 \@@_tmp:w
      { \int_set_eq:NN \allocationnumber \@tempcnta  \use_none:n }
      { \cs_set_eq:NN #3 \tex_undefined:D            \use:n      }
  }
```

Now create a token list to hold the list of changed commands, and define a temporary macro that will loop through the command list, store each in `\l_@@_restores_tl`, save a copy, and redefine each.

```
\tl_clear_new:N \l_@@_restores_tl
\cs_gset:Npn \@@_redefines:w #1 #2
  {
    \quark_if_recursion_tail_stop:N #1
    \tl_put_right:Nn \l_@@_restores_tl {#1}
    \cs_set_eq:cN { @@_ \cs_to_str:N #1 } #1
    \cs_set_eq:NN #1 #2
    \@@_redefines:w
  }
```

The redefinitions below are needed because:

`\__kernel_chk_if_free_cs:N` This function is used ubiquitously in the l3kernel to check if a control sequence is definable, and give an error otherwise (similar to `\@ifdefinable`). Making it a no-op is enough for most cases (except when defining new registers);

`\e@alloc` In the case of new registers, we waste an allocation number if we do `\new\meta {thing}` in a register that's already allocated, so the redefinition of `\e@alloc` checks if the new register is really necessary. This code does not clear the register, which might cause problems in the future, if a register is allocated but not properly cleared before using;

`\__kernel_msg_error:nnx` This command is used to error on already defined scan marks. Just making the error do nothing is enough, as no action is taken in that case;

`\msg_new:nnnn` Used to define new messages. Making it `_gset` is enough. Other msg commands like `\msg_new:nnn` and `\__kernel_msg_new:nnn(n)` are defined in terms of `\msg_new:nnnn`, so there is no need to change the other ones;

`\NewDocumentCommand` Used to define user-level commands in the kernel. Making it equal to `\DeclareDocumentCommand` solves the problem;

`\newcommand` Same as above.

And here we go:

```
\@@_redefines:w
  \__kernel_chk_if_free_cs:N \use_none:n
  \e@alloc \@@_e@alloc:NnnnnN
  \__kernel_msg_error:nnx \use_none:nnn
  \msg_new:nnnn \msg_gset:nnnn
  % \NewDocumentCommand \DeclareDocumentCommand % after ltcmd.dtx
  \newcommand \@@_declare_command:w
```

Temp addition ...

```
  \__kernel_msg_error:nnn  \use_none:nnn  % needed while redirect for kernel msgs doesn't wor
  \q_recursion_tail \q_recursion_tail
  \q_recursion_stop
```

Finally, redirect the error thrown by `\NewHook` to nowhere so it can be safely reused (the hook isn't redeclared if it already exists).

```
\msg_redirect_name:nnn { hooks } { exists } { none }
```

Now a one-off for `ltcmd.dtx`: we need to make `\NewDocumentCommand` not complain on an already existing command, but it has to be done after `\NewDocumentCommand` is defined, so this is separate from the `\latexrelease@postltexpl` actions above:

```
\cs_gset_protected:Npn \latexrelease@postltcmd
  {
    \@@_redefines:w
      \NewDocumentCommand \DeclareDocumentCommand
      \q_recursion_tail \q_recursion_tail
      \q_recursion_stop
  }
  }}%
\endgroup
⟨/latexrelease⟩
```

6.3 Undoing the temp modifications

If `\ExplSyntaxOn` exists (defined and not equal `\relax`), then use the expl3 restore code, otherwise restore `\ExplSyntaxOn` and `\ExplSyntaxOff` to be undefined.

```
⟨*latexrelease-finish⟩
\@ifundefined{ExplSyntaxOn}%
  {\let\ExplSyntaxOn\@undefined
   \let\ExplSyntaxOff\@undefined
   \@gobble}%
  {\ExplSyntaxOn
   \@firstofone}%
  {%
```

Now just loop through the list of redefined commands and restore their previous meanings.

```
\tl_map_inline:Nn \l_@@_restores_tl
  {
    \cs_set_eq:Nc #1 { @@_ \cs_to_str:N #1 }
    \cs_undefine:c { @@_ \cs_to_str:N #1 }
  }
\tl_clear:N \l_@@_restores_tl
```

And restore the hook error message.

