The ltfilehook documentation*

Frank Mittelbach, Phelype Oleinik, LATEX Project Team

August 22, 2024

Contents

1	Intr	roduction	2
	1.1	Provided hooks	2
	1.2	General hooks for file reading	2
	1.3	Hooks for package and class files	3
	1.4	Hooks for \include files	4
	1.5	High-level interfaces for LATEX	5
	1.6	Kernel, class, and package interfaces for IAT_EX	6
	1.7	A sample package for structuring the log output	6
2	The	e Implementation	7
	2.1	Document and package-level commands	7
	2.2	expl3 helpers	8
	2.3	Declaring the file-related hooks	11
	2.4	Patching LATEX's \InputIfFileExists command	11
	2.5	Declaring a file substitution	13
	2.6	Selecting a file (\set@curr@file)	15
	2.7	Replacing a file and detecting loops	18
	2	.7.1 The Tortoise and Hare algorithm	19
	2.8	Preventing a package from loading	21
	2.9	High-level interfaces for LATEX	22
	2.10	Internal commands needed elsewhere	22
3	A s	ample package for structuring the log output	23
4	Pac	kage emulations	24
	4.1	Package atveryend emulation	24

Index

25

*This code has version v1.0o dated 2024/03/13, $\ensuremath{\textcircled{O}}$ IATEX Project.

1 Introduction

1.1 Provided hooks

The code offers a number of hooks into which packages (or the user) can add code to support different use cases. Many hooks are offered as pairs (i.e., the second hook is reversed. Also important to know is that these pairs are properly nested with respect to other pairs of hooks.

There are hooks that are executed for all files of a certain type (if they contain code), e.g., for all "include files" or all "packages", and there are also hooks that are specific to a single file, e.g., do something after the package foo.sty has been loaded.

1.2 General hooks for file reading

There are four hooks that are called for each file that is read using document-level commands such as \input, \include, \usepackage, etc. They are not called for files read using internal low-level methods, such as \@input or \openin.

file/before
file/.../before
file/.../after
file/after

These are:

- file/before, file/{file-name}/before These hooks are executed in that order just before the file is loaded for reading. The code of the first hook is used with every file, while the second is executed only for the file with matching (file-name) allowing you to specify code that only applies to one file.
- file/(file-name)/after, file/after These hooks are after the file with name (file-name) has been fully consumed. The order is swapped (the specific one comes first) so that the /before and /after hooks nest properly, which is important if any of them involve grouping (e.g., contain environments, for example). Furthermore both hooks are reversed hooks to support correct nesting of different packages adding code to both /before and /after hooks.

So the overall sequence of hook processing for any file read through the user interface commands of LAT_{FX} is:

```
\UseHook{(file/before)}
\UseHook{(file/(file name)/before)}
  (file contents)
\UseHook{(file/(file name)/after)}
\UseHook{(file/after)}
```

The file hooks only refer to the file by its name and extension, so the $\langle file name \rangle$ should be the file name as it is on the filesystem with extension (if any) and without paths. Different from \input and similar commands, the .tex extension is not assumed in hook $\langle file name \rangle$, so .tex files must be specified with their extension to be recognized. Files within subfolders should also be addressed by their name and extension only.

Extensionless files also work, and should then be given without extension. Note however that T_EX prioritizes .tex files, so if two files foo and foo.tex exist in the search path, only the latter will be seen.

When a file is input, the $\langle file name \rangle$ is available in CurrentFile, which is then used when accessing the file/ $\langle file name \rangle$ /before and file/ $\langle file name \rangle$ /after.

- \CurrentFile The name of the file about to be read (or just finished) is available to the hooks through \CurrentFile (there is no expl3 name for it for now). The file is always provided with its extension, i.e., how it appears on your hard drive, but without any specified path to it. For example, \input{sample} and \input{app/sample.tex} would both have \CurrentFile being sample.tex.
- \CurrentFilePathThe path to the current file (complement to \CurrentFile) is available in \CurrentFilePathif needed. The paths returned in \CurrentFilePath are only user paths, given through\input@path (or expl3's equivalent \l_file_search_path_seq) or by directly typing inthe path in the \input command or equivalent. Files located by kpsewhich get the pathadded internally by the TEX implementation, so at the macro level it looks as if the filewere in the current folder, so the path in \CurrentFilePath is empty in these cases(package and class files, mostly).

\CurrentFileUsed In normal circumstances these are identical to \CurrentFile and \CurrentFilePath. \CurrentFilePathUsed They will differ when a file substitution has occurred for \CurrentFile. In that case, \CurrentFileUsed and \CurrentFilePathUsed will hold the actual file name and path loaded by IATEX, while \CurrentFile and \CurrentFilePath will hold the names that were asked for. Unless doing very specific work on the file being read, \CurrentFile and \CurrentFilePath should be enough.

1.3 Hooks for package and class files

Commands to load package and class files (e.g., \usepackage, \RequirePackage, \LoadPackageWithOptions, etc.) offer the hooks from section 1.2 when they are used to load a package or class file, e.g., file/array.sty/after would be called after the array package got loaded. But as packages and classes form as special group of files, there are some additional hooks available that only apply when a package or class is loaded.

These	are:

package/before

package/after	
package//before	<pre>package/before, package/after These hooks are called for each package being loaded.</pre>
package//after class/before class/after	<pre>package/(name)/before, package/(name)/after These hooks are additionally called if the package name is (name) (without extension).</pre>
class//before class//after	class/before, class/after These hooks are called for each class being loaded.
	$\verb+class/(name)/before, class/(name)/after These hooks are additionally called if the class name is (name) (without extension).$
	All /after hooks are implemented as reversed hooks. The overall sequence of execution for \usepackage and friends is therefore:

```
\UseHook{(package/before)}
\UseHook{(package/(package name)/before)}
\UseHook{(file/before)}
```

```
\UseHook{(file/(package name).sty/before)}
  (package contents)
\UseHook{(file/(package name).sty/after)}
\UseHook{(file/after)}
```

code from \AtEndOfPackage if used inside the package

```
\UseHook{\package/\package name\/after\}
\UseHook{\package/after\}
```

and similar for class file loading, except that package/ is replaced by class/ and \AtEndOfPackage by \AtEndOfClass.

If a package or class is not loaded (or it was loaded before the hooks were set) none of the hooks are executed!

All class or package hooks involving the name of the class or package are implemented as one-time hooks, whereas all other such hooks are normal hooks. This allows for the following use case

```
\AddToHook{package/varioref/after}
  { ... apply my customizations if the package gets
        loaded (or was loaded already) ... }
```

without the need to first test if the package is already loaded.

1.4 Hooks for \include files

To manage $\ \ EAT_EX$ issues a $\ \ earpage$ before and after loading such a file. Depending on the use case one may want to execute code before or after these $\ \ earpage$ especially for the one that is issued at the end.

Executing code before the final \clearpage, means that the code is processed while the last page of the included material is still under construction. Executing code after it means that all floats from inside the include file are placed (which might have added further pages) and the final page has finished.

Because of these different scenarios we offer hooks in three places.¹ None of the hooks are executed when an \include file is bypassed because of an \includeonly declaration. They are, however, all executed if LATEX makes an attempt to load the \include file (even if it doesn't exist and all that happens is "No file $\langle filename \rangle$.tex").

¹If you want to execute code before the first \clearpage there is no need to use a hook—you can write it directly in front of the \include.

include/before These are: include/.../before

include/end include/.../end include/after include/.../after

include/before, include/(name)/before These hooks are executed (in that order) after the initial \clearpage and after .aux file is changed to use (name).aux, but before the (name).tex file is loaded. In other words they are executed at the very beginning of the first page of the \include file.

- include/ $\langle name \rangle$ /end, include/end These hooks are executed (in that order) after IAT_EX has stopped reading from the \include file, but before it has issued a \clearpage to output any deferred floats.
- include/excluded, include/(name)/excluded The above hooks for \include files are
 only executed when the file is loaded (or more exactly the load is attempted).
 If, however, the \include file is explicitly excluded (through an \includeonly
 statement) the above hooks are bypassed and instead the include/excluded hook
 followed by the include/(name)/excluded hook are executed. This happens after
 IATEX has loaded the .aux file for this include file, i.e., after IATEX has updated its
 counters to pretend that the file was seen.

All include hooks involving the name of the included file are implemented as onetime hooks (whereas all other such hooks are normal hooks).

If you want to execute code that is run for every \include regardless of whether or not it is excluded, use the cmd/include/before or cmd/include/after hooks.

We do not provide any additional wrappers around the hooks (like filehook or scrlfile do) because we believe that for package writers the high-level commands from the hook management, e.g., \AddToHook, etc. are sufficient and in fact easier to work with, given that the hooks have consistent naming conventions.

²For that reason another $\ensuremath{\clearpage}$ is executed after these hooks which normally does nothing, but starts a new page if further material got added this way.

1.6 Kernel, class, and package interfaces for LATEX

If $\langle file \rangle$ is requested for loading replace it with $\langle replacement-file \rangle$. $\land CurrentFile$ remains pointing to $\langle file \rangle$ but $\land CurrentFileUsed$ will show the file actually loaded.

The main use case for this declaration is to provide a corrected version of a package that can't be changed (due to its license) but no longer functions because of LAT_EX kernel changes, for example, or to provide a version that makes use of new kernel functionality while the original package remains available for use with older releases. As such it is mainly meant for use in the LAT_EX kernel but other use cases are conceivable.

The $\mbox{undeclare0file0substitution}$ declaration undoes a substitution made earlier.

Please do not misuse this functionality and replace a file with another unless if really needed and only if the new version is implementing the same functionality as the original one!

\disable@package@load	\disable@package@load	$\{\langle package \rangle\}$	$\{\langle alternate-code \rangle\}$
\reenable@package@load	\reenable@package@load	$\{\langle package \rangle\}$	

If $\langle package \rangle$ is requested, do not load it but instead run $\langle alternate-code \rangle$ which could issue a warning, error or any other code.

