

The filehook Package

Martin Scharrer
martin@scharrer-online.de

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

Version v0.8a – 2020/09/29

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 Changes due to new \LaTeX core hooks

With the \LaTeX release 2020/10/01 file hooks are now provided by the \LaTeX core. This package has been adjusted to use these hooks while trying to provide the same interface and behaviour than before. Simpler usages should work without any differences but advanced usages which rely on the exact hook order and position might see some unwanted changes. Users should try to use the \LaTeX hooks instead for new documents. Please see \LaTeX core filehook documentation [ltfilehook-doc](#) for the new hook system.

Support of other hooking systems in other packages and classes has been dropped as this package no longer installs own hooks.

If this package is run under a \LaTeX release prior to 2020/10/01 the old implementation will be loaded. For this switch two sub-packages `filehook-2019` and `filehook-2020` are used and loaded according to the \LaTeX release version. Please do not load these packages directly as they might be changes or disappear on later releases.

2 Introduction

These package (under \LaTeX prior 2020/10/01) 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{ \TeX code}`.

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 6 for information how to upgrade older documents.

3 Usage

The below macros can be used to add material (T_EX 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

```
\AtBeginOfEveryFile{<TEX code>}  
\AtEndOfEveryFile{<TEX code>}
```

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 `{<code>}` 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 `<file name>` 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.

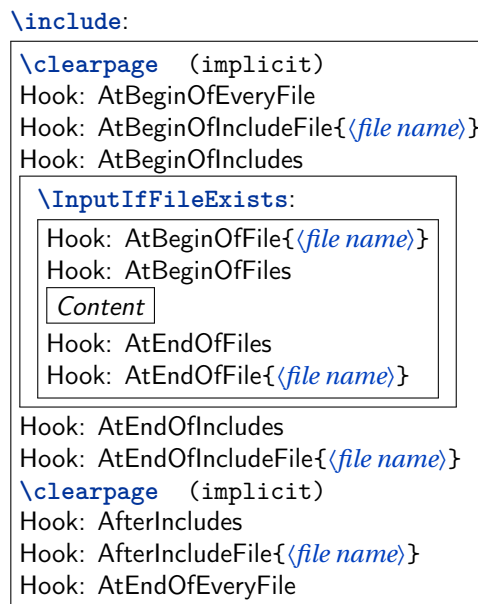
```

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

```

These file-specific macros take the two arguments. The *<code>* is only executed for the file with the given *<file name>* and only if it is read using `\include`. The *<file name>* 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>}

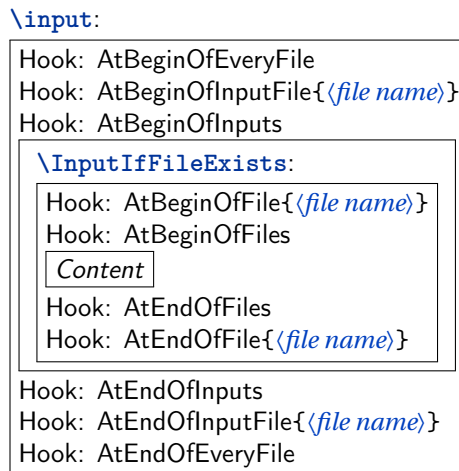
```

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:

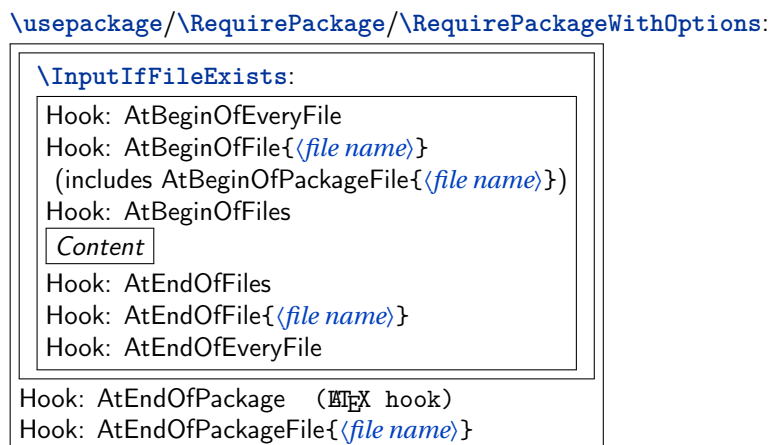


Package Files