```
\msg_redirect_name:nnn { hooks } { exists } { }
```

```
⟨@@=⟩
  \ExplSyntaxOff}%
⟨/latexrelease-finish⟩
```

6.4 Individual Changes

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

6.5 fixltx2e

Generate a stub fixltx2e package:

```
⟨*fixltx2e⟩
\IncludeInRelease{2015/01/01}{\fixltxe}{Old fixltx2e package}
\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}
\EndIncludeInRelease
\IncludeInRelease{0000/00/00}{\fixltxe}{Old fixltx2e package}
\def\@outputdblcol{%
  \if@firstcolumn
    \global\@firstcolumnfalse
    \global\setbox\@leftcolumn\copy\@outputbox
    \splitmaxdepth\maxdimen
    \vbadness\maxdimen
     \setbox\@outputbox\vbox{\unvbox\@outputbox\unskip}%
     \setbox\@outputbox\vsplit\@outputbox to\maxdimen
    \toks@\expandafter{\topmark}%
    \xdef\@firstcoltopmark{\the\toks@}%
    \toks@\expandafter{\splitfirstmark}%
    \xdef\@firstcolfirstmark{\the\toks@}%
    \ifx\@firstcolfirstmark\@empty
      \global\let\@setmarks\relax
    \else
      \gdef\@setmarks{%
        \let\firstmark\@firstcolfirstmark
        \let\topmark\@firstcoltopmark}%
    \fi
  \else
    \global\@firstcolumntrue
    \setbox\@outputbox\vbox{%
     \hb@xt@\textwidth{%
        \hb@xt@\columnwidth{\box\@leftcolumn \hss}%
        \hfil
        {\normalcolor\vrule \@width\columnseprule}%
        \hfil
       \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
  \@combinedblfloats
    \@setmarks
    \@outputpage
    \begingroup
      \@dblfloatplacement
      \@startdblcolumn
      \@whilesw\if@fcolmade \fi{\@outputpage\@startdblcolumn}%
    \endgroup
  \fi}
```

```
\def\end@dblfloat{%
  \if@twocolumn
    \@endfloatbox
    \ifnum\@floatpenalty <\z@
      \@largefloatcheck
      \global\dp\@currbox1sp %
      \@cons\@currlist\@currbox
      \ifnum\@floatpenalty <-\@Mii
        \penalty -\@Miv
        \@tempdima\prevdepth
        \vbox{}%
        \prevdepth\@tempdima
        \penalty\@floatpenalty
      \else
        \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
      \fi
    \fi
  \else
    \end@float
  \fi
}
\def\@testwrongwidth #1{%
  \ifdim\dp#1=\f@depth
  \else
    \global\@testtrue
  \fi}
\let\f@depth\z@
\def\@dblfloatplacement{\global\@dbltopnum\c@dbltopnumber
   \global\@dbltoproom \dbltopfraction\@colht
   \@textmin \@colht
   \advance \@textmin -\@dbltoproom
   \@fpmin \dblfloatpagefraction\textheight
   \@fptop \@dblfptop
   \@fpsep \@dblfpsep
   \@fpbot \@dblfpbot
   \def\f@depth{1sp}}
\def \@doclearpage {%
     \ifvoid\footins
       \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
       \setbox\@tempboxa\box\@cclv
       \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
       \global \let \@toplist \@empty
       \global \let \@botlist \@empty
       \global \@colroom \@colht
       \ifx \@currlist\@empty
       \else
          \@latexerr{Float(s) lost}\@ehb
          \global \let \@currlist \@empty
       \fi
       \@makefcolumn\@deferlist
       \@whilesw\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
       \if@twocolumn
         \if@firstcolumn
           \xdef\@deferlist{\@dbltoplist\@deferlist}%
```