The main use case is for classes that want to restrict the set of supported packages or contain code that make the use of some packages impossible. So rather than waiting until the document breaks they can set up informative messages why certain packages are not available.

The function is only implemented for packages not for arbitrary files and again it should only be applied if there are good reasons for doing this.³

1.7 A sample package for structuring the log output

As an application we provide the package structuredlog that adds lines to the .log when a file is opened and closed for reading keeping track of nesting level es well. For example, for the current document it adds the lines

```
= (LEVEL 1 START) t1lmr.fd
= (LEVEL 1 STOP) t1lmr.fd
= (LEVEL 1 STOP) supp-pdf.mkii
= (LEVEL 1 STOP) supp-pdf.mkii
= (LEVEL 1 START) nameref.sty
== (LEVEL 2 START) refcount.sty
== (LEVEL 2 STOP) refcount.sty
== (LEVEL 2 STOP) gettitlestring.sty
== (LEVEL 1 STOP) nameref.sty
= (LEVEL 1 START) ltfilehook-doc.out
```

³Just to be sure: "I don't like this package by somebody else" is not a good one :-)

```
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.out
 (LEVEL 1 STOP) ltfilehook-doc.out
 (LEVEL 1 START) ltfilehook-doc.hd
= (LEVEL 1 STOP) ltfilehook-doc.hd
= (LEVEL 1 START) ltfilehook.dtx
== (LEVEL 2 START) ot1lmr.fd
== (LEVEL 2 STOP) ot11mr.fd
== (LEVEL 2 START) omllmm.fd
== (LEVEL 2 STOP) omllmm.fd
== (LEVEL 2 START) omslmsy.fd
== (LEVEL 2 STOP) omslmsy.fd
== (LEVEL 2 START) omxlmex.fd
== (LEVEL 2 STOP) omxlmex.fd
== (LEVEL 2 START) umsa.fd
== (LEVEL 2 STOP) umsa.fd
== (LEVEL 2 START) umsb.fd
== (LEVEL 2 STOP) umsb.fd
== (LEVEL 2 START) ts1lmr.fd
== (LEVEL 2 STOP) ts1lmr.fd
== (LEVEL 2 START) t11mss.fd
== (LEVEL 2 STOP) t11mss.fd
= (LEVEL 1 STOP) ltfilehook.dtx
```

Thus if you inspect an issue in the .log it is easy to figure out in which file it occurred, simply by searching back for LEVEL and if it is a STOP then remove 1 from the level value and search further for LEVEL with that value which should then be the START level of the file you are in.

2 The Implementation

- 1 (*2ekernel)
- 2 (@@=filehook)

2.1 Document and package-level commands

\CurrentFile \CurrentFilePath \CurrentFileUsed \CurrentFilePathUsed User-level macros that hold the current file name and file path. These are used internally as well because the code takes care to protect against a possible redefinition of these macros in the loaded file (it's necessary anyway to make hooks work with nested \input). The versions $\...$ Used hold the *actual* file name and path that is loaded by IATEX, whereas the other two hold the name as requested. They will differ in case there's a file substitution.

{\CurrentFile}{Hook management file}%

- ₃ (/2ekernel)
- 4 (*2ekernel | latexrelease)
- 5 (latexrelease) \IncludeInRelease{2020/10/01}%
- 6 (latexrelease)
- 7 \ExplSyntaxOn
- 8 \tl_new:N \CurrentFile
- 9 \tl_new:N \CurrentFilePath
- 10 \tl_new:N \CurrentFileUsed
- 11 \tl_new:N \CurrentFilePathUsed

```
12 \ExplSyntaxOff
13 (/2ekernel | latexrelease)
14 (latexrelease) \EndIncludeInRelease
15 (latexrelease)\IncludeInRelease{0000/00/00}%
16 (latexrelease)
                                  {\CurrentFile}{Hook management file}%
17 (latexrelease)
18 (latexrelease)\let \CurrentFile
                                             \@undefined
19 (latexrelease) \let \CurrentFilePath
                                             \@undefined
20 (latexrelease)\let \CurrentFileUsed
                                             \@undefined
21 (latexrelease) \let \CurrentFilePathUsed \Qundefined
22 (latexrelease)
23 (latexrelease) \EndIncludeInRelease
24 (*2ekernel)
```

 $(\textit{End of definition for \CurrentFile and others. These functions are documented on page 3.})$

2.2 expl3 helpers

- $_{25}$ $\langle /2ekernel \rangle$
- $_{26}$ (*2ekernel | latexrelease)
- $_{\rm 27}\ {\rm (latexrelease} \ {\rm IncludeInRelease} 2020/10/01} \ {\rm (latexrelease} \ {\rm Control of Control of$
- 28 (latexrelease) {__filehook_file_parse_full_name:nN}{File helpers}%
- 29 \ExplSyntaxOn

__filehook_file_parse_full_name:nN
__filehook_full_name:nn

A utility macro to trigger expl3's file-parsing and lookup, and return a normalized representation of the file name. If the queried file doesn't exist, no normalization takes place. The output of $_filehook_file_parse_full_name:nN$ is passed on to the #2—a 3-argument macro that takes the $\langle path \rangle$, $\langle base \rangle$, and $\langle ext \rangle$ parts of the file name.

```
\cs_new:Npn \__filehook_file_parse_full_name:nN #1
30
    {
31
      \exp_args:Nf \file_parse_full_name_apply:nN
32
         {
33
           \exp_args:Nf \__filehook_full_name:nn
34
             { \file_full_name:n {#1} } {#1}
35
        7
36
    }
37
  \cs_new:Npn \__filehook_full_name:nn #1 #2
38
39
    {
      \tl_if_empty:nTF {#1}
40
         { \tl_trim_spaces:n {#2} }
41
        { \tl_trim_spaces:n {#1} }
42
    }
43
```

(End of definition for __filehook_file_parse_full_name:nN and __filehook_full_name:nn.)

__filehook_if_no_extension:nTF __filehook_drop_extension:N Some actions depend on whether the file extension was explicitly given, and sometimes the extension has to be removed. The macros below use __filehook_file_parse_-full_name:nN to split up the file name and either check if $\langle ext \rangle$ (#3) is empty, or discard it.

```
\cs_new_protected:Npn \__filehook_drop_extension:N #1
49
50
    {
      \tl_gset:Nx #1
51
        ſ
52
           \exp_args:NV \__filehook_file_parse_full_name:nN #1
53
             \__filehook_drop_extension_aux:nnn
54
        }
55
    }
56
  \cs_new:Npn \__filehook_drop_extension_aux:nnn #1 #2 #3
57
     { \tl_if_empty:nF {#1} { #1 / } #2 }
58
```

(End of definition for __filehook_if_no_extension:nTF and __filehook_drop_extension:N.)

\g__filehook_input_file_seq
 \l__filehook_internal_tl
 __filehook_file_push:
 __filehook_file_pop:
 _filehook_file_pop_assign:nnnn

Yet another stack, to keep track of \CurrentFile and \CurrentFilePath with nested \inputs. At the beginning of \InputIfFileExists, the current value of \CurrentFilePath and \CurrentFile is pushed to \g_filehook_input_file_seq, and at the end, it is popped and the value reassigned. Some other places don't use \InputIfFileExists directly (\include) or need \CurrentFile earlier (\@onefilewithoptions), so these are manually used elsewhere as well.

```
59 \tl_new:N \l_filehook_internal_tl
  \seq_if_exist:NF \g__filehook_input_file_seq
    { \seq_new:N \g__filehook_input_file_seq }
61
62
  \cs_new_protected:Npn \__filehook_file_push:
63
    {
      \seq_gpush:Nx \g__filehook_input_file_seq
64
        ł
65
          { \CurrentFilePathUsed } { \CurrentFileUsed }
66
          { \CurrentFilePath
                                 } { \CurrentFile
                                                         }
67
        7
68
    }
69
  \cs_new_protected:Npn \__filehook_file_pop:
70
71
    {
      \seq_gpop:NNTF \g_filehook_input_file_seq \l_filehook_internal_tl
        { \exp_after:wN \__filehook_file_pop_assign:nnnn \l__filehook_internal_tl }
73
        {
74
          \msg_error:nnn { latex2e } { should-not-happen }
75
            { Tried-to-pop-from-an-empty-file-name-stack. }
76
        }
77
    }
78
  \cs_new_protected:Npn \__filehook_file_pop_assign:nnnn #1 #2 #3 #4
79
80
    ł
      \tl_set:Nn \CurrentFilePathUsed {#1}
81
      \tl_set:Nn \CurrentFileUsed {#2}
82
      \tl_set:Nn \CurrentFilePath {#3}
83
      \tl_set:Nn \CurrentFile {#4}
84
    }
85
86 \ExplSyntaxOff
```

(End of definition for \g_filehook_input_file_seq and others.)

```
87 (/2ekernel | latexrelease)
```