```
\AtBeginOfPackageFile*{<package name>}{<TEX code>}
\AtEndOfPackageFile*{<package name>}{<TEX 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:



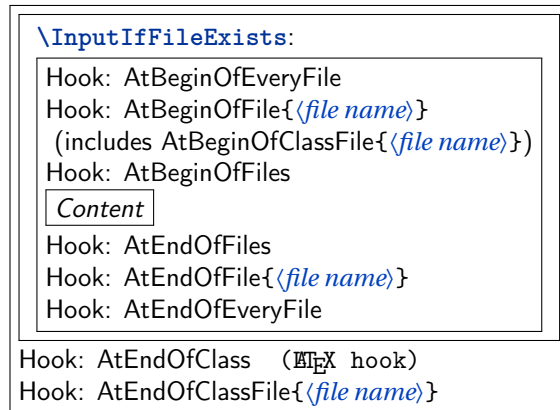
Class Files

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

This macros install the given *<TeX 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:`



3.1 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 5 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>}
```

4 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`).

5 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.

5.1 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.

5.2 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{...}`.

6 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.

7 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]
```

7.1 Options

```
10 \DeclareOption{force}{\PassOptionsToPackage{force}{/
    filehook-2019}}
11 \ProcessOptions\relax
```

7.2 Load actual package

```
12 \@ifl@t@r\fmtversion{2020/10/01}{\RequirePackage{/
    filehook-2020}}{\RequirePackage{filehook-2019}}

13 %<!COPYRIGHT>
14 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
15 \ProvidesPackage{filehook-2019}[% filehook subpackage/
    , do not load directly
16 %<!DATE>
17 %<!VERSION>
18 %<*DRIVER>
19     2099/01/01 develop
20 %</DRIVER>
21     Hooks for input files]
```

7.3 Options

```
22 \newif\iffilehook@force
23 \DeclareOption{force}{\filehook@forcetrue}
24 \ProcessOptions\relax
```

7.4 General stuff

`\iffilehook@newfmt`

```
25 \newif\iffilehook@newfmt
26 \@ifl@t@r\fmtversion{2019/10/01}{\filehook@newfmttrue/
    }\filehook@newfmtfalse}
```

`\filehook@let`

```
    #1: <macro name 1>
    #2: <macro name 2>
27 \def\filehook@let#1#2{%
28   \expandafter\ifx\csname #2\space\endcsname\relax
29   \expandafter\let\csname #1\expandafter\endcsname\
    csname #2\endcsname
30   \else
31   \expandafter\def\csname #1\expandafter\endcsname\
    expandafter{\expandafter\protect\csname #1\
    space\endcsname}%
32   \expandafter\let\csname #1\space\expandafter\
    endcsname\csname #2\space\endcsname
33   \fi
34 }
```

`\filehook@glet`

```
    #1: <macro name 1>
    #2: <macro name 2>
35 \def\filehook@glet#1#2{%
36   \expandafter\ifx\csname #2\space\endcsname\relax
37   \expandafter\global\expandafter\let\csname #1\
    expandafter\endcsname\csname #2\endcsname
38   \else
39   \expandafter\global\expandafter\def\csname #1\
    expandafter\endcsname\expandafter{\expandafter\
    \protect\csname #1\space\endcsname}%
40   \expandafter\global\expandafter\let\csname #1\
    space\expandafter\endcsname\csname #2\space\
    endcsname
41   \fi
42 }
```