```
           \global \let \@dbltoplist \@empty
           \global \@colht \textheight
           \begingroup
              \@dblfloatplacement
              \@makefcolumn\@deferlist
              \@whilesw\if@fcolmade \fi{\@outputpage
                                        \@makefcolumn\@deferlist}%
           \endgroup
         \else
           \vbox{}\clearpage
         \fi
       \fi
       \ifx\@deferlist\@empty \else\clearpage \fi
     \else
       \setbox\@cclv\vbox{\box\@cclv\vfil}%
       \@makecol\@opcol
       \clearpage
     \fi
}
\def \@startdblcolumn {%
  \@tryfcolumn \@deferlist
  \if@fcolmade
  \else
    \begingroup
      \let \reserved@b \@deferlist
      \global \let \@deferlist \@empty
      \let \@elt \@sdblcolelt
      \reserved@b
    \endgroup
  \fi
}
\def\@addtonextcol{%
  \begingroup
   \@insertfalse
   \@setfloattypecounts
   \ifnum \@fpstype=8
   \else
     \ifnum \@fpstype=24
     \else
       \@flsettextmin
       \@reqcolroom \ht\@currbox
       \advance \@reqcolroom \@textmin
       \ifdim \@colroom>\@reqcolroom
         \@flsetnum \@colnum
         \ifnum\@colnum>\z@
            \@bitor\@currtype\@deferlist
            \@testwrongwidth\@currbox
            \if@test
            \else
              \@addtotoporbot
            \fi
         \fi
       \fi
     \fi
```

```
   \fi
   \if@insert
   \else
     \@cons\@deferlist\@currbox
   \fi
  \endgroup
}
\def\@addtodblcol{%
  \begingroup
   \@insertfalse
   \@setfloattypecounts
   \@getfpsbit \tw@
   \ifodd\@tempcnta
     \@flsetnum \@dbltopnum
     \ifnum \@dbltopnum>\z@
       \@tempswafalse
       \ifdim \@dbltoproom>\ht\@currbox
         \@tempswatrue
       \else
         \ifnum \@fpstype<\sixt@@n
           \advance \@dbltoproom \@textmin
           \ifdim \@dbltoproom>\ht\@currbox
             \@tempswatrue
           \fi
           \advance \@dbltoproom -\@textmin
         \fi
       \fi
       \if@tempswa
           \@bitor \@currtype \@deferlist
          \@testwrongwidth\@currbox
           \if@test
           \else
              \@tempdima -\ht\@currbox
              \advance\@tempdima
                -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
                                          \dblfloatsep \fi
              \global \advance \@dbltoproom \@tempdima
              \global \advance \@colht \@tempdima
              \global \advance \@dbltopnum \m@ne
              \@cons \@dbltoplist \@currbox
              \@inserttrue
           \fi
       \fi
     \fi
   \fi
   \if@insert
   \else
     \@cons\@deferlist\@currbox
   \fi
  \endgroup
}
\def \@addtocurcol {%
   \@insertfalse
   \@setfloattypecounts
```

```
\ifnum \@fpstype=8
\else
  \ifnum \@fpstype=24
  \else
    \@flsettextmin
    \advance \@textmin \@textfloatsheight
    \@reqcolroom \@pageht
    \ifdim \@textmin>\@reqcolroom
      \@reqcolroom \@textmin
    \fi
    \advance \@reqcolroom \ht\@currbox
    \ifdim \@colroom>\@reqcolroom
      \@flsetnum \@colnum
      \ifnum \@colnum>\z@
        \@bitor\@currtype\@deferlist
       \@testwrongwidth\@currbox
        \if@test
        \else
          \@bitor\@currtype\@botlist
          \if@test
            \@addtobot
          \else
            \ifodd \count\@currbox
              \advance \@reqcolroom \intextsep
              \ifdim \@colroom>\@reqcolroom
                \global \advance \@colnum \m@ne
                \global \advance \@textfloatsheight \ht\@currbox
                \global \advance \@textfloatsheight 2\intextsep
                \@cons \@midlist \@currbox
                \if@nobreak
                  \nobreak
                  \@nobreakfalse
                  \everypar{}%
                \else
                  \addpenalty \interlinepenalty
                \fi
                \vskip \intextsep
                \box\@currbox
                \penalty\interlinepenalty
                \vskip\intextsep
                \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi
                \outputpenalty \z@
                \@inserttrue
              \fi
            \fi
            \if@insert
            \else
              \@addtotoporbot
            \fi
          \fi
        \fi
      \fi
    \fi
```