 $_{\tt 88}~{\rm (latexrelease) \ EndIncludeInRelease}$

When rolling forward the following expl3 functions may not be defined. If we roll back the code does nothing.

```
(latexrelease)\IncludeInRelease{2020/10/01}%
89
   (latexrelease)
                                    {\file_parse_full_name_apply:nN}{Roll forward help}%
90
   (latexrelease)
91
   (latexrelease)\ExplSyntaxOn
92
   (latexrelease)\cs_if_exist:NF\file_parse_full_name_apply:nN
93
   (latexrelease){
   (latexrelease)\cs_new:Npn \file_parse_full_name_apply:nN #1
95
   (latexrelease)
                     \exp_args:Ne \__file_parse_full_name_auxi:nN
   (latexrelease)
97
   (latexrelease)
                       { \__kernel_file_name_sanitize:n {#1} }
98
   (latexrelease)
                  }
   (latexrelease)\cs_new:Npn \__file_parse_full_name_auxi:nN #1
100
   (latexrelease)
101
                  -{
                     \__file_parse_full_name_area:nw { } #1
   (latexrelease)
102
   (latexrelease)
                       / \s__file_stop
103
   (latexrelease)
104
   (latexrelease)\cs_new:Npn \__file_parse_full_name_area:nw #1 #2 / #3 \s__file_stop
105
   (latexrelease)
106
                  {
                     \tl_if_empty:nTF {#3}
   (latexrelease)
107
                       { \__file_parse_full_name_base:nw { } #2 . \s__file_stop {#1} }
   (latexrelease)
108
                       { \__file_parse_full_name_area:nw { #1 / #2 }
   (latexrelease)
109
   (latexrelease)
                                                               #3 \s__file_stop }
110
   (latexrelease)
                  }
   (latexrelease)\cs_new:Npn \__file_parse_full_name_base:nw #1 #2 . #3 \s__file_stop
   (latexrelease)
                  {
113
   (latexrelease)
                     \tl_if_empty:nTF {#3}
114
   (latexrelease)
115
                       ł
   (latexrelease)
                         \tl_if_empty:nTF {#1}
116
   (latexrelease)
117
                            £
   (latexrelease)
                              \tl_if_empty:nTF {#2}
   (latexrelease)
                                { \__file_parse_full_name_tidy:nnnN { } { } }
119
                                { \__file_parse_full_name_tidy:nnnN { .#2 } { } }
   (latexrelease)
120
   (latexrelease)
                            7
121
   (latexrelease)
                            { \__file_parse_full_name_tidy:nnnN {#1} { .#2 } }
122
   (latexrelease)
                       }
123
   (latexrelease)
                       { \__file_parse_full_name_base:nw { #1 . #2 }
124
   (latexrelease)
                                                               #3 \s__file_stop }
125
   (latexrelease)
                  }
126
   (latexrelease)\cs_new:Npn \__file_parse_full_name_tidy:nnnN #1 #2 #3 #4
128
   (latexrelease)
                  ſ
129
   (latexrelease)
                     \exp_args:Nee #4
   (latexrelease)
130
                         \str_if_eq:nnF {#3} { / } { \use_none:n }
   (latexrelease)
131
   (latexrelease)
                         #3 \prg_do_nothing:
132
   (latexrelease)
                       7
133
   (latexrelease)
                       { \use_none:n #1 \prg_do_nothing: }
134
   (latexrelease)
                       {#2}
135
   (latexrelease)
                  }
136
   \langle |atexrelease \rangle \}
137
   (latexrelease)\ExplSyntaxOff
138
139
   (latexrelease)
140
   (latexrelease)\EndIncludeInRelease
141 (*2ekernel)
142 (@@=)
```

2.3 Declaring the file-related hooks

These hooks have names with three-parts that start with file/, include/, class/ or package/ and end with /before or /after (or /end in the case of include/). They are all generic hooks so will be declared only if code is added to them; this declaration is done for you automatically and, indeed, they should not be declared explicitly.

Those named .../after and include/.../end are, when code is added, declared as reversed hooks.

2.4 Patching LATEX's \InputIfFileExists command

Most of what we have to do is adding UseHook into several IAT_EX 2_{ε} core commands, because of some circular dependencies in the kernel we do this only now and not in ltfiles.

\InputIfFileExists \@input@file@exists@with@hooks \unqu@tefilef@und \InputIfFileExists loads any file if it is available so we have to add the hooks file/before and file/after in the right places. If the file doesn't exist no hooks should be executed.

```
143 (/2ekernel)
144 (latexrelease) \IncludeInRelease {2020/10/01}%
                          {\InputIfFileExists}{Hook management (files)}%
145 (latexrelease)
  (*2ekernel | latexrelease)
146
147 \let\InputIfFileExists\@undefined
   \DeclareRobustCommand \InputIfFileExists[2]{%
148
     \IfFileExists{#1}%
149
150
       {%
          \@expl@@@filehook@file@push@@
151
         \@filehook@set@CurrentFile
152
```

We pre-expand \@filef@und so that in case another file is loaded in the true branch of \InputIfFileExists, these don't change their value meanwhile. This isn't a worry with \CurrentFile... because they are kept in a stack.

153	\expandafter\@swaptwoargs\expandafter
154	{\expandafter\@input@file@exists@with@hooks
155	\expandafter{\@filef@und}}%
156	{ # 2}%
157	\@expl@@@filehook@file@pop@@
158	}%
159	}
160	\def\@input@file@exists@with@hooks#1{%

If the file exists then **\CurrentFile** holds its name. But we can't rely on that still being true after the file has been processed. Thus for using the name in the file hooks we need to preserve the name and then restore it for the file/.../after hook.

The hook always refers to the file requested by the user. The hook is *always* loaded for \CurrentFile which usually is the same as \CurrentFileUsed. In the case of a file replacement, the \CurrentFileUsed holds the actual file loaded. In any case the file names are normalized so that the hooks work on the real file name, rather than what the user typed in.

expl3's \file_full_name:n normalizes the file name (to factor out differences in the .tex extension), and then does a file lookup to take into account a possible path from \l_file_search_path_seq and \input@path. However only the file name and extension are returned so that file hooks can refer to the file by their name only. The path to the file is returned in \CurrentFilePath.

161	\edef\reserved@a{%
162	\@expl@@@filehook@file@pop@assign@@nnnn
163	{\CurrentFilePathUsed}%
164	{\CurrentFileUsed}%
165	{\CurrentFilePath}%
166	{\CurrentFile}}%
167	$\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $

Before adding to the file list we need to make all (letter) characters catcode 11, because several packages use constructions like

```
\filename@parse{<filename>}
\ifx\filename@ext\@clsextension
```

\fi

. . .

and that doesn't work if \filename@ext is \detokenized. Making \@clsextension a string doesn't help much because some packages define their own \<prefix>@someextension with normal catcodes. This is not entirely correct because packages loaded (somehow) with catcode 12 alphabetic tokens (say, as the result of a \string or \detokenize command, or from a TEX string like \jobname) will have these character tokens incorrectly turned into letter tokens. This however is rare, so we'll go for the all-letters approach (grepping the packages in TEX Live didn't bring up any obvious candidate for breaking with this catcode change).

```
168 {\edef\reserved@a{\unqu@tefilef@und#1\@nil}%
```

```
169 \@addtofilelist{\string@makeletter\reserved@a}%
```

```
170 \UseHook{file/before}%
```

The current file name is available in \CurrentFile so we use that in the specific hook.

```
171 \UseHook{file/\CurrentFile/before}%
```

```
\@@input #1% <- trailing space comes from \@filef@und</pre>
```

```
173 }%
```

And here, \CurrentFile is restored (by \@expl@@@filehook@file@pop@assign@@nnnn) so we can use it once more.

174 \UseHook{file/\CurrentFile/after}%

```
175 \UseHook{file/after}}
```

176 \def\unqu@tefilef@und"#1" \@nil{#1}

Now declare the non-generic file hooks used above:

- 177 \NewHook{file/before}
- 178 \NewReversedHook{file/after}
- 179 $(latexrelease) \ EndIncludeInRelease$
- $_{180}$ $\langle /2ekernel | latexrelease \rangle$

Now define \InputIfFileExists to input #1 if it seems to exist. Immediately prior to the input, #2 is executed. If the file #1 does not exist, execute '#3'.

```
181 (latexrelease)\IncludeInRelease{2019/10/01}%
```

```
182 (latexrelease) {\InputIfFileExists}{Hook management (files)}%
```

```
183 (latexrelease)
```

```
184 (latexrelease)\DeclareRobustCommand \InputIfFileExists[2]{%
```

```
185 (latexrelease) \IfFileExists{#1}%
```

```
186 (latexrelease) {%
```

- 187 (latexrelease) \expandafter\@swaptwoargs\expandafter
- 189 (latexrelease)\let\@input@file@exists@with@hooks\@undefined
- 190 (latexrelease)\let\unqu@tefilef@und\@undefined
- ¹⁹¹ $(latexrelease) \ EndIncludeInRelease$
- ¹⁹² (latexrelease)\IncludeInRelease{0000/00/00}%
- 193 (latexrelease) {\InputIfFileExists}{Hook management (files)}%
- 194 (latexrelease)\long\def \InputIfFileExists#1#2{%
- 195 $\langle | atexrelease \rangle \land IfFileExists{#1}%$
- $_{196}$ (latexrelease) {#2\@addtofilelist{#1}\@@input \@filef@und}}

Also undo the internal command as some packages unfortunately test for their existence instead of using \lfFormatAtLeastTF.

- $|197\rangle$ (latexrelease) expandafter let csname Input If FileExists endcsname Qundefined
- 198 (latexrelease)\let\@input@file@exists@with@hooks\@undefined
- 199 (latexrelease)\let\unqu@tefilef@und\@undefined
- 200 (latexrelease) \EndIncludeInRelease
- 201 (*2ekernel)

 $(End of definition for \liputIfFileExists, \@inputGfile@exists@with@hooks, and \unqu@tefile@und. These functions are documented on page \ref{eq:liputIfFileExists}).$

2.5 Declaring a file substitution

- 202 $\langle @@=filehook \rangle$
- (/2ekernel)
- 204 (*2ekernel | latexrelease)
- 205 (latexrelease)\IncludeInRelease{2020/10/01}%
- 206 (latexrelease) {__filehook_subst_add:nn}{Declaring file substitution}%
- 207 \ExplSyntaxOn

__filehook_subst_add:nn
__filehook_subst_remove:n
 __filehook_subst_file_normalize:Nn
 __filehook_subst_empty_name_chk:NW