`\filehook@cmp`

```
    #1: <macro name 1>
    #2: <macro name 2>
Compare two macros definition including its space form in case of robust macros.
```

```

43 \def\filehook@cmp#1#2{%
44   \expandafter\ifx\csname #2\space\endcsname\relax
45   \expandafter\ifx\csname #1\expandafter\endcsname\
      csname #2\endcsname
46   \expandafter\expandafter\expandafter\
      @firstoftwo
47   \else
48   \expandafter\expandafter\expandafter\
      @secondoftwo
49   \fi
50   \else
51   \expandafter\ifx\csname #1\space\expandafter\
      endcsname\csname #2\space\endcsname
52   \expandafter\expandafter\expandafter\
      @firstoftwo
53   \else
54   \expandafter\expandafter\expandafter\
      @secondoftwo
55   \fi
56   \fi
57 }

```

7.5 Initialisation of Hooks

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

`\filehook@csuse`

```

58 \begingroup
59 \gdef\filehook@csuse#1{\ifcsname #1\endcsname\csname /
      #1\expandafter\endcsname\fi}
60 \expandafter\ifx\csname csuse\endcsname\relax
61   \expandafter\ifx\csname ifcsname\endcsname\relax
62   \gdef\filehook@csuse#1{\expandafter\ifx\
      csname #1\endcsname\relax\else\csname #1\
      expandafter\endcsname\fi}
63   \fi
64 \else
65   \global\let\filehook@csuse\csuse
66 \fi
67 \endgroup

```

`\filehook@include@atbegin`

```

68 \def\filehook@include@atbegin#1{%
69   \filehook@let{InputIfFileExists}{\
      filehook@@InputIfFileExists}%

```

```

70 \filehook@csuse{\filehook@include@atbegin@#1}%
71 \filehook@include@@atbegin
72 }

```

\filehook@include@@atbegin

```

73 \def\filehook@include@@atbegin{}

```

\filehook@include@atend

```

74 \def\filehook@include@atend#1{%
75 \filehook@include@@atend
76 \filehook@csuse{\filehook@include@atend@#1}%
77 }

```

\filehook@include@@atend

```

78 \def\filehook@include@@atend{}

```

\filehook@include@after

```

79 \def\filehook@include@after#1{%
80 \filehook@include@@after
81 \filehook@csuse{\filehook@include@after@#1}%
82 }

```

\filehook@include@@after

```

83 \def\filehook@include@@after{}

```

\filehook@input@atbegin

```

84 \def\filehook@input@atbegin#1{%
85 \filehook@let{InputIfFileExists}{\
      filehook@@InputIfFileExists}%
86 \filehook@csuse{\filehook@input@atbegin@\
      filehook@ensureext{#1}}%
87 \filehook@input@@atbegin
88 }

```

`\filehook@input@@atbegin`

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

`\filehook@input@atend`

```
90 \def\filehook@input@atend#1{%  
91   \filehook@input@@atend  
92   \filehook@csuse{\filehook@input@atend@\  
     filehook@ensureext{#1}}%  
93 }
```

`\filehook@input@@atend`

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

`\filehook@atbegin`

```
95 \def\filehook@atbegin#1{%  
96   \filehook@csuse{\filehook@atbegin@\  
     filehook@ensureext{#1}}%  
97   \filehook@@@atbegin  
98 }
```

`\filehook@@@atbegin`

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

`\filehook@atend`

```
100 \def\filehook@atend#1{%  
101   \filehook@@@atend  
102   \filehook@csuse{\filehook@atend@\  
     filehook@ensureext/  
     {#1}}%  
103 }
```

`\filehook@@@atend`

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

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

```
105 \def\filehook@every@atbegin#1{%  
106     \filehook@every@@atbegin  
107 }
```

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

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

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

```
109 \def\filehook@every@atend#1{%  
110     \filehook@every@@atend  
111 }
```

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

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

7.6 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.

```
113 \def\filehook@include@atbegin@{/  
    filehook@include@atbegin@}  
114 \def\filehook@include@atend@{filehook@include@atend@}  
115 \def\filehook@include@after@{filehook@include@after@}  
116 \def\filehook@input@atbegin@{filehook@input@atbegin@}  
117 \def\filehook@input@atend@{filehook@input@atend@}  
118 \def\filehook@input@after@{filehook@input@after@}  
119 \def\filehook@atbegin@{filehook@atbegin@}  
120 \def\filehook@atend@{filehook@atend@}  
121 \def\filehook@after@{filehook@after@}
```


`\filehook@append`

Uses default \LaTeX macro.

```
122 \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.

