

The filehook Package

Martin Scharrer

martin@scharrer-online.de

CTAN: <http://www.ctan.org/pkg/filehook>

Version v0.6 – 2019/10/03

Abstract

This package provides hooks for input files. Document and package authors can use these hooks to execute code at begin or the end of specific or all input files.

1 Introduction

These package changes some internal \LaTeX macros used to load input files so that they include ‘hooks’. A hook is an (internal) macro executed at specific points. Normally it is initially empty, but can be extended using an user level macro. The most common hook in \LaTeX is the ‘At-Begin-Document’ hook. Code can be added to this hook using `\AtBeginDocument{ \langle TeX code \rangle }`.

This package provides hooks for files read by the \LaTeX macros `\input`, `\include` and `\InputIfFileExists` as well as (since v0.3 from 2010/12/20) for class and package files, i.e. macros `\documentclass`, `\LoadClassWithOptions` and `\LoadClass` as well as `\usepackage`, `\RequirePackageWithOptions` and `\RequirePackage`. Note that `\InputIfFileExists`, and therefore its hooks, is used by the aforementioned macros. In v0.4 from 2011/03/01 special hooks were added which are executed for every read file, but will not be executed a second time by the internal `\InputIfFileExists` inside `\input` and `\include`.

For all files a ‘AtBegin’ and a ‘AtEnd’ hook is installed. For `\include` files there is also a ‘After’ hook which is executed *after* the page break (`\clearpage`) is inserted by the `\include` code. In contrast, the ‘AtEnd’ hook is executed before the trailing page break and the ‘AtBegin’ hook is executed after the *leading* page break. The ‘AtBegin’ hook can be used to set macros to file specific values. These macros can be reset in the ‘AtEnd’ hook to the parent file values. If these macros appear in the page header or footer they need to be reset ‘After’ hook to ensure that the correct values are used for the last page.

In addition to general hooks which are executed for all files of there type, file specific one can be defined which are only executed for the named file. The hooks for classes and packages are always specific to one file.

Older versions of this package provided the file name as argument #1 for the general hooks. This has been changed in v0.4 from 2011/01/03: the hook code is stored and executed without modifications, i.e. macro argument characters (#) are

now handled like normal and don't have to be doubled. See section 5 for information how to upgrade older documents.

2 Usage

The below macros can be used to add material (TeX code) to the related hooks. All 'AtBegin' macros will *append* the code to the hooks, but the 'AtEnd' and 'After' macros will *prefix* the code instead. This ensures that two different packages adding material in 'AtBegin'/'AtEnd' pairs do not overlap each other. Instead the later used package adds the code closer to the file content, 'inside' the material added by the first package. Therefore it is safely possible to surround the content of a file with multiple L^AT_EX environments using multiple 'AtBegin'/'AtEnd' macro calls. If required inside another package a different order can be enforced by using the internal hook macros shown in the implementation section.

Every File

<pre>\AtBeginOfEveryFile{<TeX code>} \AtEndOfEveryFile{<TeX code>}</pre>
--

Sometime certain code should be executed at the begin and end of every read file, e.g. pushing and popping a file stack. The 'At...OfFiles' hooks already do a good job here. Unfortunately there is the issue with the `\clearpage` in `\include`. The `\AtEndOfFiles` is executed before it, which can cause issues with page headers and footers. A workaround, e.g. done by older versions of the `currfile` package, is to execute the code twice for include files: once in the `include` related hooks and once in the `OfFiles` hooks.

A better solution for this problem was added in v0.4 from 2011/01/03: the `EveryFile` hooks will be executed exactly once for every file, independent if it is read using `\input`, `\include` or `\InputIfFileExists`. Special care is taken to suppress them for the `\InputIfFileExists` inside `\input` and `\include`.

These hooks are located around the more specific hooks: For `\input` files the 'Begin' hook is executed before the `\AtBeginOfInputs` hook and the 'End' hook after the `\AtEndOfInputs`. Similarly, for `\include` files the 'Begin' hook is executed before the `\AtBeginOfIncludes` hook and the 'End' hook after the `\AfterIncludes` (!). For files read by `\InputIfFileExists` (e.g. also for `\usepackage`, etc.) they are executed before and after the `\AtBeginOfFiles` and `\AtEndOfFiles` hooks, respectively. Note that the `\AtBeginOfEveryFile` hook is executed before the `\AtBeginOfPackageFile`/`\AtBeginOfClassFile` hooks and that the `\AtEndOfEveryFile` hook is executed also before the hooks `\AtEndOfPackageFile`/`\AtEndOfClassFile`. Therefore the 'Every' and 'PackageFile'/'ClassFile' hooks do not nest correctly like all other hooks do.

All Files

```
\AtBeginOfFiles{<TEX code>}
\AtEndOfFiles{<TEX code>}
```

These macros add the given $\langle code \rangle$ to two hooks executed for all files read using the `\InputIfFileExists` macro. This macro is used internally by the `\input`, `\include` and `\usepackage/\RequirePackage` macros. Packages and classes might use it to include additional or auxiliary files. Authors can exclude those files from the hooks by using the following code instead:

```
\IfFileExists{<file name>}{\@input\@filef@und}{}{}
```

```
\AtBeginOfFile{<file name>}{<TEX code>}
\AtEndOfFile{<file name>}{<TEX code>}
```

Like the `\...OfIncludeFile{<file name>}{<TEX code>}` macros above, just for ‘all’ read files. If the $\langle file name \rangle$ does not include a file extension it will be set to ‘.tex’.

The ‘all files’ hooks are closer to the file content than the `\input` and `\include` hook, i.e. the `\AtBeginOfFiles` comes *after* the `\AtBeginOfIncludes` and the `\AtEndOfFiles` comes *before* the `\AtEndOfIncludes` hook.

The following figure shows the positions of the hooks inside the macro:

```
\InputIfFileExists:
Hook: AtBeginOfEveryFile
Hook: AtBeginOfFile{<file name>}
Hook: AtBeginOfFiles
Content
Hook: AtEndOfFiles
Hook: AtEndOfFile{<file name>}
Hook: AtEndOfEveryFile
```

Include Files

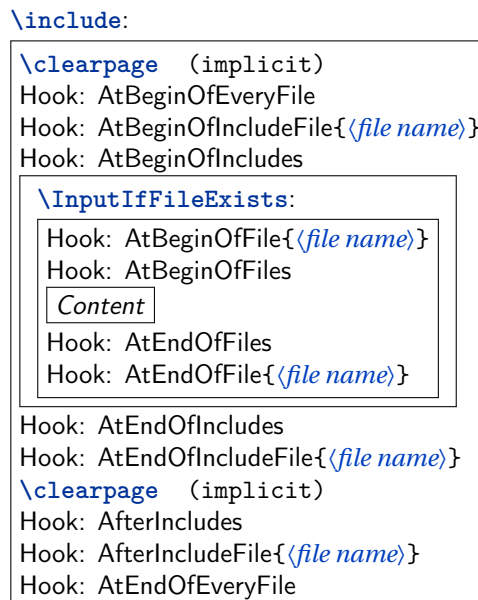
```
\AtBeginOfIncludes{<TEX code>}
\AtEndOfIncludes{<TEX code>}
\AfterIncludes{<TEX code>}
```

As described above the ‘AtEnd’ hook is executed before and the ‘After’ hook is executed after the trailing `\clearpage`. Note that material which appears in the page header or footer should be updated in the ‘After’ hook, not the ‘AtEnd’ hook, to ensure that the old values are still valid for the last page.

```
\AtBeginOfIncludeFile{<file name>}{<TEX code>}
\AtEndOfIncludeFile{<file name>}{<TEX code>}
\AfterIncludeFile{<file name>}{<TEX code>}
```

These file-specific macros take the two arguments. The $\langle code \rangle$ is only executed for the file with the given $\langle file name \rangle$ and only if it is read using `\include`. The $\langle file name \rangle$ should be identical to the name used for `\include` and not include the ‘.tex’ extension. Files with a different extension are neither supported by `\include` nor this hooks.