 $_{filehook_subst_add:nn}$ declares a file substitution by doing a (global) definition of the form $def @file-subst@(file){(replacement)}$. The file names are properly sanitised, and normalized with the same treatment done for the file hooks. That is, a file replacement is declared by using the file name (and extension, if any) only, and the file path should not be given. If a file name is empty it is replaced by .tex (the empty csname is used to check that).

```
\cs_new_protected:Npn \__filehook_subst_add:nn #1 #2
208
     {
209
       \group_begin:
         \cs_set:cpx { } { \exp_not:o { \cs:w\cs_end: } }
         \int_set:Nn \tex_escapechar:D { -1 }
         \cs_gset:cpx
           {
214
             @file-subst@
             \__filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
216
           7
217
           { \__filehook_subst_file_normalize:Nn \__filehook_file_name_compose:nnn
218
                                           {#2} }
219
       \group_end:
220
    }
221
222 \cs_new_protected:Npn \__filehook_subst_remove:n #1
     ł
```

```
224
                                          \group_begin:
                                            \cs_set:cpx { } { \exp_not:o { \cs:w\cs_end: } }
                                  225
                                            \int_set:Nn \tex_escapechar:D { -1 }
                                  226
                                            \cs_undefine:c
                                              ł
                                  228
                                                 @file-subst@
                                  229
                                                   __filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
                                  230
                                              }
                                          \group_end:
                                       }
                                  233
                                     \cs_new:Npn \__filehook_subst_file_normalize:Nn #1 #2
                                  234
                                  235
                                       {
                                          \exp_after:wN \__filehook_subst_empty_name_chk:NN
                                  236
                                            \cs:w \exp_after:wN \cs_end:
                                              \cs:w \__filehook_file_parse_full_name:nN {#2} #1 \cs_end:
                                  238
                                       }
                                  239
                                     \cs_new:Npn \__filehook_subst_empty_name_chk:NN #1 #2
                                  240
                                       { \if_meaning:w #1 #2 .tex \else: \token_to_str:N #2 \fi: }
                                  241
                                  (End of definition for \__filehook_subst_add:nn and others.)
                                 A variant of \use_... to discard the first of three arguments.
              \use_ii_iii:nnn
                                        Todo: this should move to expl3
                                  242 \cs_gset:Npn \use_ii_iii:nnn #1 #2 #3 {#2 #3}
                                  (End of definition for \use_ii_iii:nnn.)
                                  243 \ExplSyntaxOff
                                     \langle /2ekernel | latexrelease \rangle
                                  244
                                  245 (latexrelease)\EndIncludeInRelease
                                  246 (*2ekernel)
                                 For two internals we provide \operatorname{IAT}_{FX} 2_{\varepsilon} names so that we can use them elsewhere in the
  \declare@file@substitution
                                 kernel (and so that they can be used in packages if really needed, e.g., scrlfile).
\undeclare@file@substitution
                                  247 (/2ekernel)
                                     {*2ekernel | latexrelease}
                                  248
                                  249 (latexrelease) \IncludeInRelease {2020/10/01}%
                                                             {\declare@file@substitution}{File substitution}%
                                  250 (latexrelease)
                                     \ExplSyntaxOn
                                  251
                                     \cs_new_eq:NN \declare@file@substitution
                                                                                      \__filehook_subst_add:nn
                                  252
                                  253 \cs_new_eq:NN \undeclare@file@substitution \__filehook_subst_remove:n
                                  254 \ExplSyntaxOff
                                  255 (/2ekernel | latexrelease)
```

```
256 (latexrelease) \EndIncludeInRelease
```

We are not fully rolling back the file substitutions in case a rollback encounters a package that contains them, but is itself not setup for rollback. So we just bypass them and hope for the best.

```
257 (latexrelease)\IncludeInRelease{0000/00/00}%
```

```
_{258} (latexrelease) {\declare@file@substitution}{File substitution}%
```

```
259 (latexrelease)
```

```
_{261} (latexrelease) \let \underlare@file@substitution \@gobble
```

 $_{262}$ (latexrelease)

²⁶³ (latexrelease) \EndIncludeInRelease

264 (***2ekernel**)

 $(End of definition for \declareOfileOsubstitution and \undeclareOfileOsubstitution. These functions are documented on page 6.)$

265 (QQ=)

2.6 Selecting a file (\set@curr@file)

\set@curr@file \set@curr@file@nosearch \@curr@file \@curr@file@reqd Now we hook into \set@curr@file to resolve a possible file substitution, and add \@expl@@@filehook@set@curr@file@@nNN at the end, after \@curr@file is set.

A file name is built using \expandafter\string\csname $\langle filename \rangle$ \endcsname to avoid expanding utf8 active characters. The \csname expands the normalization machinery and the routine to resolve a file substitution, returning a control sequence with the same name as the file.

It happens that when $\langle filename \rangle$ is empty, the generated control sequence is \csname\endcsname, and doing \string on that results in the file csnameendcsname.tex. To guard against that we \ifx-compare the generated control sequence with the empty csname. To do so, \csname\endcsname has to be defined, otherwise it would be equal to \relax and we would have false positives. Here we define \csname\endcsname to expand to itself to avoid it matching the definition of some other control sequence.

```
\langle /2ekernel \rangle
   (*2ekernel | latexrelease)
267
   (latexrelease) \IncludeInRelease {2022/06/01}%
268
   (latexrelease)
                          {\set@curr@file}{Setting current file name}%
269
   \def\set@curr@file{%
270
     \begingroup
271
       \set@curr@file@aux}
   \edef\set@curr@file@nosearch{%
     \begingroup
274
       \let\noexpand\input@path\noexpand\@empty
275
       \csname seq_clear:N\endcsname
276
          \expandafter\noexpand\csname l_file_search_path_seq\endcsname
       \noexpand\set@curr@file@aux}
278
   \def\set@curr@file@aux#1{%
279
        \escapechar\m@ne
280
       \let\protect\string
281
       \edef~{\string~}%
282
       \expandafter\def\csname\expandafter\endcsname
283
          \expandafter{\csname\endcsname}%
284
```

Two file names are set here: \@curr@file@reqd which is the file requested by the user, and \@curr@file which should be the same, except when we have a file substitution, in which case it holds the actual loaded file. \@curr@file is resolved first, to check if a substitution happens. If it doesn't, \@expl@@@filehook@if@file@replaced@@TF short-cuts and just copies \@curr@file, otherwise the full normalization procedure is executed.

At this stage the file name is parsed and normalized, but if the input doesn't have an extension, the default .tex is *not* added to \@curr@file because for applications other than \input (graphics, for example) the default extension may not be .tex. First check if the input has an extension, then if the input had no extension, call

\@expl@@@filehook@drop@extension@@N. In case of a file substitution, \@curr@file will have an extension.

```
\@expl@@@filehook@if@no@extension@@nTF{#1}%
285
         {\@tempswatrue}{\@tempswafalse}%
286
       \@kernel@make@file@csname\@curr@file
287
         \@expl@@@filehook@resolve@file@subst@@w {#1}%
288
       \@expl@@@filehook@if@file@replaced@@TF
289
         {\@kernel@make@file@csname\@curr@file@reqd
290
             \@expl@@@filehook@normalize@file@name@@w{#1}%
291
          \if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
292
         {\if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file \fi
293
          \global\let\@curr@file@reqd\@curr@file}%
       \@expl@@@filehook@clear@replacement@flag@@
     \endgroup}
    /2ekernel | latexrelease>
297
   (latexrelease)\EndIncludeInRelease
298
   (latexrelease)\IncludeInRelease{2021/06/01}%
299
   (latexrelease)
                          {\set@curr@file}{Setting current file name}%
300
   (latexrelease)\def\set@curr@file#1{%
301
  (latexrelease) \begingroup
303 (latexrelease)
                   \escapechar\m@ne
304 (latexrelease)
                   \let\protect\string
305 (latexrelease)
                   \edef~{\string~}%
306 (latexrelease)
                   \expandafter\def\csname\expandafter\endcsname
307 (latexrelease)
                     \expandafter{\csname\endcsname}%
308 (latexrelease)
                   \@expl@@@filehook@if@no@extension@@nTF{#1}%
309 (latexrelease)
                     {\@tempswatrue}{\@tempswafalse}%
310 (latexrelease)
                   \@kernel@make@file@csname\@curr@file
  (latexrelease)
                     \@expl@@@filehook@resolve@file@subst@@w {#1}%
311
312
   (latexrelease)
                   \@expl@@@filehook@if@file@replaced@@TF
313
   (latexrelease)
                     {\@kernel@make@file@csname\@curr@file@reqd
                         \@expl@@@filehook@normalize@file@name@@w{#1}%
   (latexrelease)
314
                       \if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
315
   (latexrelease)
                     {\if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file \fi
   (latexrelease)
316
                       \global\let\@curr@file@reqd\@curr@file}%
   (latexrelease)
                   \@expl@@@filehook@clear@replacement@flag@@
   (latexrelease)
318
   (latexrelease)
                 \endgroup}
319
   (latexrelease)\let\set@curr@file@nosearch\@undefined
320
   (latexrelease)\EndIncludeInRelease
321
322 (latexrelease)\IncludeInRelease{2020/10/01}%
323 (latexrelease)
                          {\set@curr@file}{Setting current file name}%
324 (latexrelease)\def\set@curr@file#1{%
325 (latexrelease)
                \begingroup
326 (latexrelease)
                   \escapechar\m@ne
327 (latexrelease)
                   \expandafter\def\csname\expandafter\endcsname
328 (latexrelease)
                     \expandafter{\csname\endcsname}%
329 (latexrelease)
                   \@expl@@@filehook@if@no@extension@@nTF{#1}%
330 (latexrelease)
                     {\@tempswatrue}{\@tempswafalse}%
   (latexrelease)
                   \@kernel@make@file@csname\@curr@file
331
  (latexrelease)
                     \@expl@@@filehook@resolve@file@subst@@w {#1}%
332
                   \@expl@@@filehook@if@file@replaced@@TF
   (latexrelease)
333
  (latexrelease)
                     {\@kernel@make@file@csname\@curr@file@reqd
334
                         \@expl@@@filehook@normalize@file@name@@w{#1}%
335 (latexrelease)
```

```
(latexrelease)
                       \if@tempswa \@expl@@dfilehook@drop@extension@@N\@curr@file@reqd \fi}%
336
   (latexrelease)
                      {\if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file \fi
337
   (latexrelease)
                       \global\let\@curr@file@regd\@curr@file}%
338
  (latexrelease)
                    \@expl@@@filehook@clear@replacement@flag@@
330
  (latexrelease)
                 \endgroup}
340
   (latexrelease)\let\set@curr@file@nosearch\@undefined
341
   (latexrelease)\EndIncludeInRelease
342
   (latexrelease)\IncludeInRelease{2019/10/01}%
343
                          {\set@curr@file}{Setting current file name}%
344 (latexrelease)
345 (latexrelease)\def\set@curr@file#1{%
  (latexrelease) \begingroup
346
   (latexrelease)
                   \escapechar\m@ne
347
   (latexrelease)
                   \xdef\@curr@file{%
348
   (latexrelease)
                      \expandafter\expandafter\unquote@name
349
350
   (latexrelease)
                      \expandafter\expandafter\expandafter{%
351
  (latexrelease)
                      \expandafter\string
  \langle \mathsf{latexrelease} \rangle
                        \csname\@firstofone#1\@empty\endcsname}}%
352
  (latexrelease)
                 \endgroup
353
  (latexrelease)}
354
  (latexrelease)\let\set@curr@file@nosearch\@undefined
355
  (latexrelease)\EndIncludeInRelease
356
357 (latexrelease)\IncludeInRelease{0000/00/00}%
358 (latexrelease)
                          {\set@curr@file}{Setting current file name}%
359 (latexrelease)\let\set@curr@file\@undefined
360 (latexrelease)\let\set@curr@file@nosearch\@undefined
361 (latexrelease)\EndIncludeInRelease
```

362 (*2ekernel)

(End of definition for \set@curr@file and others. These functions are documented on page ??.)