```
123 \long\def\filehook@appendwarg#1#2{%
124   \begingroup
125   \toks@\expandafter{#1{##1}#2}%
126   \edef\@tempa{\the\toks@}%
127   \expandafter\gdef\expandafter#1\expandafter##\
      expandafter1\expandafter{\@tempa}%
128   \endgroup
129 }
```

`\filehook@prefix`

Prefixes code to a hook.

```
130 \long\def\filehook@prefix#1#2{%
131   \begingroup
132   \@temptokena{#2}%
133   \toks@\expandafter{#1}%
134   \xdef#1{\the\@temptokena\the\toks@}%
135   \endgroup
136 }
```

`\filehook@prefixwarg`

Prefixes code with an argument to a hook.

```
137 \long\def\filehook@prefixwarg#1#2{%
138   \begingroup
139   \@temptokena{#2}%
140   \toks@\expandafter{#1{##1}}%
141   \edef\@tempa{\the\@temptokena\the\toks@}%
142   \expandafter\gdef\expandafter#1\expandafter##\
      expandafter1\expandafter{\@tempa}%
143   \endgroup
144 }
```

`\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.

```
145 \def\filehook@addtohook#1#2#3{%
146   \begingroup
147   \edef\@tempa{#3}%
148   \edef\@tempa{#2\filehook@ensureext{\@tempa}}%
149   \@ifundefined{\@tempa}{\global\@namedef{\@tempa/
      }{}}{}%
150   \expandafter\endgroup
151   \expandafter#1\csname\@tempa\endcsname
152 }
```

User Level Macros

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

`\AtBeginOfIncludes`

```
153 \newcommand*\AtBeginOfIncludes{%
154   \filehook@append\filehook@include@@atbegin
155 }
```

`\AtEndOfIncludes`

```
156 \newcommand*\AtEndOfIncludes{%
157   \filehook@prefix\filehook@include@@atend
158 }
```

`\AfterIncludes`

```
159 \newcommand*\AfterIncludes{%
160   \filehook@prefix\filehook@include@@after
161 }
```

\AtBeginOfIncludeFile

```
162 \newcommand*\AtBeginOfIncludeFile [1]{%
163   \filehook@addtohook\filehook@append\
      filehook@include@atbegin@{\filehook@ensuretex/
      {#1}}%
164 }
```

\AtEndOfIncludeFile

```
165 \newcommand*\AtEndOfIncludeFile [1]{%
166   \filehook@addtohook\filehook@prefix\
      filehook@include@atend@{\filehook@ensuretex{#1}}\
      %
167 }
```

\AfterIncludeFile

```
168 \newcommand*\AfterIncludeFile [1]{%
169   \filehook@addtohook\filehook@prefix\
      filehook@include@after@{\filehook@ensuretex{#1}}\
      %
170 }
```

\AtBeginOfInputs

```
171 \newcommand*\AtBeginOfInputs{%
172   \filehook@append\filehook@input@@atbegin
173 }
```

\AtEndOfInputs

```
174 \newcommand*\AtEndOfInputs{%
175   \filehook@prefix\filehook@input@@atend
176 }
```

\AtBeginOfInputFile

```
177 \newcommand*\AtBeginOfInputFile{%
178   \filehook@addtohook\filehook@append\
      filehook@input@atbegin@
179 }
```

\AtEndOfInputFile

```
180 \newcommand*\AtEndOfInputFile{%  
181   \filehook@addtohook\filehook@prefix\  
      filehook@input@atend@  
182 }
```

\AtBeginOfFiles