The following figure shows the positions of the hooks inside the macro:



Input Files

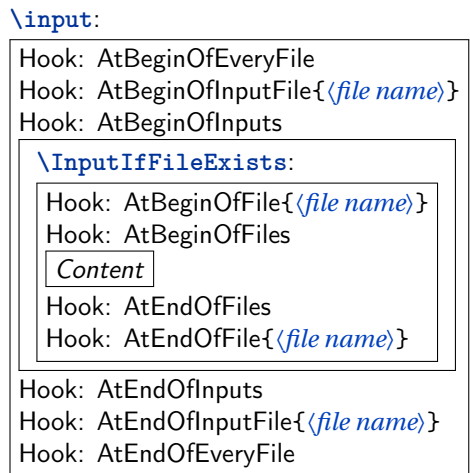
`\AtBeginOfInputs{<TeX code>}`
`\AtEndOfInputs{<TeX code>}`

Like the `\...OfIncludes{code}` macros above, just for file read using `\input`.

`\AtBeginOfInputFile{<file name>}{<TeX code>}`
`\AtEndOfInputFile{<file name>}{<TeX code>}`

Like the `\...OfIncludeFile{<file name>}{code}` macros above, just for file read using `\input`. If the *<file name>* does not include a file extension it will be set to `.tex`.

The following figure shows the positions of the hooks inside the macro:

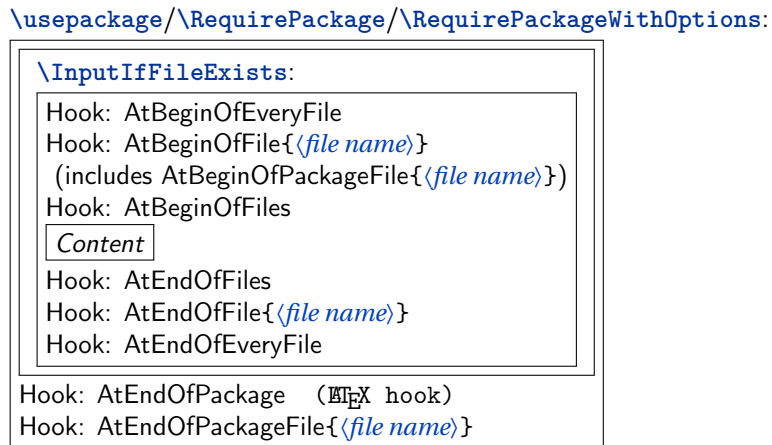


Package Files

<code>\AtBeginOfPackageFile*{<package name>}{<TeX code>}</code> <code>\AtEndOfPackageFile*{<package name>}{<TeX code>}</code>
--

This macros install the given *<TeX code>* in the 'AtBegin' and 'AtEnd' hooks of the given package file. The `\AtBeginOfPackageFile` simply executes `\AtBeginOfFile{<package name>.sty}{<TeXcode>}`. Special care is taken to ensure that the 'AtEnd' code is executed *after* any code installed by the package itself using the \TeX macro `\AtEndOfPackage`. Note that it is therefore executed after the 'AtEndOfEveryFile' hook. If the starred version is used and the package is already loaded the code is executed right away.

The following figure shows the positions of the hooks inside the macros:



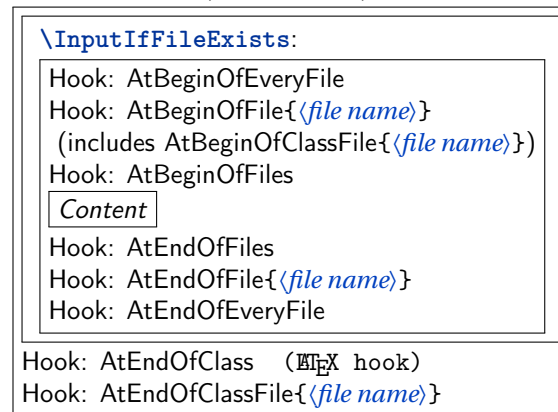
Class Files

```
\AtBeginOfClassFile*{<class name>}{<TEX code>}
\AtEndOfClassFile*{<class name>}{<TEX code>}
```

This macros install the given *<T_EX code>* in the 'AtBegin' and 'AtEnd' hooks of the given class file. They work with classes loaded using `\LoadClass`, `\LoadClassWithOptions` and also `\documentclass`. However, in the latter case filehook must be loaded using `\RequirePackage` beforehand. The macro `\AtBeginOfClassFile` simply executes `\AtBeginOfFile{<class name>.cls}{. . .}`. Special care is taken to ensure that the 'AtEnd' code is executed *after* any code installed by the class itself using the \TeX macro `\AtEndOfClass`. Note that it is therefore executed after the 'AtEnd-OfEveryFile' hook. If the starred version is used and the class is already loaded the code is executed right away.

The following figure shows the positions of the hooks inside the macros:

`\documentclass/\LoadClass/\LoadClassWithOptions:`



2.1 Clearing Hooks

```
\ClearHook\At...Of...<argument(s) of hook macro>
```

New in v0.5 2011/01/09 Using this macro existing hooks can be globally cleared, i.e. set to empty. This should be used with care because it will also remove all (user level) hook code set by packages into this hook. Note that the special hook code installed by the packages `currfile` and `svn-multi` as well as the compatibility code described in section 4 is not affected. The syntax for this macro is the same as for the normal hook macros only with a leading `\ClearHook`, where the *<code>* argument is mandatory but its content is ignored. Examples:

```
\ClearHook\AtBeginOfInputFile{<file name>}{<ignored>}
\ClearHook\AtBeginOfFiles{<ignored>}
```

3 PGF Key Interface

An auxiliary package `pgf-filehook` is provided which adds support for the versatile `pgfkeys` interface. This interface is heavily used by `pgf` (portable graphics format) and its higher level format `TikZ`. It allows the definition and execution of styles and commands (macros) using a `<key>=<value>` format. Main benefits over similar formats is the support for a “directory structure” inside the key and the ability to call functions on the value before it gets processed by the key. The main way to define and execute keys is the macro `\pgfkeys{<key>=<value>, ...}`. `TikZ` provides the similar macro `\tikzstyle` which defaults to the main path `'/tikz'`. More detailed information can be found in the official `pgfmanual`.