Todo: This should get internalized using <code>@expl0</code> names

\@filehook@set@CurrentFile \@kernel@make@file@csname \@set@curr@file@aux

```
363 {/2ekernel>
364 {*2ekernel|latexrelease>
365 {latexrelease}\IncludeInRelease{2020/10/01}%
366 {latexrelease} {\0kernel@make@file@csname}{Make file csname}%
367 \def\@kernel@make@file@csname#1#2#3{%
368 \xdef#1{\expandafter\@set@curr@file@aux
369 \csname\expandafter#2\@firstofone#3\@nil\endcsname}}
```

This auxiliary compares $\langle filename \rangle$ with $csname \endcsname$ to check if the empty .tex file was requested.

```
370 \long\def\@set@curr@file@aux#1{%
```

```
371 \expandafter\ifx\csname\endcsname#1%
```

```
.tex\else\string#1\fi}
```

Then we call \@expl@@@filehook@set@curr@file@@nNN once for \@curr@file to set \CurrentFile(Path)Used and once for \@curr@file@reqd to set \CurrentFile(Path). Here too the slower route is only used if a substitution happened, but here \@expl@@@filehook@if@file@replaced@@TF can't be used because the flag is reset at the \endgroup above, so we check if \@curr@file and \@curr@file@reqd differ. This macro is issued separate from \set@curr@file because it changes \CurrentFile, and side-effects would quickly get out of control.

```
373 \def\@filehook@set@CurrentFile{%
```

```
\@expl@@@filehook@set@curr@file@@nNN{\@curr@file}%
374
       \CurrentFileUsed\CurrentFilePathUsed
     \ifx\@curr@file@regd\@curr@file
376
       \let\CurrentFile\CurrentFileUsed
377
       \let\CurrentFilePath\CurrentFilePathUsed
378
     \else
379
       \@expl@@@filehook@set@curr@file@@nNN{\@curr@file@reqd}%
380
          \CurrentFile\CurrentFilePath
381
     fi
382
   \langle /2ekernel | latexrelease \rangle
383
   (latexrelease)\EndIncludeInRelease
384
```

385 (*2ekernel)

(End of definition for \@filehook@set@CurrentFile, \@kernel@make@file@csname, and \@set@curr@file@aux. These functions are documented on page ??.)

\@@_set_curr_file:nNN
\@@_set_curr_file_assign:mnNN
\@_set_curr_file_assign:mnNN
\U_file_search_path_seq) and returns the actual file name ((base) plus (ext))
in \CurrentFileUsed, and in case there's a file substitution, the requested file in
\CurrentFile (otherwise both are the same). Only the base and extension are returned,
regardless of the input (both path/to/file.tex and file.tex end up as file.tex in
\CurrentFile). The path is returned in \CurrentFilePath, in case it's needed.

```
386 (/2ekernel)
   (*2ekernel | latexrelease)
387
   (latexrelease) \IncludeInRelease{2020/10/01}%
388
  (latexrelease)
                           {@@_set_curr_file:nNN}{Set curr file}%
389
390 \ExplSyntaxOn
  \langle @@=filehook \rangle
391
392 \cs_new_protected:Npn \__filehook_set_curr_file:nNN #1
     {
393
        \exp_args:Nf \__filehook_file_parse_full_name:nN {#1}
394
          \__filehook_set_curr_file_assign:nnnNN
395
     }
396
   \cs_new_protected:Npn \__filehook_set_curr_file_assign:nnnNN #1 #2 #3 #4 #5
397
     ł
398
        \str_set:Nn #5 {#1}
399
       \str_set:Nn #4 {#2#3}
400
     }
401
402 \ExplSyntaxOff
403 (/2ekernel | latexrelease)
404 (latexrelease) \EndIncludeInRelease
405 (*2ekernel)
```

(End of definition for \@@_set_curr_file:nNN and \@@_set_curr_file_assign:nnnNN. These functions are documented on page ??.)

2.7 Replacing a file and detecting loops

Start by sanitizing the file with $_file_parse_full_name:nN$ then do $_file_subst_begin:nnn{\langle path \rangle}{\langle name \rangle}{\langle ext \rangle}.$

_filehook_resolve_file_subst:w
__filehook_normalize_file_name:w
__filehook_file_name_compose:nnn

- 406 (/2ekernel)
 407 (*2ekernel | latexrelease)
- 408 (latexrelease) \IncludeInRelease {2020/10/01}%
- 409 (latexrelease) {__filehook_resolve_file_subst:w}{Replace files detect loops}%

```
410 \ExplSyntaxOn
411 \cs_new:Npn \__filehook_resolve_file_subst:w #1 \@nil
412 { \__filehook_file_parse_full_name:nN {#1} \__filehook_file_subst_begin:nnn }
413 \cs_new:Npn \__filehook_normalize_file_name:w #1 \@nil
414 { \__filehook_file_parse_full_name:nN {#1} \__filehook_file_name_compose:nnn }
415 \cs_new:Npn \__filehook_file_name_compose:nnn #1 #2 #3
416 { \tl_if_empty:nF {#1} { #1 / } #2#3 }
```

```
__filehook_file_replaced
\__filehook_if_file_replaced:TF
\__filehook_clear_replacement_flag:
```

```
Since the file replacement is done expandably in a \csname, use a flag to remember if
a substitution happened. We use this in \set@curr@file to short-circuit some of it in
case no substitution happened (by far the most common case, so it's worth optimizing).
The flag raised during the file substitution algorithm must be explicitly cleared after the
\__filehook_if_file_replaced:TF conditional is no longer needed, otherwise further
uses of \__filehook_if_file_replaced:TF will wrongly return true.
417 \flag_new:n { __filehook_file_replaced }
```

```
418 \cs_new:Npn \__filehook_if_file_replaced:TF #1 #2
419 { \flag_if_raised:nTF { __filehook_file_replaced } {#1} {#2} }
420 \cs_new_protected:Npn \__filehook_clear_replacement_flag:
421 { \flag_clear:n { __filehook_file_replaced } }
```

__filehook_file_subst_begin:nnn

```
First off, start by checking if the current file (\langle name \rangle + \langle ext \rangle) has a declared substitution.
If not, then just put that as the name (including a possible \langle path \rangle in this case): this is the default case with no substitutions, so it's the first to be checked. The auxiliary \_filebook_file_subst_tortoise_hare:nn sees that there's no replacement for #2#3 and does nothing else.
```

```
422 \cs_new:Npn \__filehook_file_subst_begin:nnn #1 #2 #3
423 {
424 \__filehook_file_subst_tortoise_hare:nn { #2#3 } { #2#3 }
425 { \__filehook_file_name_compose:nnn {#1} {#2} {#3} }
426 }
427 \ExplSyntaxOff
428 {/2ekernel | latexrelease>
429 {latexrelease>\EndIncludeInRelease
430 {*2ekernel>
```

2.7.1 The Tortoise and Hare algorithm

If there is a substitution $(\langle true \rangle$ in the first $\cs_if_exist:cTF$ below), then first check if there is no substitution down the line: this should be the second most common case, of one file replaced by another. In that case just leave the substitution there and the job is done. If any substitution happens, then the $\flag __filehook_file_replaced$ is raised (conditionally, because checking if a flag is raised is much faster than raising it over and over again).

If, however there are more substitutions, then we need to check for a possible loop in the substitutions, which would otherwise put T_EX in an infinite loop if just an exhaustive expansion was used.

To detect a loop, the *Tortoise and Hare* algorithm is used. The name of the algorithm is an analogy to Aesop's fable, in which the Hare outruns a Tortoise. The two pointers here are the csnames which contains each file replacement, both of which start at the position zero, which is the file requested. In the inner part of the macro below, $_filehook_file_subst_loop:cc$ is called with $\langle file_subst_d(fi$

 $\file=subst@\@file=subst@\file}; that is, the substitution of <math>\langle file \rangle$ and the substitution of that substitution: the Tortoise walks one step while the Hare walks two.