```
    \fi
    \if@insert
    \else
      \@resethfps
      \@cons\@deferlist\@currbox
    \fi
}
\def\@xtryfc #1{%
  \@next\reserved@a\@trylist{}{}%
  \@currtype \count #1%
  \divide\@currtype\@xxxii
  \multiply\@currtype\@xxxii
  \@bitor \@currtype \@failedlist
  \@testfp #1%
  \@testwrongwidth #1%
  \ifdim \ht #1>\@colht
     \@testtrue
  \fi
  \if@test
    \@cons\@failedlist #1%
  \else
    \@ytryfc #1%
  \fi}
\def\@ztryfc #1{%
  \@tempcnta\count #1%
  \divide\@tempcnta\@xxxii
  \multiply\@tempcnta\@xxxii
  \@bitor \@tempcnta {\@failedlist \@flfail}%
  \@testfp #1%
  \@testwrongwidth #1%
  \@tempdimb\@tempdima
  \advance\@tempdimb\ht #1%
  \advance\@tempdimb\@fpsep
  \ifdim \@tempdimb >\@colht
    \@testtrue
  \fi
  \if@test
    \@cons\@flfail #1%
  \else
    \@cons\@flsucceed #1%
    \@tempdima\@tempdimb
  \fi}
\def\@{\spacefactor\@m{}}
\def\@tempa#1#2{#1#2\relax}
\ifx\setlength\@tempa
  \def\setlength#1#2{#1 #2\relax}
\fi
\def\addpenalty#1{%
  \ifvmode
    \if@minipage
    \else
      \if@nobreak
      \else
        \ifdim\lastskip=\z@
```

```
          \penalty#1\relax
        \else
          \@tempskipb\lastskip
          \begingroup
            \advance \@tempskipb
              \ifdim\prevdepth>\maxdepth\maxdepth\else
                 \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
               \fi
             \vskip -\@tempskipb
             \penalty#1%
             \vskip\@tempskipb
          \endgroup
          \vskip -\@tempskipb
          \vskip \@tempskipb
        \fi
      \fi
    \fi
  \else
    \@noitemerr
  \fi}
\def\@fnsymbol#1{%
   \ifcase#1\or \TextOrMath\textasteriskcentered *\or
   \TextOrMath \textdagger \dagger\or
   \TextOrMath \textdaggerdbl \ddagger \or
   \TextOrMath \textsection  \mathsection\or
   \TextOrMath \textparagraph \mathparagraph\or
   \TextOrMath \textbardbl \|\or
   \TextOrMath {\textasteriskcentered\textasteriskcentered}{**}\or
   \TextOrMath {\textdagger\textdagger}{\dagger\dagger}\or
   \TextOrMath {\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger}\else
   \@ctrerr \fi
}
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname eTeXversion\endcsname\relax
\DeclareRobustCommand\TextOrMath{%
  \ifmmode  \expandafter\@secondoftwo
  \else     \expandafter\@firstoftwo  \fi}
\protected@edef\TextOrMath#1#2{\TextOrMath{#1}{#2}}
\else
\protected\expandafter\def\csname TextOrMath\space\endcsname{%
  \ifmmode  \expandafter\@secondoftwo
  \else     \expandafter\@firstoftwo  \fi}
\edef\TextOrMath#1#2{%
  \expandafter\noexpand\csname TextOrMath\space\endcsname
  {#1}{#2}}
\fi
\def\@esphack{%
  \relax
  \ifhmode
    \spacefactor\@savsf
    \ifdim\@savsk>\z@
      \nobreak \hskip\z@skip  % <------
      \ignorespaces
    \fi
```