All `filehook` macros described in the previous section (`\AtXXXOfYYY`) can also be accessed using the `pgf keys` directory `'/filehook'`, where all hook type have an own sub-directory (`/filehook/YYY`) in which the hooks for this type are located (`/filehook/YYY/AtXXX`). For example `\AtBeginOfInputs{<code>}` can also be accessed using

```
\pgfkeys{/filehook/Inputs/AtBegin={<code>}}
or \AfterIncludeFile{<file name>}{<code>} as
\pgfkeys{/filehook/IncludeFile/After={<file name>}{<code>}}
as well as \AtEndOfClassFile*{<file name>}{<code>} as
\pgfkeys{/filehook/ClassFile/AtEnd=*{<file name>}{<code>}}.
```

`\pgffilehook{<key>=<value>, ...}`

This macro is like `\pgfkeys` but defaults to the `'/filehook'` directory, so that it can be dropped from the `<key>`. Note that `pgfkeys` also supports to “change the directory” using `<directory>/ . cd`, so that it does not need to be included in further keys. All directories are defined as *'is family'* so that the `/ . cd` is assumed if the directory is used on its own. For example

```
\pgfkeys{/filehook/Inputs/AtBegin={<code>},/filehook/Inputs/AtEnd={<code>}}
can be shorten as
```

```
\pgffilehook{Inputs,AtBegin={<code>},AtEnd={<code>}}.
```

Some of the `pgf key` functions can become useful, e.g. if the hook code should be expanded before it is added to the hook:

```
\pgffilehook{EveryFile/AtBegin/.expand once={\headertext \currfilename}}
```

will expand the first macro `\headertext` (actually the first token) in the hook code once (using `\expandafter`), but not any other tokens. In this example future changes of `\headertext` would not have any effect on the hook code, but `\currfilename` will be expanded for every file. Other useful functions are `' . expand twice'` (expand the first token twice) and `' . expanded'` (expand the whole hook code using `\edef`).

4 Compatibility Issues with Classes and other Packages

The `filehook` package might clash with other packages or classes which also redefine `\InputIfFileExists` or internal macros used by `\include` and `\input` (which are `\@input@` and `\@iinput`). Special compatibility code is in place for the packages listed below (in their current implementation). If any other unknown definition of `\InputIfFileExists` is found an error will be raised. The package option ‘force’ can be used to prevent this and to force the redefinition of this macro. Then any previous modifications will be lost, which will most likely break the other package. Table 1 lists all packages and classes which were found to be incompatible. The packages `auxhook`, `stampinclude`, `rerunfilecheck` and `excludeonly` redefine one or more of the above macros but have been found compatible with `filehook`. Please do not hesitate to inform the author of `filehook` of any encountered problems with other packages.

4.1 Supported Classes and Packages

The following classes and packages are actively supported and should work as normal when used together with `filehook`. Please note that most of them are incompatible to each other, which `filehook` might not fix.

memoir

The `memoir` class redefines `\InputIfFileExists` to add own hooks identical to the ‘At...OfFiles’ hooks (there called `\AtBeginFile` and `\AtEndFile`). These hooks will be moved to the corresponding ones of `filehook` and will keep working as normal. Since v0.4 from 2011/01/03 this modification will be also applied when the `filehook` package is loaded (using `\RequirePackage`) before the `memoir` class. However, the hooks from `filehook` need to be temporally disabled while reading the `memoir` class. They will not be triggered for all files read directly by this class, like configuration and patch files. Note that the ‘At...OfClassFile’ hooks still work for the `memoir` class file itself. In fact they are used to restore the default definition of `\InputIfFileExists` at the begin and patch it at the end of the class file. The `filehook` package should be loaded either before the class (using `\RequirePackage`) or directly after it. Because the `memoir` hook code is moved to the `filehook` hooks this class should then be compatible with below packages if `memoir` and `filehook` are loaded before them.

scrfile

The `scrfile` package from the *koma-script* bundle redefines `\InputIfFileExists` to allow file name aliases and to also add hooks. If required it should be loaded before `filehook`, which will add its hooks correctly to the modified definition. Since v0.4 from 2011/01/03 this modification will be also applied when the `scrfile` package is loaded after `filehook`.

fink

The `filehook` and `currfile` packages were written as replacements for the `fink` package, where `filehook` provides the necessary hooks for `currfile`. The `fink` package has now been deprecated in favour of `currfile` and should not be used anymore. The `fink` compatibility code has been removed from `filehook` and both

Table 1: Incompatible packages and classes

Name	Type	Note	Affected Hooks
paper	class	with journal option	All hocks for <code>\include</code> 'd files
journal	class		All hocks for <code>\include</code> 'd files
gmparts	package		<code>\include</code> hooks
newclude	package	formally <code>includex</code>	All hocks for <code>\include</code> 'd files

cannot be used successfully together as both redefine the `\InputIfFileExists` macro.

listings

The `listings` package uses `\input` inside `\lstinputlisting`. Therefore the `InputFile(s)` and `File(s)` hooks are also triggered for these files. Please note that this hooks are executing inside a verbatim environment. While the code in the hook is not affected (because it was added outside the verbatim environment), any further code read using any input macro (`\input`, `\@input`, `\@@input` (TeX's `\input`), ...) will be processed verbatim and typeset as part of the listing. Since v0.4 this macro is automatically patched so `\@input` is used instead to avoid this issue.

4.2 Other Classes and Packages

jmlrbook

The `jmlrbook` class from the `jmlr` bundle temporary redefines `\InputIfFileExists` to import papers. The 'original' definition is saved away at load time of the package and is used internally by the new definition. This means that the hooks will not be active for this imported files because `filehook` is loaded after the class. This should not affect its normal usage. Note that, in theory, the package could be loaded before `\documentclass` using `\RequirePackage` to enable the file hooks also for these files.

TeX's `\bibliography`

The standard TeX macro `\bibliography` uses the same internal macro `\@input@` to read a file as `\include` does. The 'include' hooks will also be executed for this `.bb1` file if the macro is directly followed by `\clearpage`, because the `filehook` code will assume it is executed inside `\include`. This rare case can be easily avoided by placing a `\relax` after `\bibliography{...}`.

5 Upgrade Guide

This sections gives information for users of older versions of this package which unfortunately might not be 100% backwards compatible.

Upgrade to v0.4 - 2011/01/03

- The macro `\AfterIncludeFile` was misspelled as `\AfterOfIncludeFile` in the implementation of earlier versions, but not in the documentation. This has now be corrected. Please adjust your code to use the correct name and to require the `filehook` package from 2011/01/03.
- All general hooks (the one not taking a file argument) used to have an implicit argument #1 which was expanded to the file name (i.e. the argument of `\input` etc.). This has now be changed, so that macro arguments are not handled special in hook code, which e.g. simplifies macro definitions. Older hook code might need to change `##` to `#` to compensate for this change. If the file name is required the macros (e.g. `\currfilename`) of the partner package `currfile` should be used. These macros are available everywhere including in all hocks.

6 Implementation

```
1 %<!COPYRIGHT>
2 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
3 \ProvidesPackage{filehook}[%
4 %<!DATE>
5 %<!VERSION>
6 %<*DRIVER>
7     2099/01/01 develop
8 %</DRIVER>
9     Hooks for input files]
```

6.1 Options

```
10 \newif\iffilehook@force
11 \DeclareOption{force}{\filehook@forcetrue}
12 \ProcessOptions\relax
```

6.2 Initialisation of Hooks

The general hooks are initialised to call the file specific hooks.

```
\filehook@csuse
```

```
13 \begingroup
14 \gdef\filehook@csuse#1{\ifcsname #1\endcsname\csname /
15     #1\expandafter\endcsname\fi}
16 \expandafter\ifx\csname csuse\endcsname\relax
17     \expandafter\ifx\csname ifcsname\endcsname\relax
18         \gdef\filehook@csuse#1{\expandafter\ifx\
19             csname #1\endcsname\relax\else\csname #1\
20             expandafter\endcsname\fi}
21     \fi
22 \else
23     \global\let\filehook@csuse\csuse
24 \fi
25 \endgroup
```

```
\filehook@include@atbegin
```

```
23 \def\filehook@include@atbegin#1{%
24     \let\InputIfFileExists\filehook@@InputIfFileExists
25     \filehook@csuse{\filehook@include@atbegin@#1}%
26     \filehook@include@@@atbegin
27 }
```