Within __filehook_file_subst_loop:NN the two substitutions are compared, and if they lead to the same file it means that there is a loop in the substitutions. If there's no loop, __filehook_file_subst_tortoise_hare:nn is called again with the Tortoise at position 1 and the hare at 2. Again, the substitutions are checked ahead of the Hare pointer to check that it won't run too far; in case there is no loop in the declarations, eventually one of the \cs_if_exist:cTF below will go $\langle false \rangle$ and the algorithm will end; otherwise it will run until the Hare reaches the same spot as the tortoise and a loop is detected.

```
431 \langle /2ekernel \rangle
```

```
432
   (*2ekernel | latexrelease)
   (latexrelease) \IncludeInRelease{2020/10/01}%
433
                {\__filehook_file_subst_tortoise_hare:nn}{Tortoise and Hare}%
   (latexrelease)
434
  \ExplSyntaxOn
435
436
   \cs_new:Npn \__filehook_file_subst_tortoise_hare:nn #1 #2 #3
     ł
437
       \cs_if_exist:cTF { @file-subst@ #2 }
438
         ſ
439
           \flag_if_raised:nF { __filehook_file_replaced }
440
              { \flag_raise:n { __filehook_file_replaced } }
441
            \cs_if_exist:cTF { @file-subst@ \use:c { @file-subst@ #2 } }
442
              {
443
                \__filehook_file_subst_loop:cc
444
                  { @file-subst@ #1 }
445
                  { @file-subst@ \use:c { @file-subst@ #2 } }
446
              }
447
              {
                \use:c { @file-subst@ #2 } }
448
         }
449
         { #3 }
450
451
     }
```

This is just an auxiliary to check if a loop was found, and continue the algorithm otherwise. If a loop is found, the .tex file is used as fallback and __filehook_file_subst_-cycle_error:cN is called to report the error.

```
\cs_new:Npn \__filehook_file_subst_loop:NN #1 #2
452
453
     ł
       \token_if_eq_meaning:NNTF #1 #2
454
455
         ſ
456
            .tex
               _filehook_file_subst_cycle_error:cN {    @file-subst0 #1    } #1
457
         }
458
         Ł
              _filehook_file_subst_tortoise_hare:nn {#1} {#2} {#2} }
459
     }
460
  \cs_generate_variant:Nn \__filehook_file_subst_loop:NN { cc }
461
```

Showing this type of error expandably is tricky, as we have a very limited amount of characters to show and a potentially large list. As a work around, several errors are printed, each showing one step of the loop, until all the error messages combined show the loop.

```
462 \cs_new:Npn \__filehook_file_subst_cycle_error:NN #1 #2
463 {
464 \msg_expandable_error:nnff { latex2e } { file-cycle }
```

__filehook_file_subst_cycle_error:NN
__filehook_file_subst_cycle_error:cN

```
465 {#1} { \use:c { @file-subst@ #1 } }
466 \token_if_eq_meaning:NNF #1 #2
467 { \__filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #2 }
468 }
469 \cs_generate_variant:Nn \__filehook_file_subst_cycle_error:NN { c }
469 And the error message:
470 \msg_new:nnn { latex2e } { file-cycle }
471 { File~loop!~#1~replaced~by~#2... }
(End of definition for \__filehook_resolve_file_subst:w and others.)
```

```
472 \ExplSyntaxOff
```

```
473 (/2ekernel | latexrelease)
```

```
474 (latexrelease) \EndIncludeInRelease
```

```
475 \langle *2ekernel \rangle
```

476 **(00=**)

2.8 Preventing a package from loading

We support the use case of preventing a package from loading but not any other type of files (e.g., classes).

\disable@package@load \disable@package@load defines \@pkg-disable@\package\ to expand to some code #2 \reenable@package@load instead of loading the package.

477 (/2ekernel)

\@disable@packageload@do

- 478 (*2ekernel | latexrelease)
- 479 (latexrelease)\IncludeInRelease{2020/10/01}%
- 480 (latexrelease) {\disable@package@load}{Disable packages}%
- 481 \def\disable@package@load#1#2{%
- 482 \global\@namedef{@pkg-disable@#1.\@pkgextension}{#2}}

Here we check if a control sequence named \@pkg-disable@(name).sty is defined, and if so don't use the package loading code #2, but use the replacement code stored in that control sequence, write something to the log, and then prevent \@onefilewithoptions from sanity-checking the requested package date (the \expandafter here triggers one in \@onefilewithoptions that ends a conditional there, and the \@gobbletwo removes the date checking code from the input stream).

```
483 \def\@disable@packageload@do#1#2{%
```

- 484 \@ifundefined{@pkg-disable@#1}%
- 485 {#2}%
- 486 {\@nameuse{@pkg-disable@#1}%

```
487 \@latex@info{Package '#1' has been disabled.%
```

- 488 \MessageBreak Load request ignored}%
- 489 \expandafter\@gobbletwo}}

- 491 \global\expandafter\let
- 492 \csname @pkg-disable@#1.\@pkgextension \endcsname \@undefined}

```
(/2ekernel | latexrelease)
493
   (latexrelease)\EndIncludeInRelease
101
   (latexrelease)\IncludeInRelease{0000/00/00}%
105
   (latexrelease)
                           {\disable@package@load}{Disable packages}%
496
   (latexrelease)
497
   {latexrelease}\let\disable@package@load
                                                \@undefined
498
   (latexrelease)\let\@disable@packageload@do\@undefined
499
   {latexrelease}\let\reenable@package@load \@undefined
500
   (latexrelease)\EndIncludeInRelease
501
  (*2ekernel)
502
```

 $(End \ of \ definition \ for \ \ description \ \ description$

None so far and the general feeling for now is that the hooks are enough. Packages like filehook, etc., may use them to set up their interfaces (samples are given below) but for the now the kernel will not provide any.

2.10 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) $\text{LAT}_{\text{EX}} 2_{\varepsilon}$ names to allow for internal commands to be used outside this module (and in parts that still use $\text{LAT}_{\text{EX}} 2_{\varepsilon}$ syntax. We have to unset the QQ since we want double "at" sign in place of double underscores.

```
(@@=)
503
  \langle /2ekernel \rangle
504
   (*2ekernel | latexrelease)
505
   (latexrelease) \IncludeInRelease {2020/10/01}%
   (latexrelease)
                   {\@expl@@@filehook@if@no@extension@@nTF}{2e tmp interfaces}%
507
  \ExplSyntaxOn
508
   \cs_new_eq:NN \@expl@@@filehook@if@no@extension@@nTF
509
                  \__filehook_if_no_extension:nTF
   \cs_new_eq:NN \@expl@@@filehook@set@curr@file@@nNN
511
                  \__filehook_set_curr_file:nNN
  \cs_new_eq:NN \@expl@@@filehook@resolve@file@subst@@w
513
                  \__filehook_resolve_file_subst:w
514
   \cs_new_eq:NN \@expl@@@filehook@normalize@file@name@@w
                  \__filehook_normalize_file_name:w
516
  \cs_new_eq:NN \@expl@@@filehook@if@file@replaced@@TF
                  \__filehook_if_file_replaced:TF
518
  \cs_new_eq:NN \@expl@@@filehook@clear@replacement@flag@@
519
                  \__filehook_clear_replacement_flag:
  \cs_new_eq:NN \@expl@@@filehook@drop@extension@@N
521
                  \__filehook_drop_extension:N
  \cs_new_eq:NN \@expl@@@filehook@file@push@@
523
                  \__filehook_file_push:
524
  \cs_new_eq:NN \@expl@@@filehook@file@pop@@
525
                  \__filehook_file_pop:
526
```

527 \cs_new_eq:NN \@expl@@@filehook@file@pop@assign@@nnnn

```
528 \__filehook_file_pop_assign:nnnn
```

529 \ExplSyntaxOff

This one specifically has to be undefined because it is left over in the input stream from \InputIfFileExists and executed when latexrelease is loaded. It cannot be \let to \@undefined otherwise it would error as well, so it is \let to \relax to be silently ignored when loading \latexrelease.

530 2ekerne</th <th> latexrelease ></th>	latexrelease >
--	----------------

- $_{531}$ (latexrelease) LendIncludeInRelease
- 532 $\langle |atexrelease \rangle$
- $_{534}$ (latexrelease) {\@expl@@filehook@if@no@extension@@nTF}{2e tmp interfaces}%
- 535 (latexrelease)\let\@expl@@@filehook@file@pop@@\relax
- 536 $(latexrelease) \ EndIncludeInRelease$
- 537 $\langle *2ekernel \rangle$

This ends the kernel code in this file.

538 $\langle /2ekernel \rangle$

3 A sample package for structuring the log output

```
539 (*structuredlog)
```

540 $\langle @@=filehook \rangle$

```
541 \ProvidesExplPackage
```

```
542 {structuredlog}{\ltfilehookdate}{\ltfilehookversion}
```

543 {Structuring the TeX transcript file}

```
\g_filehook_nesting_level_int Stores the current package nesting level.
```

```
544 int_new: N \g_filehook_nesting_level_int
```

Initialise the counter with the number of files in the \@currnamestack (the number of items divided by 3) minus one, because this package is skipped when printing to the log.

```
545 \int_gset:Nn \g__filehook_nesting_level_int
546 { ( \tl count:N \@currnamestack ) / 3 - 1 }
```

 $_{546}$ { (\tl_count:N \@currnamestack) / 3 - 1 }

(End of definition for \g__filehook_nesting_level_int.)

```
\__filehook_log_file_record:n This macro is responsible for increasing and decreasing the file nesting level, as well as printing to the log. The argument is either STOPTART or STOP and the action it takes on the nesting integer depends on that.
```

```
547 \cs_new_protected:Npn \__filehook_log_file_record:n #1
548 {
549 \str_if_eq:nnT {#1} {START} { \int_gincr:N \g__filehook_nesting_level_int }
550 \iow_term:x
551 {
552 \prg_replicate:nn { \g__filehook_nesting_level_int } { = } ~
553 (LEVEL ~ \int_use:N \g__filehook_nesting_level_int \c_space_tl #1 ) ~
554 \CurrentFileUsed
```

If there was a file replacement, show that as well:

```
555 \str_if_eq:NNF \CurrentFileUsed \CurrentFile
556 { ~ (\CurrentFile \c_space_tl requested ) }
557 \iow_newline:
558 }
559 \str_if_eq:nnT {#1} {STOP} { \int_gdecr:N \g_filehook_nesting_level_int }
560 }
```

Now just hook the macro above in the generic file/before...