```
183 \newcommand*\AtBeginOfFiles{%  
184   \filehook@append\filehook@@atbegin  
185 }
```

\AtEndOfFiles

```
186 \newcommand*\AtEndOfFiles{%  
187   \filehook@prefix\filehook@@atend  
188 }
```

\AtBeginOfEveryFile

```
189 \newcommand*\AtBeginOfEveryFile{%  
190   \filehook@append\filehook@every@@atbegin  
191 }
```

\AtEndOfEveryFile

```
192 \newcommand*\AtEndOfEveryFile{%  
193   \filehook@prefix\filehook@every@@atend  
194 }
```

\AtBeginOfFile

```
195 \newcommand*\AtBeginOfFile{%  
196   \filehook@addtohook\filehook@append\  
      filehook@atbegin@  
197 }
```

\AtEndOfFile

```
198 \newcommand*\AtEndOfFile{%
199   \filehook@addtohook\filehook@prefix\filehook@atend@
200 }
```

\AtBeginOfClassFile

```
201 \newcommand*\AtBeginOfClassFile{%
202   \@ifnextchar*
203     {\AtBeginOfXFile@star\@clsextension}%
204     {\AtBeginOfXFile@normal\@clsextension}%
205 }
```

\AtBeginOfPackageFile

```
206 \newcommand*\AtBeginOfPackageFile{%
207   \@ifnextchar*
208     {\AtBeginOfXFile@star\@pkgextension}%
209     {\AtBeginOfXFile@normal\@pkgextension}%
210 }
```

\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.

```
211 \def\AtBeginOfXFile@star#1*#2{%
212   \@ifl@aded{#1}{#2}%
213   {\@firstofone}%
214   {\AtBeginOfXFile@normal{#1}{#2}}%
215 }
```

\AtBeginOfXFile@normal

#1: extension

#2: name

```
216 \def\AtBeginOfXFile@normal#1#2{%
217   \AtBeginOfFile{#2.#1}%
218 }
```

`\AtEndOfClassFile`

```
219 \newcommand*\AtEndOfClassFile{%
220     \@ifnextchar*
221         {\AtEndOfXFile@star\@clsextension}%
222         {\AtEndOfXFile@normal\@clsextension}%
223 }
```

`\AtEndOfPackageFile`

```
224 \newcommand*\AtEndOfPackageFile{%
225     \@ifnextchar*
226         {\AtEndOfXFile@star\@pkgextension}%
227         {\AtEndOfXFile@normal\@pkgextension}%
228 }
```

`\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.

```
229 \def\AtEndOfXFile@star#1*#2{%
230     \@ifloaded{#1}{#2}%
231     {\@firstofone}%
232     {\AtEndOfXFile@normal{#1}{#2}}%
233 }
```

`\AtEndOfXFile@normal`

#1: extension

#2: name

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

```
234 \long\def\AtEndOfXFile@normal#1#2#3{%
235     \AtEndOfFile{#2.#1}{\AtEndOfPackage{#3}}%
236 }
```

`\ClearHook`

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

```

237 \newcommand*\ClearHook{%
238     \begingroup
239     \def\filehook@prefix##1##2{%
240         \gdef##1{}%
241         \endgroup
242     }%
243     \let\filehook@append\filehook@prefix
244 }

```

7.7 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@
```

```
245 \let\filehook@orig@@input@\@input@
```

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

```
246 \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.

```

247 \def\@input@#1{%
248     \@ifnextchar\clearpage
249     {%
250         \filehook@every@atbegin{#1}%
251         \filehook@include@atbegin{#1}%
252         \filehook@orig@@input@{#1}%
253         \filehook@include@atend{#1}%
254         \clearpage
255         \filehook@include@after{#1}%
256         \filehook@every@atend{#1}%
257         \@gobble
258     }%
259     {\filehook@orig@@input@{#1}}%
260 }

```

`\@iinput`

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

```
261 \def\filehook@@iinput#1{%
262   \filehook@every@atbegin{#1}%
263   \filehook@input@atbegin{#1}%
264   \filehook@orig@@iinput{#1}%
265   \filehook@input@atend{#1}%
266   \filehook@every@atend{#1}%
267 }
268 \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.

```
269 \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.