`\filehook@include@@atbegin`

```
28 \def\filehook@include@@atbegin{}
```

`\filehook@include@atend`

```
29 \def\filehook@include@atend#1{%  
30   \filehook@include@@atend  
31   \filehook@csuse{\filehook@include@atend@#1}%  
32 }
```

`\filehook@include@@atend`

```
33 \def\filehook@include@@atend{}
```

`\filehook@include@after`

```
34 \def\filehook@include@after#1{%  
35   \filehook@include@@after  
36   \filehook@csuse{\filehook@include@after@#1}%  
37 }
```

`\filehook@include@@after`

```
38 \def\filehook@include@@after{}
```

`\filehook@input@atbegin`

```
39 \def\filehook@input@atbegin#1{%  
40   \let\InputIfFileExists\filehook@@InputIfFileExists  
41   \filehook@csuse{\filehook@input@atbegin@\  
         filehook@ensureext{#1}}%  
42   \filehook@input@@atbegin  
43 }
```

`\filehook@input@@atbegin`

```
44 \def\filehook@input@@atbegin{}
```

`\filehook@input@atend`

```
45 \def\filehook@input@atend#1{%  
46   \filehook@input@@atend  
47   \filehook@csuse{\filehook@input@atend@\  
      filehook@ensureext{#1}}%  
48 }
```

`\filehook@input@@atend`

```
49 \def\filehook@input@@atend{}
```

`\filehook@atbegin`

```
50 \def\filehook@atbegin#1{%  
51   \filehook@csuse{\filehook@atbegin@\  
      filehook@ensureext{#1}}%  
52   \filehook@@@atbegin  
53 }
```

`\filehook@@@atbegin`

```
54 \def\filehook@@@atbegin{}
```

`\filehook@atend`

```
55 \def\filehook@atend#1{%  
56   \filehook@@@atend  
57   \filehook@csuse{\filehook@atend@\  
      filehook@ensureext/  
      {#1}}%  
58 }
```

`\filehook@@@atend`

```
59 \def\filehook@@@atend{}
```

`\filehook@every@atbegin`

```
60 \def\filehook@every@atbegin#1{%  
61   \filehook@every@@@atbegin  
62 }
```

```
\filehook@every@@atbegin
```

```
63 \def\filehook@every@@atbegin{}
```

```
\filehook@every@atend
```

```
64 \def\filehook@every@atend#1{%  
65     \filehook@every@@atend  
66 }
```

```
\filehook@every@@atend
```

```
67 \def\filehook@every@@atend{}
```

6.3 Hook Modification Macros

The following macros are used to modify the hooks, i.e. to prefix or append code to them.

Internal Macros

The macro prefixes for the file specific hooks are stored in macros to reduce the number of tokens in the following macro definitions.

```
68 \def\filehook@include@atbegin@{/  
    filehook@include@atbegin@}  
69 \def\filehook@include@atend@{filehook@include@atend@}  
70 \def\filehook@include@after@{filehook@include@after@}  
71 \def\filehook@input@atbegin@{filehook@input@atbegin@}  
72 \def\filehook@input@atend@{filehook@input@atend@}  
73 \def\filehook@input@after@{filehook@input@after@}  
74 \def\filehook@atbegin@{filehook@atbegin@}  
75 \def\filehook@atend@{filehook@atend@}  
76 \def\filehook@after@{filehook@after@}
```

```
\filehook@append
```

Uses default \LaTeX macro.

```
77 \def\filehook@append{\g@addto@macro}
```

`\filehook@appendwarg`

Appends code with one macro argument. The `\@tempa` intermediate step is required because of the included `##1` which wouldn't correctly expand otherwise.

```
78 \long\def\filehook@appendwarg#1#2{%
79   \begingroup
80     \toks@\expandafter{#1{##1}#2}%
81     \edef\@tempa{\the\toks@}%
82     \expandafter\gdef\expandafter#1\expandafter##\
      expandafter1\expandafter{\@tempa}%
83   \endgroup
84 }
```

`\filehook@prefix`

Prefixes code to a hook.

```
85 \long\def\filehook@prefix#1#2{%
86   \begingroup
87     \@temptokena{#2}%
88     \toks@\expandafter{#1}%
89     \xdef#1{\the\@temptokena\the\toks@}%
90   \endgroup
91 }
```

`\filehook@prefixwarg`

Prefixes code with an argument to a hook.

```
92 \long\def\filehook@prefixwarg#1#2{%
93   \begingroup
94     \@temptokena{#2}%
95     \toks@\expandafter{#1{##1}}%
96     \edef\@tempa{\the\@temptokena\the\toks@}%
97     \expandafter\gdef\expandafter#1\expandafter##\
      expandafter1\expandafter{\@tempa}%
98   \endgroup
99 }
```

`\filehook@addtohook`

#1: Macro which should be used to add the material to the hook

#2: Macro name prefix

#3: End of macro name (file name)

The macro first expands the file name (**#3**) to flatten all included macros. An extension is added if missing, as well as the prefix. All modifications of `\@tempa` are made inside a group to keep them local.

```

100 \def\filehook@addtohook#1#2#3{%
101   \begingroup
102   \edef\@tempa{#3}%
103   \edef\@tempa{#2\filehook@ensureext{\@tempa}}%
104   \@ifundefined{\@tempa}{\global\@namedef{\@tempa/
      }{}}{}%
105   \expandafter\endgroup
106   \expandafter#1\csname\@tempa\endcsname
107 }

```

User Level Macros

The user level macros simple use the above defined macros on the appropriate hook.

\AtBeginOfIncludes

```

108 \newcommand*\AtBeginOfIncludes{%
109   \filehook@append\filehook@include@@atbegin
110 }

```

\AtEndOfIncludes

```

111 \newcommand*\AtEndOfIncludes{%
112   \filehook@prefix\filehook@include@@atend
113 }

```

\AfterIncludes

```

114 \newcommand*\AfterIncludes{%
115   \filehook@prefix\filehook@include@@after
116 }

```

\AtBeginOfIncludeFile

```

117 \newcommand*\AtBeginOfIncludeFile [1]{%
118   \filehook@addtohook\filehook@append\
      filehook@include@atbegin@{\filehook@ensuretex/
      {#1}}%
119 }

```


\AtEndOfIncludeFile

```
120 \newcommand*\AtEndOfIncludeFile [1]{%
121   \filehook@addtohook\filehook@prefix\
      filehook@include@atend@{\filehook@ensuretex{#1}}\
      %
122 }
```

\AfterIncludeFile

```
123 \newcommand*\AfterIncludeFile [1]{%
124   \filehook@addtohook\filehook@prefix\
      filehook@include@after@{\filehook@ensuretex{#1}}\
      %
125 }
```

\AtBeginOfInputs

```
126 \newcommand*\AtBeginOfInputs{%
127   \filehook@append\filehook@input@@atbegin
128 }
```

\AtEndOfInputs

```
129 \newcommand*\AtEndOfInputs{%
130   \filehook@prefix\filehook@input@@atend
131 }
```

\AtBeginOfInputFile

```
132 \newcommand*\AtBeginOfInputFile{%
133   \filehook@addtohook\filehook@append\
      filehook@input@atbegin@
134 }
```

\AtEndOfInputFile

```
135 \newcommand*\AtEndOfInputFile{%
136   \filehook@addtohook\filehook@prefix\
      filehook@input@atend@
137 }
```

`\AtBeginOfFiles`

```
138 \newcommand*\AtBeginOfFiles{%  
139   \filehook@append\filehook@@atbegin  
140 }
```

`\AtEndOfFiles`

```
141 \newcommand*\AtEndOfFiles{%  
142   \filehook@prefix\filehook@@atend  
143 }
```