561 \AddToHook{file/before}{ __filehook_log_file_record:n { START } }

... and file/after hooks. We don't want to install the file/after hook immediately, because that would mean it is the first time executed when the package finishes. We therefore put the declaration inside \AddToHookNext so that it gets only installed when we have left this package.

```
562 \AddToHookNext{file/after}
```

```
563 { \AddToHook{file/after}{ \__filehook_log_file_record:n { STOP } } }
```

(End of definition for __filehook_log_file_record:n.)

564 〈@@=〉
565 〈/structuredlog〉

4 Package emulations

4.1 Package atveryend emulation

With the new hook management and the hooks in **\enddocument** all of **atveryend** is taken care of. We can make an emulation only here after the substitution functionality is available:

```
566 \langle *2ekernel \rangle
```

568 $\langle /2ekernel \rangle$

572

Here is the package file we point to:

- $_{\rm 569}~\langle *atveryend-ltx \rangle$
- 570 \ProvidesPackage{atveryend-ltx}
- 571 [2020/08/19 v1.0a
 - Emulation of the original atveryend package ``J with kernel methods]

Here are new definitions for its interfaces now pointing to the hooks in \enddocument

```
573 \newcommand\AfterLastShipout {\AddToHook{enddocument/afterlastpage}}
```

574 \newcommand\AtVeryEndDocument {\AddToHook{enddocument/afteraux}}

Next one is a bit of a fake, but the result should normally be as expected. If not, one needs to add a rule to sort the code chunks in enddocument/info.

```
575 \newcommand\AtEndAfterFileList{\AddToHook{enddocument/info}}
```

```
576 \newcommand\AtVeryVeryEnd {\AddToHook{enddocument/end}}
```

\BeforeClearDocument This one is the only one we don't implement or rather don't have a dedicated hook in the code.

```
577 \ExplSyntaxOn
```

```
578 \newcommand\BeforeClearDocument[1]
```

```
579 { \AtEndDocument{#1}
```

```
580 \atveryend@DEPRECATED{BeforeClearDocument \tl_to_str:n{#1}}
```

581 }

```
582 \cs_new:Npn\atveryend@DEPRECATED #1
583 {\iow_term:x{=====~DEPRECATED~USAGE~#1~=====}}
584 \ExplSyntaxOff
```

 $(\mathit{End of definition for \verb+BeforeClearDocument}. \ \mathit{This function is documented on page ??.})$

585 $\langle /atveryend-Itx \rangle$

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	$cs_new_protected:Npn \dots 49,$
@@ commands:	62, 70, 79, 208, 222, 392, 397, 420, 547
\@@_set_curr_file:nNN <u>386</u>	$cs_set:Npx \dots 211, 225$
$\column{0}{0}\co$	$cs_undefine: N \dots 227$
$\langle filename \rangle$ 17	\csname 197, 276, 277, 283, 284, 306,
	$307, \ 327, \ 328, \ 352, \ 369, \ 371, \ 492, \ 15$
\mathbf{A}	\csname\endcsname 15
\AddToHook . 561, 563, 573, 574, 575, 576, 5	$CurrentFile \dots 67, 84,$
\AddToHookNext 562, 24	$166, 171, 174, 377, 381, 555, 556, \underline{3}, 3$
\AfterLastShipout 573	\CurrentFilePath $67, 83, 165, 378, 381, \underline{3}, 9$
\AtEndAfterFileList 575	\CurrentFilePathUsed
\AtEndDocument 579	$\dots \dots \dots \dots \dots 66, 81, 163, 375, 378, \underline{3}, 3$
\AtEndOfClass 4	\CurrentFileUsed
\AtEndOfPackage 4	. 66, 82, 164, 375, 377, 554, 555, $\underline{3}$, 11
\AtVeryEndDocument 574	7
\AtVeryVeryEnd 576	D
_	\DeclareRobustCommand 148, 184
В	\def 160, 176,
\BeforeClearDocument 577	194, 270, 279, 283, 301, 306, 324,
1000000000000000000000000000000000000	297 245 267 270 272 481 482 400
\begingroup 271, 274, 302, 325, 346	327, 345, 367, 370, 373, 481, 483, 490
	\detokenize 12
С	\detokenize 12
C class//after 3	\detokenize 12 E
C class//after 3 class//before 3	\detokenize 12 E \edef 161, 168, 273, 282, 305
C class//after	\detokenize 12 E \edef 161, 168, 273, 282, 305 \else 372, 379
C class//after	\detokenize 12 E \edef 161, 168, 273, 282, 305 \else 372, 379 else commands:
C class//after	\detokenize 12 E \edef 161, 168, 273, 282, 305 \else 372, 379 else commands: \else: 241
C class//after	\detokenize 12 E \edef 161, 168, 273, 282, 305 \else 372, 379 else commands: 241 \endcsname 197, 276, 277, 283, 284, 306,
C class//after	\detokenize 12 E 161, 168, 273, 282, 305 \else 372, 379 else commands: 241 \endcsname 197, 276, 277, 283, 284, 306, 307, 327, 328, 352, 369, 371, 492, 15
C class//after	Letokenize 12 E Ledef 161, 168, 273, 282, 305 Lelse 372, 379 else commands: 241 Lendcsname 197, 276, 277, 283, 284, 306, 307, 327, 328, 352, 369, 371, 492, 15 Lenddocument 24
C class//after	Letokenize 12 E 161, 168, 273, 282, 305 Lelse 372, 379 else commands: 241 Lendcsname 197, 276, 277, 283, 284, 306, 307, 327, 328, 352, 369, 371, 492, 15 Lenddocument 24 Lendgroup 296, 319, 340, 353, 17
C class//after	\detokenize 12 E 161, 168, 273, 282, 305 \else 372, 379 else commands: 241 \endcsname 197, 276, 277, 283, 284, 306, 307, 327, 328, 352, 369, 371, 492, 15 \endgroup 24 \endgroup 296, 319, 340, 353, 17 \endIncludeInRelease 23,
C class//after	Ketokenize 12 E \edef 161, 168, 273, 282, 305 \else 372, 379 else commands: 372, 379 else: 241 \endcsname 197, 276, 277, 283, 284, 306, 307, 327, 328, 352, 369, 371, 492, 15 \enddocument 24 \endgroup 296, 319, 340, 353, 17 \EndIncludeInRelease 23, 88, 140, 179, 191, 200, 245, 256,
C class//after	Ketokenize 12 E \edef 161, 168, 273, 282, 305 \else 372, 379 else commands: 372, 379 else: 241 \endcsname 197, 276, 277, 283, 284, 306, 307, 327, 328, 352, 369, 371, 492, 15 \endgroup 296, 319, 340, 353, 17 \endgroup 296, 319, 340, 353, 17 \endIncludeInRelease 23, 88, 140, 179, 191, 200, 245, 256, 263, 298, 321, 342, 356, 361, 384,
C class//after	Letokenize 12 E 161, 168, 273, 282, 305 \else 161, 168, 273, 282, 305 \else 372, 379 else commands: 241 \else: 241 \endcsname 197, 276, 277, 283, 284, 306, 307, 327, 328, 352, 369, 371, 492, 15 \endgroup 296, 319, 340, 353, 17 \endgroup 298, 321, 342, 356, 361, 384, 404, 429, 474, 494, 501, 531, 536, 14
C class//after	Letokenize 12 E \edef 161, 168, 273, 282, 305 \else 372, 379 else commands: 241 \else: 241 \endcsname 197, 276, 277, 283, 284, 306, 307, 327, 328, 352, 369, 371, 492, 15 \endgroup 296, 319, 340, 353, 17 \endgroup 296, 319, 340, 354, 256, 263, 298, 321, 342, 356, 361, 384, 404, 429, 474, 494, 501, 531, 536, 14 \escapechar 280, 303, 326, 347
C class//after	Letokenize 12 E \edef 161, 168, 273, 282, 305 \else 372, 379 else commands: 372, 379 else commands: 241 \endcsname 197, 276, 277, 283, 284, 306, 307, 327, 328, 352, 369, 371, 492, 15 \endgroup 296, 319, 340, 353, 17 \endgroup 296, 319, 342, 356, 361, 384, 404, 429, 474, 494, 501, 531, 536, 14 \escapechar 280, 303, 326, 347 exp commands: 280, 303, 326, 347
C class//after	Letokenize 12 E \edef 161, 168, 273, 282, 305 \else 372, 379 else commands: 372, 379 else: 241 \endcsname 197, 276, 277, 283, 284, 306, 307, 327, 328, 352, 369, 371, 492, 15 \endgroup 296, 319, 340, 353, 17 \endgroup 296, 319, 340, 354, 256, 263, 298, 321, 342, 356, 361, 384, 404, 429, 474, 494, 501, 531, 536, 14 \escapechar 280, 303, 326, 347

\exp_args:Nee 129 \exp_args:Nf 32, 34, 394
\exp_args:NV 53
\exp_not:n 211, 225
\expandafter
. 153, 154, 155, 167, 187, 197, 277,
283, 284, 306, 307, 327, 328, 349,
350, 351, 368, 369, 371, 489, 491, 15
\ExplSyntaxOff 86, 138,
243, 254, 402, 427, 472, 529, 584, 12
$\Sigma 29, 92, \dots$
207, 251, 390, 410, 435, 508, 577, 7

\mathbf{F}

\fi 292, 293, 315, 316, 336, 337, 372, 382
fi commands:
\fi: 241
file commands:
\file_full_name:n 35, 11
\file_parse_full_name_apply:nN
$\dots \dots $
$l_file_search_path_seq$
file internal commands:
$_file_parse_full_name_area:nw$.
$\dots \dots $
<pre>\file_parse_full_name_auxi:nN .</pre>
$\dots \dots $
\file_parse_full_name_base:nw .
108, 112, 124
<pre>\file_parse_full_name_tidy:nnnN</pre>
119, 120, 122, 127
file//after 2
file//before 2
file/after 2
file/before 2
filehook internal commands:
\filehook_clear_replacement
flag: <u>417</u> , 420, 520
\filehook_drop_extension:N
$\underbrace{44}, 49, 522$
\filehook_drop_extension
aux:nnn 54, 57
\filehook_file_name_compose:nnn
\filehook_file_parse_full
name:nN 28,
<u>30</u> , 30, 53, 238, 394, 412, 414, 8
\filehook_file_pop: <u>59</u> , 70, 526
\filehook_file_pop_assign:nnnn
<u></u>
\filehook_file_push: <u>59</u> , 62, 524
filehook_file_replaced <u>417</u>
\filehook_file_subst_begin:nnn
······································