```
270 \def\filehook@ensureext#1{%
271   \expandafter\filehook@@ensureext#1\empty.tex\
      empty\empty
272 }
```

`\filehook@@ensureext`

```
273 \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.

```
274 \def\filehook@ensuret看#1{%
275   \expandafter\filehook@@ensuret看#1\empty.tex\
      empty\empty
276 }
```


`\filehook@@ensuretex`

```
277 \def\filehook@@ensuretex#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.

`\latex@InputIfFileExists`

Standard \TeX definition of `\InputIfFileExists`.

```
278 \iffilehook@newfmt
279 \expandafter\def\expandafter\latex@InputIfFileExists\l/
    \expandafter{%
280   \expandafter\protect\csname InputIfFileExists\space/
    \endcsname
281 }
282 \expandafter\long\expandafter\def\csname /
    latex@InputIfFileExists\space\endcsname#1#2{%
283   \IfFileExists{#1}%
284   {%
285     \expandafter\@swaptwoargs\expandafter
286     {\@filef@und}{#2\@addtofilelist{#1}\@input}}
287   \else
288   \long\def\latex@InputIfFileExists#1#2{%
289     \IfFileExists{#1}%
290     {#2\@addtofilelist{#1}%
291     \@input\@filef@und
292     }%
293   }
294 \fi
```

`\filehook@default@InputIfFileExists`

```
295 \DeclareRobustCommand\l/
    filehook@default@InputIfFileExists [2]{%
296   \IfFileExists{#1}%
297   {\expandafter\filehook@swap
298    \expandafter{\@filef@und}%
299    {#2\@addtofilelist{#1}%
300    \filehook@every@atbegin{#1}%
301    \filehook@atbegin{#1}%
302    \@input}%
303    \filehook@atend{#1}%
304    \filehook@every@atend{#1}%
```

```

305     }%
306 }

```

Make sure definition is global:

```

307 \filehook@glet{filehook@default@InputIfFileExists}{\
    filehook@default@InputIfFileExists}%

```

`\filehook@@default@InputIfFileExists`

```

308 \DeclareRobustCommand\
    filehook@@default@InputIfFileExists [2]{%
309 \filehook@let{InputIfFileExists}{\
    filehook@InputIfFileExists}%
310 \IfFileExists{#1}%
311     {\expandafter\filehook@swap
312     \expandafter{\@filef@und}%
313     {#2\@addtofilelist{#1}%
314     \filehook@atbegin{#1}%
315     \@@input}%
316     \filehook@atend{#1}%
317     }%
318 }

```

Make sure definition is global:

```

319 \filehook@glet{filehook@@default@InputIfFileExists}{\
    filehook@@default@InputIfFileExists}%

```

`\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.

```

320 \AtBeginOfPackageFile{scrfile}{%
321     \filehook@glet{InputIfFileExists}{\
        latex@InputIfFileExists}%
322 }%
323 \AtEndOfPackageFile*{scrfile}{%
324     \RequirePackage{filehook-scrfile}%
325 }%

```

Fink:

```

326 \AtBeginOfPackageFile*{fink}{%
327     \RequirePackage{kvoptions}%
328     \begingroup
329     \filehook@let{InputIfFileExists}{\
        latex@InputIfFileExists}%
330 }%

```

```

331 \AtEndOfPackageFile*{fink}{%
332   \edef\@tempa{\noexpand\PassOptionsToPackage{/
      mainext=\fnk@mainext ,maindir=\fnk@maindir}{/
      currfile}}%
333   \expandafter\endgroup\@tempa
334   \RequirePackage{filehook-fink}%
335 }%

```

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.