`\AtBeginOfEveryFile`

```
144 \newcommand*\AtBeginOfEveryFile{%  
145   \filehook@append\filehook@every@@atbegin  
146 }
```

`\AtEndOfEveryFile`

```
147 \newcommand*\AtEndOfEveryFile{%  
148   \filehook@prefix\filehook@every@@atend  
149 }
```

`\AtBeginOfFile`

```
150 \newcommand*\AtBeginOfFile{%  
151   \filehook@addtohook\filehook@append\  
       filehook@atbegin@  
152 }
```

`\AtEndOfFile`

```
153 \newcommand*\AtEndOfFile{%  
154   \filehook@addtohook\filehook@prefix\filehook@atend@  
155 }
```

`\AtBeginOfClassFile`

```
156 \newcommand*\AtBeginOfClassFile{%
157     \@ifnextchar*
158         {\AtBeginOfXFile@star\@clsextension}%
159         {\AtBeginOfXFile@normal\@clsextension}%
160 }
```

`\AtBeginOfPackageFile`

```
161 \newcommand*\AtBeginOfPackageFile{%
162     \@ifnextchar*
163         {\AtBeginOfXFile@star\@pkgextension}%
164         {\AtBeginOfXFile@normal\@pkgextension}%
165 }
```

`\AtBeginOfXFile@star`

#1: extension
#2: name

If the class or package is already loaded the code is executed right away. Otherwise it is installed normally.

```
166 \def\AtBeginOfXFile@star#1*#2{%
167     \@ifl@aded{#1}{#2}%
168     {\@firstofone}%
169     {\AtBeginOfXFile@normal{#1}{#2}}%
170 }
```

`\AtBeginOfXFile@normal`

#1: extension
#2: name

```
171 \def\AtBeginOfXFile@normal#1#2{%
172     \AtBeginOfFile{#2.#1}%
173 }
```

`\AtEndOfClassFile`

```
174 \newcommand*\AtEndOfClassFile{%
175     \@ifnextchar*
176         {\AtEndOfXFile@star\@clsextension}%
177         {\AtEndOfXFile@normal\@clsextension}%
178 }
```

`\AtEndOfPackageFile`

```
179 \newcommand*\AtEndOfPackageFile{%
180     \@ifnextchar*
181         {\AtEndOfXFile@star\@pkgextension}%
182         {\AtEndOfXFile@normal\@pkgextension}%
183 }
```

`\AtEndOfXFile@star`

#1: extension

#2: name

If the class or package is already loaded the code is executed right away. Otherwise it is installed normally.

```
184 \def\AtEndOfXFile@star#1*#2{%
185     \@ifl@aded{#1}{#2}%
186     {\@firstofone}%
187     {\AtEndOfXFile@normal{#1}{#2}}%
188 }
```

`\AtEndOfXFile@normal`

#1: extension

#2: name

Note that `\AtEndOfClass` is identical to `\AtEndOfPackage`, so no differentiation between classes and packages is needed here.

```
189 \long\def\AtEndOfXFile@normal#1#2#3{%
190     \AtEndOfFile{#2.#1}{\AtEndOfPackage{#3}}%
191 }
```

`\ClearHook`

Clears the hook by temporary redefining the prefix and append macros to do a simple definition to empty.

```
192 \newcommand*\ClearHook{%
193     \begingroup
194     \def\filehook@prefix##1##2{%
195         \gdef##1{}%
196         \endgroup
197     }%
198     \let\filehook@append\filehook@prefix
199 }
```

6.4 Installation of Hooks

The `\@input@` and `\@iinput` macros from `latex.ltx` are redefined to install the hooks.

First the original definitions are saved away.

```
\filehook@orig@@input@
```

```
200 \let\filehook@orig@@input@\@input@
```

```
\filehook@orig@@iinput
```

```
201 \let\filehook@orig@@iinput\@iinput
```

```
\@input@
```

This macro is redefined for the `\include` file hooks. Checks if the next command is `\clearpage` which indicates that we are inside `\@include`. If so the hooks are installed, otherwise the original macro is used unchanged. For the ‘after’ hook an own `\clearpage` is inserted and the original one is gobbled.

```
202 \def\@input@#1{%
203   \ifnextchar\clearpage
204     {%
205       \filehook@every@atbegin{#1}%
206       \filehook@include@atbegin{#1}%
207       \filehook@orig@@input@{#1}%
208       \filehook@include@atend{#1}%
209       \clearpage
210       \filehook@include@after{#1}%
211       \filehook@every@atend{#1}%
212       \@gobble
213     }%
214     {\filehook@orig@@input@{#1}}%
215 }
```

```
\@iinput
```

This macro is redefined for the `\input` file hooks. it simply surrounds the original macro with the hooks.

```
216 \def\filehook@@iinput#1{%
217   \filehook@every@atbegin{#1}%
218   \filehook@input@atbegin{#1}%
219   \filehook@orig@@iinput{#1}%
220   \filehook@input@atend{#1}%
```

```

221 \filehook@every@atend{#1}%
222 }
223 \let\@iinput\filehook@@iinput

```

`\filehook@swap`

Auxiliary macro which swaps the two arguments. This is needed to expand `\@filef@und`, which is given as first argument but needed then as the second one.

```

224 \def\filehook@swap#1#2{#2#1}

```

`\filehook@ensureext`

This macro ensures the existence of a file name extension. If non is given ‘.tex’ is added.

```

225 \def\filehook@ensureext#1{%
226 \expandafter\filehook@@ensureext#1\empty.tex\
      empty\empty
227 }

```

`\filehook@@ensureext`

```

228 \def\filehook@@ensureext#1.#2\empty#3\empty{#1.#2}

```

`\filehook@ensuret看`

Ensures a ‘.tex’ extension, i.e. adds it if missing, even if there is a different one.

```

229 \def\filehook@ensuret看#1{%
230 \expandafter\filehook@@ensuret看#1\empty.tex\
      empty\empty
231 }

```

`\filehook@@ensuret看`

```

232 \def\filehook@@ensuret看#1.tex\empty#2\empty{#1.tex}

```

The filehook default definition of `\InputIfFileExists` is defined here together with alternatives definitions for comparison. There are stored first in a token register and later stored in a macro which is expanded if required. This is always done inside a group to keep them temporary only. The token register is used to avoid doubling of macro argument characters.

```

233 \newif\iffilehook@newfmt
234 \@ifl@t@r\fmtversion{2019/10/01}{\filehook@newfmttrue\
      }{\filehook@newfmtfalse}

```

`\latex@InputIfFileExists`

Standard \LaTeX definition of `\InputIfFileExists`.

```
235 \iffilehook@newfmt
236 \DeclareRobustCommand \latex@InputIfFileExists [2]{%
237   \IfFileExists{#1}%
238   {%
239     \expandafter\@swaptwoargs\expandafter
240     {\@filef@und}{#2\@addtofilelist{#1}\@@input}}
241   \else
242   \long\def\latex@InputIfFileExists#1#2{%
243     \IfFileExists{#1}%
244     {#2\@addtofilelist{#1}%
245      \@@input\@filef@und
246     }%
247   }
248 \fi
```

`\filehook@default@InputIfFileExists`

```
249 \long\gdef\filehook@default@InputIfFileExists#1#2{%
250   \IfFileExists{#1}%
251   {\expandafter\filehook@swap
252    \expandafter{\@filef@und}%
253    {#2\@addtofilelist{#1}%
254     \filehook@every@atbegin{#1}%
255     \filehook@atbegin{#1}%
256     \@@input}%
257    \filehook@atend{#1}%
258    \filehook@every@atend{#1}%
259   }%
260 }
```