```
  \fi}
\def\@Esphack{%
  \relax
  \ifhmode
    \spacefactor\@savsf
    \ifdim\@savsk>\z@
      \nobreak \hskip\z@skip  % <------
      \@ignoretrue
      \ignorespaces
    \fi
   \fi}
\DeclareRobustCommand\em
        {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
                        \eminnershape \else \itshape \fi}
\def\eminnershape{\upshape}
\DeclareRobustCommand*\textsubscript[1]{%
  \@textsubscript{\selectfont#1}}
\def\@textsubscript#1{%
  {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\z@#1}}}}}
\def\@DeclareMathSizes #1#2#3#4#5{%
  \@defaultunits\dimen@ #2pt\relax\@nnil
  \if $#3$%
    \expandafter\let\csname S@\strip@pt\dimen@\endcsname\math@fontsfalse
  \else
    \@defaultunits\dimen@ii #3pt\relax\@nnil
    \@defaultunits\@tempdima #4pt\relax\@nnil
    \@defaultunits\@tempdimb #5pt\relax\@nnil
    \toks@{#1}%
    \expandafter\xdef\csname S@\strip@pt\dimen@\endcsname{%
      \gdef\noexpand\tf@size{\strip@pt\dimen@ii}%
      \gdef\noexpand\sf@size{\strip@pt\@tempdima}%
      \gdef\noexpand\ssf@size{\strip@pt\@tempdimb}%
      \the\toks@
    }%
  \fi
}
\providecommand*\MakeRobust[1]{%
  \@ifundefined{\expandafter\@gobble\string#1}{%
    \@latex@error{The control sequence `\string#1' is undefined!%
      \MessageBreak There is nothing here to make robust}%
    \@eha
  }%
  {%
    \@ifundefined{\expandafter\@gobble\string#1\space}%
    {%
      \expandafter\let\csname
      \expandafter\@gobble\string#1\space\endcsname=#1%
      \edef\reserved@a{\string#1}%
      \def\reserved@b{#1}%
      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
      \edef#1{%
        \ifx\reserved@a\reserved@b
          \noexpand\x@protect\noexpand#1%
        \fi
```

```
        \noexpand\protect\expandafter\noexpand
        \csname\expandafter\@gobble\string#1\space\endcsname}%
    }%
    {\@latex@info{The control sequence `\string#1' is already robust}}%
   }%
}
\MakeRobust\(
\MakeRobust\)
\MakeRobust\[
\MakeRobust\]
\MakeRobust\makebox
\MakeRobust\savebox
\MakeRobust\framebox
\MakeRobust\parbox
\MakeRobust\rule
\MakeRobust\raisebox
\def\@xfloat #1[#2]{%
  \@nodocument
  \def \@captype {#1}%
   \def \@fps {#2}%
   \@onelevel@sanitize \@fps
   \def \reserved@b {!}%
   \ifx \reserved@b \@fps
     \@fpsadddefault
   \else
     \ifx \@fps \@empty
       \@fpsadddefault
     \fi
   \fi
   \ifhmode
     \@bsphack
     \@floatpenalty -\@Mii
   \else
     \@floatpenalty-\@Miii
   \fi
  \ifinner
     \@parmoderr\@floatpenalty\z@
  \else
    \@next\@currbox\@freelist
      {%
       \@tempcnta \sixt@@n
       \expandafter \@tfor \expandafter \reserved@a
         \expandafter :\expandafter =\@fps
         \do
          {%
           \if \reserved@a h%
             \ifodd \@tempcnta
             \else
               \advance \@tempcnta \@ne
             \fi
           \else\if \reserved@a t%
             \@setfpsbit \tw@
           \else\if \reserved@a b%
             \@setfpsbit 4%
```

```
            \else\if \reserved@a p%
              \@setfpsbit 8%
            \else\if \reserved@a !%
              \ifnum \@tempcnta>15
                \advance\@tempcnta -\sixt@@n\relax
              \fi
            \else
              \@latex@error{Unknown float option `\reserved@a'}%
              {Option `\reserved@a' ignored and `p' used.}%
              \@setfpsbit 8%
            \fi\fi\fi\fi\fi
            }%
        \@tempcntb \csname ftype@\@captype \endcsname
        \multiply \@tempcntb \@xxxii
        \advance \@tempcnta \@tempcntb
        \global \count\@currbox \@tempcnta
        }%
    \@fltovf
  \fi
  \global \setbox\@currbox
    \color@vbox
      \normalcolor
      \vbox \bgroup
        \hsize\columnwidth
        \@parboxrestore
        \@floatboxreset
}
 \def\@stpelt#1{\global\csname c@#1\endcsname \m@ne\stepcounter{#1}}
\EndIncludeInRelease
```
⟨/fixltx2e⟩