```

336 \AtBeginOfClassFile{memoir}{%
337   \filehook@let{InputIfFileExists}{/
      latex@InputIfFileExists}%
338   \let\@iinput\filehook@orig@@iinput
339 }%
340 \AtEndOfClassFile*{memoir}{%
341   \let\@iinput\filehook@@iinput
342   \RequirePackage{filehook-memoir}%
343 }%

```

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.

```

344 \filehook@cmp{InputIfFileExists}{/
      filehook@InputIfFileExists}%
345 {}% already set up
346 {%
347   \filehook@cmp{InputIfFileExists}{/
      latex@InputIfFileExists}%
348   {%
349     \filehook@let{filehook@InputIfFileExists}{/
      filehook@default@InputIfFileExists}%
      \filehook@let{filehook@@InputIfFileExists}{/
      filehook@@default@InputIfFileExists}%
350     \filehook@let{InputIfFileExists}{/
      filehook@InputIfFileExists}%
351   }%
352   {%
353     \iffilehook@force
354     \filehook@let{filehook@InputIfFileExists}{/
      filehook@default@InputIfFileExists}%
      \filehook@let{filehook@@InputIfFileExists}{/
      filehook@@default@InputIfFileExists}%
355     \filehook@let{InputIfFileExists}{/
      filehook@InputIfFileExists}%
356     \PackageWarning{filehook}{Detected unknown /
      definition of \string\InputIfFileExists/
      .^^J%

```

```

359             The 'force' /
                option of ' /
                filehook' is /
                in effect. /
                Macro is /
                overwritten /
                with default!} /
                %
360     \else
361         \PackageError{filehook}{Detected unknown /
                definition of \string\InputIfFileExists /
                .^^J%
362             Use the 'force' /
                option of ' /
                filehook' to /
                overwrite it.}{} /
                %
363     \fi
364     }%
365 }%

366 \AtBeginDocument{%
367     % Check if definition got changed again. For the /
        new LaTeX format we check again \ /
        InputIfFileExists<space>,
368     % for the old format to \InputIfFileExists /
        directly.
369     \filehook@cmp{InputIfFileExists}{ /
        filehook@InputIfFileExists}{}{%
370     \PackageWarning{filehook}{Macro \string\ /
        InputIfFileExists\space got redefined /
        after 'filehook' was loaded.^^J%
371             Certain file hooks /
                might now be /
                dysfunctional!}%
372     }%
373 }

374 %<!COPYRIGHT>
375 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
376 \ProvidesPackage{filehook-2020}[% filehook subpackage /
        , do not load directly
377 %<!DATE>
378 %<!VERSION>
379 %< *DRIVER>
380     2099/01/01 develop
381 %</DRIVER>
382     Hooks for input files]

```

7.8 Options

```
383 \DeclareOption{force}{}
384 \ProcessOptions\relax
```

7.9 General stuff

7.10 Initialisation of Hooks

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

`\filehook@csuse`

```
385 \begingroup
386 \gdef\filehook@csuse#1{\ifcsname #1\endcsname\csname /
#1\expandafter\endcsname\fi}
387 \expandafter\ifx\csname csuse\endcsname\relax
388   \expandafter\ifx\csname ifcsname\endcsname\relax
389     \gdef\filehook@csuse#1{\expandafter\ifx\
csname #1\endcsname\relax\else\csname #1\
expandafter\endcsname\fi}
390   \fi
391 \else
392   \global\let\filehook@csuse\csuse
393 \fi
394 \endgroup
```

`\filehook@include@atbegin`

```
395 \def\filehook@include@atbegin#1{%
396   \filehook@csuse{\filehook@include@atbegin@#1}%
397   \filehook@include@@atbegin
398 }
```

`\filehook@include@@atbegin`

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

`\filehook@include@atend`

```
400 \def\filehook@include@atend#1{%
401   \filehook@include@@atend
402   \filehook@csuse{\filehook@include@atend@#1}%
403 }
```

`\filehook@include@@atend`

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

`\filehook@include@after`

```
405 \def\filehook@include@after#1{%  
406   \filehook@include@@after  
407   \filehook@csuse{\filehook@include@after@#1}%  
408 }
```

`\filehook@include@@after`

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

`\filehook@input@atbegin`