`\filehook@@default@InputIfFileExists`

```
261 \long\gdef\filehook@@default@InputIfFileExists#1#2{%
262   \let\InputIfFileExists\filehook@InputIfFileExists
263   \IfFileExists{#1}%
264   {\expandafter\filehook@swap
265    \expandafter{\@filef@und}%
266    {#2\@addtofilelist{#1}%
267     \filehook@atbegin{#1}%
268     \@@input}%
269    \filehook@atend{#1}%
270   }%
271 }
```

\scrfile@InputIfFileExists

```
272 \long\def\scrfile@InputIfFileExists#1#2{%
273   \begingroup\expandafter\expandafter\expandafter\
      endgroup
274   \expandafter\ifx\csname #1-@alias\endcsname\relax
275     \expandafter\@secondoftwo
276   \else
277     \scr@replacefile@msg{\csname #1-@alias\endcsname/
      }{#1}%
278     \expandafter\@firstoftwo
279   \fi
280   {%
281     \expandafter\InputIfFileExists\expandafter{\
      csname
282       #1-@alias\endcsname}{#2}%
283   }%
284   {\IfFileExists{#1}{%
285     \scr@load@hook{before}{#1}%
286     #2\@addtofilelist{#1}%
287     \@input \@filef@und
288     \scr@load@hook{after}{#1}%
289   }}%
290 }
```

\filehook@scrfile@InputIfFileExists

```
291 \long\def\filehook@scrfile@InputIfFileExists#1#2{%
292   \begingroup\expandafter\expandafter\expandafter\
      endgroup
293   \expandafter\ifx\csname #1-@alias\endcsname\relax
294     \expandafter\@secondoftwo
295   \else
296     \scr@replacefile@msg{\csname #1-@alias\endcsname/
      }{#1}%
297     \expandafter\@firstoftwo
298   \fi
299   {%
300     \expandafter\InputIfFileExists\expandafter{\
      csname
301       #1-@alias\endcsname}{#2}%
302   }%
303   {\IfFileExists{#1}{%
304     \expandafter\filehook@swap
305     \expandafter{\@filef@und}%
306     {\scr@load@hook{before}{#1}%
307     #2\@addtofilelist{#1}%
```



```

308     \filehook@every@atbegin{#1}%
309     \filehook@atbegin{#1}%
310     \@@input}%
311     \filehook@atend{#1}%
312     \filehook@every@atend{#1}%
313     \scr@load@hook{after}{#1}%
314   }}%
315 }

```

\filehook@@scrfile@InputIfFileExists

```

316 \long\def\filehook@@scrfile@InputIfFileExists#1#2{%
317   \let\InputIfFileExists\filehook@InputIfFileExists
318   \begingroup\expandafter\expandafter\expandafter\
319     endgroup
320   \expandafter\ifx\csname #1-@alias\endcsname\relax
321     \expandafter\@secondoftwo
322   \else
323     \scr@replacefile@msg{\csname #1-@alias\endcsname\
324       }{#1}%
325     \expandafter\@firstoftwo
326   \fi
327   {%
328     \expandafter\InputIfFileExists\expandafter{\
329       csname
330       #1-@alias\endcsname}{#2}%
331     }%
332     {\IfFileExists{#1}{%
333       \expandafter\filehook@swap
334       \expandafter{\@filef@und}%
335       {\scr@load@hook{before}{#1}%
336       #2\@addtofilelist{#1}%
337       \filehook@atbegin{#1}%
338       \@@input}%
339       \filehook@atend{#1}%
340       \scr@load@hook{after}{#1}%
341     }}%
342 }

```

\InputIfFileExists

First we test for the `scrfile` package. The test macro adds the necessary patches if so. In order to also support it when it is loaded afterwards the two hooks below are used to revert the definition before the package and patch it afterwards.

```

340 \AtBeginOfPackageFile*{scrfile}{%
341   \let\InputIfFileExists\latex@InputIfFileExists
342 }%

```

```

343 \AtEndOfPackageFile*{scrfile}{%
344   \RequirePackage{filehook-scrfile}%
345 }%

Fink:

346 \AtBeginOfPackageFile*{fink}{%
347   \RequirePackage{kvoptions}%
348   \begingroup
349   \let\InputIfFileExists\latex@InputIfFileExists
350 }%
351 \AtEndOfPackageFile*{fink}{%
352   \edef\@tempa{\noexpand\PassOptionsToPackage{/
mainext=\fnk@mainext ,maindir=\fnk@maindir}{/
currfile}}%
353   \expandafter\endgroup\@tempa
354   \RequirePackage{filehook-fink}%
355 }%

```

If memoir is detected its hooks are added to the appropriate 'At...OfFiles' hooks. This works fine because its hooks have the exact same position. Please note that the case when memoir is used together with scrfile is not explicitly covered. In this case the scrfile package will overwrite memoirs definition.

```

356 \AtBeginOfClassFile*{memoir}{%
357   \let\filehook@@InputIfFileExists\
latex@InputIfFileExists
358   \let\InputIfFileExists\latex@InputIfFileExists
359   \let\@iinput\filehook@orig@@iinput
360 }%
361 \AtEndOfClassFile*{memoir}{%
362   \let\@iinput\filehook@@iinput
363   \RequirePackage{filehook-memoir}%
364 }%

```

Finally, if no specific alternate definition is detected the original \LaTeX definition is checked for and a error is given if any other unknown definition is detected. The `force` option will change the error into a warning and overwrite the macro with the default.

```

365 \def\@tempa{9}%
366 \ifx\InputIfFileExists\filehook@InputIfFileExists
367   \def\@tempa{0}%
368 \else
369   \iffilehook@force
370     \def\@tempa{1}%
371   \else
372     \iffilehook@newfmt
373     % check if both the robust \
InputIfFileExist and its internal /
macro match the default LaTeX /
definition
374     \edef\@tempb{\noexpand\protect\
expandafter\noexpand\csname /
InputIfFileExists\space\endcsname}%

```

```

375         \ifx\InputIfFileExists \@tempb
376             \expandafter\ifx\csname /
                 InputIfFileExists\space\
                 expandafter\endcsname\csname /
                 latex@InputIfFileExists\space\
                 endcsname
377             \def\@tempa{1}%
378         \fi
379     \fi
380 \else\ifx\InputIfFileExists\
                 latex@InputIfFileExists
381     \def\@tempa{1}%
382 \fi\fi
383 \fi
384 \fi
385 %
386 \ifcase\@tempa
387 \relax% 0
388 \or% 1
389     \let\filehook@InputIfFileExists\
                 filehook@default@InputIfFileExists
390     \let\filehook@@InputIfFileExists\
                 filehook@@default@InputIfFileExists
391 \iffilehook@newfmt
392     \expandafter\let\csname InputIfFileExists\
                 space\endcsname\filehook@InputIfFileExists
393 \else
394     \let\InputIfFileExists\
                 filehook@InputIfFileExists
395 \fi
396 \iffilehook@force
397     \PackageWarning{filehook}{Detected unknown /
                 definition of \string\InputIfFileExists.^~J%
398                                     The 'force' option of/
                                     'filehook' is in /
                                     effect. Macro is /
                                     overwritten with /
                                     default!}%
399 \fi
400 \else
401     \PackageError{filehook}{Detected unknown /
                 definition of \string\InputIfFileExists.^~J%
402                                     Use the 'force' option of/
                                     'filehook' to /
                                     overwrite it.}}%
403 \fi
404 \AtBeginDocument{%
405     % Check if definition got changed again. For the /
                 new LaTeX format we check again \