\filehook_file_subst_cycle
error:NN 457, <u>462</u> , 462, 467, 469, <i>20</i>
\filehook_file_subst_loop:NN
$\dots \dots \dots \underline{431}, 444, 452, 461, 20$
\filehook_file_subst_tortoise
hare:nn . $424, \underline{431}, 434, 436, 459, 20$
$_$ filehook_full_name:nn $30, 34, 38$
<pre>\filehook_if_file_replaced:TF .</pre>
$\dots \dots \underline{417}, 418, 518, 19$
$_$ _filehook_if_no_extension:nTF .
$\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \underbrace{44}, 44, 510$
\g_filehook_input_file_seq $\dots 59, 9$
$l_filehook_internal_tl \dots 59$
\filehook_log_file_record:n
$\dots \dots $
$g_filehook_nesting_level_int$.
$\dots \dots \dots \dots \dots \underline{544}, 549, 552, 553, 559$
\filehook_normalize_file
name:w $406, 413, 516$
<pre>\filehook_resolve_file_subst:w</pre>
$\dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \underline{406}, 409, 411, 514$
\filehook_set_curr_file:nNN
392, 512
\filehook_set_curr_file
$assign:nnnNN \dots 395, 397$
\filehook_subst_add:nn
$\dots \dots $
\filehook_subst_empty_name
chk:NN $208, 236, 240$
\filehook_subst_file_normalize:Nn
$\dots \dots \dots \dots \dots \dots \underline{208}, 216, 218, 230, 234$
\filehook_subst_remove:n
208, 222, 253
flag internal commands:
$\lag_{\sqcup}_filehook_file_replaced$ 19
flag commands:
$flag_clear:n$ 421
\flag_if_raised:nTF 419, 440
\flag_new:n 417
\flag_raise:n 441
<i></i>
G
\global 294, 317, 338, 482, 491
group commands:
\group_begin: 210, 224
\group_end: 220, 232

Ι

\if_meaning:w 241 \IfFileExists 149, 185, 195

26

if commands:

include//after
include//before 5
include//end
include/after 5
include/before 5
include/end 5
\IncludeInRelease 27,
89, 144, 181, 192, 205, 249, 257,
268, 299, 322, 343, 357, 365, 388,
408, 433, 479, 495, 506, 533, 5, 15
\includeonly 4
\input
\InputIfFileExists <u>143</u> , <i>11</i>
int commands:
\int_gdecr:N 559
\int_gincr:N 549
\int_gset:Nn 545
\int_new:N 544
\int_set:Nn 212, 226
\int_use:N 553
iow commands:
\iow_newline:
\iow_term:n 550, 583
J
\jobname 12
(Jobname
К
11
kernel internal commands.
kernel internal commands:
<pre>kernel internal commands: \kernel_file_name_sanitize:n 98</pre>
\kernel_file_name_sanitize:n 98 L
\kernel_file_name_sanitize:n 98 L \latexrelease 23
\kernel_file_name_sanitize:n 98 L \latexrelease 23 \let 18,
L \latexrelease
L Latexrelease
L L L L L L L L L L L L L L L L L L L
<pre> L L L L L L L L L L L L L L L L L L L</pre>
<pre> L L L L L L L L L L L L L L L L L L L</pre>
<pre> L L L L L L L L L L L L L L L L L L L</pre>
L Latexrelease
<pre> L L L L L L L L L L L L L L L L L L L</pre>
<pre> L L L L L L L L L L L L L L L L L L L</pre>
L Latexrelease
L L latexrelease
L L latexrelease
<pre> L L L Latexrelease</pre>
L L L L L L L L L L L L L L L L L L L
<pre> L L L Latexrelease</pre>
<pre>\kernel_file_name_sanitize:n 98 L L \latexrelease</pre>
<pre>\kernel_file_name_sanitize:n 98 L L \latexrelease</pre>
<pre> L L Latexrelease</pre>
<pre>\kernel_file_name_sanitize:n 98 L \latexrelease</pre>
<pre> L L Latexrelease</pre>

0				
\openin 2				
Р				
package//after 3				
package//before 3				
package/after 3				
package/before 3				
prg commands:				
\prg_do_nothing: 132, 134				
\prg_replicate:nn 552				
$\verb protect 281, 304 $				
$\verb ProvidesExplPackage$				
$\verb ProvidesPackage$				
-				
\mathbf{R}				
\relax 535, 15				
\RequirePackage 3				
\mathbf{S}				
scan internal commands:				
\sfile_stop				

\sfile_stop
$\dots \dots $
seq commands:
$seq_gpop:NNTF \dots 72$
$seq_gpush: Nn \dots 64$
\seq_if_exist:NTF 60
$seq_new: \mathbb{N} \dots \dots$
str commands:
\str_if_eq:NNTF 555
\str_if_eq:nnTF 131, 549, 559
\str_set:Nn 399, 400
\string 281, 282, 304, 305, 351, 372, 15

Т			
T_EX and $E^T_EX 2_{\varepsilon}$ commands:			
\@@input 172, 188, 196			
$\ensuremath{\texttt{0addtofilelist}}\ \ldots \ 169, \ 188, \ 196$			
\@clsextension 12			
\@curr@file <u>266</u> , 374, 376, <i>16</i>			
\@curr@file@reqd <u>266</u> , 376, 380, <i>15</i>			
\@currnamestack $\dots \dots \dots 546, 23$			
\@disable@packageload@do $\dots \dots \frac{477}{2}$			
\@empty 275, 352			
\@expl@@@filehook@clear@replacement@flag@@			
$\dots \dots \dots \dots \dots \dots \dots \dots \dots \dots 295, 318, 339, 519$			
\@expl@@@filehook@drop@extension@@N			
292, 293, 315, 316, 336, 337, 521, 16			
\@expl@@@filehook@file@pop@@			
$\dots \dots $			
\@expl@@@filehook@file@pop@assign@@nnnn			
$\dots \dots $			
\@expl@@@filehook@file@push@@			
$\dots \dots $			

\@expl@@@filehook@if@file@replaced@@TF				
\@expl@@@filehook@if@no@extension@@nTF				
\CexplCCCfilehookCnormalizeCfileCnameCCw				
\@expl@@@filehook@resolve@file@subst@@w				
\@expl@@@filehook@set@curr@file@@nNN				
$\langle file-subst Q(file) \dots 13$				
\@filef@und 155, 172, 188, 196, 11				
\@filehook@set@CurrentFile . 152, <u>363</u>				
\@firstofone 352, 369 tex				
\@gobble				
\@gobbletwo 260, 489, 21 tl c				
\@ifundefined 484				
\@input 2				
\@input@file@exists@with@hooks . <u>143</u>				
\@kernel@make@file@csname				
\dots 287, 290, 310, 313, 331, 334, <u>363</u>				
\mathbb{Q}				
\@namedef 482				
\@nameuse 486				
$\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $				
\@onefilewithoptions				
\@pkgextension 482, 492				
\@set@curr@file@aux <u>363</u>				
\@swaptwoargs $\dots \dots \dots 153, 167, 187$				
\@tempswafalse 286, 309, 330 use				
\@tempswatrue 286, 309, 330				
\Cundefined 18, 19, 20, 21, 147,				
189, 190, 197, 198, 199, 320, 341,				
355, 359, 360, 492, 498, 499, 500, 23				
$\tatveryend@DEPRECATED$ 580, 582 \type{U}_{2}				
\declare@file@substitution \us				
$\dots \dots \underline{247}, 567, 6$				
\disable@package@load $\dots \dots \frac{477}{6}$				
\expandafter 21 \xeta				

	\filename@ext 12			
	\if@tempswa 292, 293, 315, 316, 336, 337			
	\input@path 275, 3			
	\m@ne 280, 303, 326, 347			
Qw	\reenable@package@load 477, 6			
	\reserved@a 161, 167, 168, 169			
W	\set@curr@file <u>266</u> , <u>18</u>			
	\set@curr@file@aux 272, 278, 279			
	\set@curr@file@nosearch 266			
	\string@makeletter 169			
	$\cline{247}, 6$			
	$\unqu@tefilef@und \dots 143$			
	\unquote@name 349			
tex	commands:			
	\tex_escapechar:D 212, 226			
tl c	ommands:			
	\c_space_tl 553, 556			
	\tl_count:N 546			
	$tl_gset:Nn \dots 51$			
	\tl_if_empty:nTF			
	\ldots 40, 46, 58, 107, 114, 116, 118, 416			
	\tl_new:N 59, 8, 9, 10, 11			
	\tl_set:Nn 81, 82, 83, 84			
	\tl_to_str:n 580			
	\tl_trim_spaces:n 41, 42			
tok	en commands:			
	$\verb+token_if_eq_meaning:NNTF \ . \ . \ 454, \ 466$			
	$\verb+token_to_str:N \dots 241$			
U				

\@tempswafalse 286, 309, 330	use commands:
\@tempswatrue 286, 309, 330	\use:N 442, 446, 448, 465
\@undefined $18, 19, 20, 21, 147,$	\use_ii_iii:nnn 216, 230, <u>242</u> , 242
189, 190, 197, 198, 199, 320, 341,	\use_iii:nnn 47
355, 359, 360, 492, 498, 499, 500, 23	\use_none:n 131, 134
\atveryend@DEPRECATED 580, 582	\UseHook 170, 171, 174, 175, 2
\declare@file@substitution	\usepackage 2
247, 567, 6	
$\begin{aligned} \label{eq:load} \begin{aligned} align$	X
\expandafter 21	\xdef 348, 368