```

```

    InputIfFileExists <space>,
406   % for the old format to \InputIfFileExists /
      directly.
407   \expandafter\ifx\csname InputIfFileExists\
      iffilehook@newfmt\space\fi\endcsname\
      filehook@InputIfFileExists\else
408     \PackageWarning{filehook}{Macro \string\
      InputIfFileExists\space got redefined /
      after 'filehook' was loaded.^^J%
409                                     Certain file hooks /
                                          might now be /
                                          dysfunctional!}
410   \fi
411 }

412 %<!COPYRIGHT>
413 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
414 \ProvidesPackage{filehook-memoir}[2011/01/03 v0.1 /
      filehook patch for memoir class]

415 \RequirePackage{filehook}
416 \begingroup

```

<pre> \memoir@InputIfFileExists </pre>
--

```

417 \long\def\memoir@InputIfFileExists#1#2{%
418   \IfFileExists{#1}%
419     {#2\@addtofilelist{#1}\m@matbeginf{#1}%
420       \@input \@filef@und
421       \m@matendf{#1}%
422       \killm@matf{#1}}%
423   }

424 \ifcase
425   \ifx\InputIfFileExists\latex@InputIfFileExists 0\
      else
426   \ifx\InputIfFileExists\memoir@InputIfFileExists /
      0\else
427     1%
428     \fi\fi
429 \relax
430 \global\let\filehook@InputIfFileExists\
      filehook@default@InputIfFileExists
431 \global\let\filehook@@InputIfFileExists\
      filehook@@default@InputIfFileExists
432 \global\let\InputIfFileExists\
      filehook@InputIfFileExists
433 \filehook@appendwarg\filehook@atbegin{\m@matbeginf{
      #1}}%

```

```

434 \filehook@prefixwarg\filehook@atend{\m@matendf{#1}\
      killm@matf{#1}}%
435 \PackageInfo{filehook}{Detected 'memoir' class: the
      memoir hooks will be moved to the 'At...OfFiles
      ' hooks}
436 \else
437 \iffilehook@force
438 \global\let\filehook@InputIfFileExists\
      filehook@default@InputIfFileExists
439 \global\let\filehook@@InputIfFileExists\
      filehook@@default@InputIfFileExists
440 \global\let\InputIfFileExists\
      filehook@InputIfFileExists
441 \PackageWarning{filehook}{Detected 'memoir' class
      with unknown definition of \string\
      InputIfFileExists.^~J%
442                                     The 'force' option of '
      filehook' is in
      effect. Macro is
      overwritten with
      default!}%
443 \else
444 \PackageError{filehook}{Detected 'memoir' class
      with unknown definition of \string\
      InputIfFileExists.^~J%
445                                     Use the 'force' option of
      'filehook' to
      overwrite it.}{}%
446 \fi
447 \fi
448 \endgroup
449 %<!COPYRIGHT>
450 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
451 \ProvidesPackage{filehook-listings}[2011/01/02 v0.1
      Patch for listings to avoid hooks for verbatim
      input files]
452 \begingroup
453
454 \long\def\patch#1\def\lst@next#2#3\endpatch{%
455     \toks@{#2}%
456     \edef\@tempa{\the\toks@}%
457     \def\@tempb{\input{###1}}%
458     \ifx\@tempa\@tempb
459         \gdef\lst@InputListing##1{#1\def\lst@next{\
      @input{##1}}#3}%
460     \else

```

```

461         \PackageWarning{filehook-listings}{To-be-
           patched code in macro \string\
           lst@InputListing was not found!}%
462     \fi
463 }
464
465 \@ifundefined{lst@InputListing}{%
466     \PackageWarning{filehook-listings}{To-be-patched
           Macro \string\lst@InputListing not found!}%
467 }{}
468
469 \expandafter\patch\lst@InputListing{#1}\endpatch
470
471 \endgroup
472
473 %<!COPYRIGHT>
474 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
475 \ProvidesPackage{filehook-scrfile}[2011/01/03 v0.1
           filehook patch for scrfile package]
476 \RequirePackage{filehook}
477 \begingroup

```

<pre> \scrfile@InputIfFileExists </pre>

```

477 \long\def\scrfile@InputIfFileExists#1#2{%
478     \begingroup\expandafter\expandafter\expandafter\
           endgroup
479     \expandafter\ifx\csname #1-@alias\endcsname\relax
480         \expandafter\@secondoftwo
481     \else
482         \scr@replacefile@msg{\csname #1-@alias\endcsname\
           }{#1}%
483         \expandafter\@firstoftwo
484     \fi
485     {%
486         \expandafter\InputIfFileExists\expandafter{\
           csname
           #1-@alias\endcsname}{#2}%
487     }%
488     {\IfFileExists{#1}{%
489         \scr@load@hook{before}{#1}%
490         #2\@addtofilelist{#1}%
491         \@input \@filef@und
492         \scr@load@hook{after}{#1}%
493     }}%
494 }
495 }

```

`\filehook@scrfile@InputIfFileExists`

```
496 \long\def\filehook@scrfile@InputIfFileExists#1#2{%
497   \begingroup\expandafter\expandafter\expandafter\
      endgroup
498   \expandafter\ifx\csname #1-@alias\endcsname\relax
499     \expandafter\@secondoftwo
500   \else
501     \scr@replacefile@msg{\csname #1-@alias\endcsname/
      }{#1}%
502     \expandafter\@firstoftwo
503   \fi
504   {%
505     \expandafter\InputIfFileExists\expandafter{\
      csname
506       #1-@alias\endcsname}{#2}%
507   }%
508   {\IfFileExists{#1}{%
509     \expandafter\filehook@swap
510     \expandafter{\@filef@und}%
511     {\scr@load@hook{before}{#1}%
512      #2\@addtofilelist{#1}%
513      \filehook@every@atbegin{#1}%
514      \filehook@atbegin{#1}%
515      \@input}%
516      \filehook@atend{#1}%
517      \filehook@every@atend{#1}%
518      \scr@load@hook{after}{#1}%
519     }}%
520 }
```

`\filehook@@scrfile@InputIfFileExists`

```
521 \long\def\filehook@@scrfile@InputIfFileExists#1#2{%
522   \let\InputIfFileExists\filehook@InputIfFileExists
523   \begingroup\expandafter\expandafter\expandafter\
      endgroup
524   \expandafter\ifx\csname #1-@alias\endcsname\relax
525     \expandafter\@secondoftwo
526   \else
527     \scr@replacefile@msg{\csname #1-@alias\endcsname/
      }{#1}%
528     \expandafter\@firstoftwo
529   \fi
530   {%
531     \expandafter\InputIfFileExists\expandafter{\
      csname
```

```

532     #1-@alias\endcsname}{#2}%
533 }%
534 {\IfFileExists{#1}{%
535     \expandafter\filehook@swap
536     \expandafter{\@filef@und}%
537     {\scr@load@hook{before}{#1}%
538     #2\@addtofilelist{#1}%
539     \filehook@atbegin{#1}%
540     \@input}%
541     \filehook@atend{#1}%
542     \scr@load@hook{after}{#1}%
543 }}%
544 }

```

If the `scrfile` package definition is detected the filehooks are added to that definition. Unfortunately the `\scr@load@hook{before}` hook is placed *before* not after the `#2\@addtofilelist{#1}` code. Otherwise the filehooks could simply be added to these hooks. Note that this will stop working if `scrfile` ever changes its definition of the `\InputIfFileExists` macro.

```

545 \ifcase
546     \ifx\InputIfFileExists\latex@InputIfFileExists 0\
547     else
548     \ifx\InputIfFileExists\scrfile@InputIfFileExists\
549     0\else
550     1%
551     \fi\fi
552 \relax
553 \global\let\filehook@InputIfFileExists\
554     filehook@scrfile@InputIfFileExists
555 \global\let\filehook@@InputIfFileExists\
556     filehook@@scrfile@InputIfFileExists
557 \global\let\filehook@InputIfFileExists\
558     filehook@InputIfFileExists
559 \PackageInfo{filehook}{Package 'scrfile' detected \
560     and compensated for}%
561 \else
562     \iffilehook@force
563     \global\let\filehook@InputIfFileExists\
564         filehook@default@InputIfFileExists
565     \global\let\filehook@@InputIfFileExists\
566         filehook@@default@InputIfFileExists
567     \global\let\InputIfFileExists\
568         filehook@InputIfFileExists
569     \PackageWarning{filehook}{Detected 'scrfile' \
570         package with unknown definition of \string\
571         InputIfFileExists.^^J%
572         The 'force' option of '\
573         filehook' is in \
574         effect. Macro is \
575         overwritten with \

```



```

                    default!}%
562 \else
563   \PackageError{filehook}{Detected 'scrfile' /
     package with unknown definition of \string\
     InputIfFileExists.^~J%
564                                     Use the 'force' option of /
                                     'filehook' to /
                                     overwrite it.}{}%
565 \fi
566 \fi

567 \endgroup

568 %<!COPYRIGHT>
569 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
570 \ProvidesPackage{filehook-fink}[011/01/03 v0.1 /
     filehook compatibility code for fink package]

571 \RequirePackage{filehook}
572 \RequirePackage{currfile}%
573
574 \begingroup
575
576 \long\def\fink@old@InputIfFileExists#1#2{%
577   \IfFileExists{#1}{%
578     #2\@addtofilelist{#1}%
579     \fink@prepare{#1}%
580     \expandafter\fink@input%
581     \expandafter\fink@restore\expandafter{\finkpath}}/
     %
582 }
583
584 \long\def\fink@new@InputIfFileExists#1#2{%
585   \IfFileExists{#1}{%
586     #2\@addtofilelist{#1}%
587     \edef\fink@before{\noexpand\fink@input{#1}}%
588     \edef\fink@after{\noexpand\fink@restore{\finkpath /
     }}%
589     \expandafter\fink@before\fink@after}%
590 }
591
592 \ifcase
593   \ifx\InputIfFileExists\filehook@InputIfFileExists /
     0\else
594   \ifx\InputIfFileExists\latex@InputIfFileExists /
     1\else
595   \ifx\InputIfFileExists\fink@new@InputIfFileExists /
     1\else
596   \ifx\InputIfFileExists\fink@old@InputIfFileExists /
     1\else

```

```

597     1%
598     \fi\fi\fi\fi
599 \relax
600 \or
601   \global\let\filehook@InputIfFileExists\
        filehook@default@InputIfFileExists
602   \global\let\filehook@@InputIfFileExists\
        filehook@@default@InputIfFileExists
603   \global\let\InputIfFileExists\
        filehook@InputIfFileExists
604   \PackageInfo{filehook-fink}{Package 'fink' detected\
        and replaced by 'currfile'}%
605 \else
606   \iffilehook@force
607     \global\let\filehook@InputIfFileExists\
        filehook@default@InputIfFileExists
608     \global\let\filehook@@InputIfFileExists\
        filehook@@default@InputIfFileExists
609     \global\let\InputIfFileExists\
        filehook@InputIfFileExists
610     \PackageWarning{filehook-fink}{Detected 'fink' \
        package with unknown definition of \string\
        InputIfFileExists.^~J%
611                                     The 'force' option of '\
        filehook' is in \
        effect. Macro is \
        overwritten with \
        default!}%
612 \else
613   \PackageError{filehook-fink}{Detected 'fink' \
        package with unknown definition of \string\
        InputIfFileExists.^~J%
614                                     Use the 'force' \
        option of '\
        filehook' to \
        overwrite it.}{}%
615 \fi
616 \fi
617
618 \endgroup

```

6.5 Support for PGF Keys

```

619 \ProvidesPackage{pgf-filehook}[2010/01/07 v1.0 PGF \
        keys for the filehook package]
620 \RequirePackage{filehook}
621 \RequirePackage{pgfkeys}
622
623 \pgfkeys{%

```

```

624     /filehook/.is family,
625     /filehook,
626 %
627     EveryFile/.is family,
628     EveryFile/AtBegin/.code={\AtBeginOfEveryFile/
        {#1}},
629     EveryFile/AtBegin/.value required,
630     EveryFile/AtEnd/.code={\AtEndOfEveryFile{#1}},
631     EveryFile/AtEnd/.value required,
632 %
633     Files/.is family,
634     Files/AtBegin/.code={\AtBeginOfFiles{#1}},
635     Files/AtBegin/.value required,
636     Files/AtEnd/.code={\AtEndOfFiles{#1}},
637     Files/AtEnd/.value required,
638 %
639     File/.is family,
640     File/AtBegin/.code 2 args={\AtBeginOfFile/
        {#1}{#2}},
641     File/AtBegin/.value required,
642     File/AtEnd/.code 2 args={\AtEndOfFile{#1}{#2}},
643     File/AtEnd/.value required,
644 %
645     Inputs/.is family,
646     Inputs/AtBegin/.code={\AtBeginOfInputs{#1}},
647     Inputs/AtBegin/.value required,
648     Inputs/AtEnd/.code={\AtEndOfInputs{#1}},
649     Inputs/AtEnd/.value required,
650 %
651     InputFile/.is family,
652     InputFile/AtBegin/.code 2 args={\
        AtBeginOfInputFile{#1}{#2}},
653     InputFile/AtBegin/.value required,
654     InputFile/AtEnd/.code 2 args={\AtEndOfInputFile/
        {#1}{#2}},
655     InputFile/AtEnd/.value required,
656 %
657     Includes/.is family,
658     Includes/AtBegin/.code={\AtBeginOfIncludes{#1}},
659     Includes/AtBegin/.value required,
660     Includes/AtEnd/.code={\AtEndOfIncludes{#1}},
661     Includes/AtEnd/.value required,
662     Includes/After/.code={\AfterIncludes{#1}},
663     Includes/After/.value required,
664 %
665     IncludeFile/.is family,
666     IncludeFile/AtBegin/.code 2 args={\
        AtBeginOfIncludeFile{#1}{#2}},
667     IncludeFile/AtBegin/.value required,

```

```

668 IncludeFile/AtEnd/.code 2 args={\
        AtEndOfIncludeFile{#1}{#2}},
669 IncludeFile/AtEnd/.value required,
670 IncludeFile/After/.code 2 args={\AfterIncludeFile/
        {#1}{#2}},
671 IncludeFile/After/.value required,
672 %
673 ClassFile/.is family,
674 ClassFile/AtBegin/.code={\AtBeginOfClassFile#1},
675 ClassFile/AtBegin/.value required,
676 ClassFile/AtEnd/.code={\AtEndOfClassFile#1},
677 ClassFile/AtEnd/.value required,
678 %
679 PackageFile/.is family,
680 PackageFile/AtBegin/.code={\AtBeginOfPackageFile/
        #1},
681 PackageFile/AtBegin/.value required,
682 PackageFile/AtEnd/.code={\AtEndOfPackageFile#1},
683 PackageFile/AtEnd/.value required,
684 }
685 \newcommand{\pgffilehook}{\pgfqkeys{/filehook}}
686

```