```
410 \def\filehook@input@atbegin#1{%  
411   \filehook@csuse{\filehook@input@atbegin@\  
         filehook@ensureext{#1}}%  
412   \filehook@input@@atbegin  
413 }
```

`\filehook@input@@atbegin`

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

`\filehook@input@atend`

```
415 \def\filehook@input@atend#1{%  
416   \filehook@input@@atend  
417   \filehook@csuse{\filehook@input@atend@\  
         filehook@ensureext{#1}}%  
418 }
```

`\filehook@input@@atend`

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

`\filehook@atbegin`

```
420 \def\filehook@atbegin#1{%  
421   \filehook@csuse{\filehook@atbegin@\  
       filehook@ensureext{#1}}%  
422   \filehook@@atbegin  
423 }
```

`\filehook@@atbegin`

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

`\filehook@atend`

```
425 \def\filehook@atend#1{%  
426   \filehook@@atend  
427   \filehook@csuse{\filehook@atend@\  
       filehook@ensureext/  
       {#1}}%  
428 }
```

`\filehook@@atend`

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

`\filehook@every@atbegin`

```
430 \def\filehook@every@atbegin#1{%  
431   \filehook@every@@atbegin  
432 }
```

`\filehook@every@@atbegin`

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

`\filehook@every@atend`

```
434 \def\filehook@every@atend#1{%  
435   \filehook@every@@atend  
436 }
```

`\filehook@every@@atend`

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

7.11 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.

```
438 \def\filehook@include@atbegin@{\  
    filehook@include@atbegin@}  
439 \def\filehook@include@atend@{filehook@include@atend@}  
440 \def\filehook@include@after@{filehook@include@after@}  
441 \def\filehook@input@atbegin@{filehook@input@atbegin@}  
442 \def\filehook@input@atend@{filehook@input@atend@}  
443 \def\filehook@input@after@{filehook@input@after@}  
444 \def\filehook@atbegin@{filehook@atbegin@}  
445 \def\filehook@atend@{filehook@atend@}  
446 \def\filehook@after@{filehook@after@}
```

`\filehook@append`

Uses default \LaTeX macro.

```
447 \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.

```
448 \long\def\filehook@appendwarg#1#2{%  
449   \begingroup  
450     \toks@\expandafter{#1{##1}#2}%  
451     \edef\@tempa{\the\toks@}%  
452     \expandafter\gdef\expandafter#1\expandafter##\  
         expandafter1\expandafter{\@tempa}%  
453   \endgroup  
454 }
```


`\filehook@prefix`

Prefixes code to a hook.

```
455 \long\def\filehook@prefix#1#2{%
456   \begingroup
457     \@temptokena{#2}%
458     \toks@\expandafter{#1}%
459     \xdef#1{\the\@temptokena\the\toks@}%
460   \endgroup
461 }
```

`\filehook@prefixwarg`

Prefixes code with an argument to a hook.

```
462 \long\def\filehook@prefixwarg#1#2{%
463   \begingroup
464     \@temptokena{#2}%
465     \toks@\expandafter{#1{##1}}%
466     \edef\@tempa{\the\@temptokena\the\toks@}%
467     \expandafter\gdef\expandafter#1\expandafter##\
468       expandafter1\expandafter{\@tempa}%
469   \endgroup
470 }
```

`\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.

```
470 \def\filehook@addtohook#1#2#3{%
471   \begingroup
472     \edef\@tempa{#3}%
473     \edef\@tempa{#2\filehook@ensureext{\@tempa}}%
474     \@ifundefined{\@tempa}{\global\@namedef{\@tempa/
475       }{}}{}%
476     \expandafter\endgroup
477     \expandafter#1\csname\@tempa\endcsname
478 }
```

User Level Macros

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

\AtBeginOfIncludes

```
478 \newcommand*\AtBeginOfIncludes{%  
479   \filehook@append\filehook@include@@atbegin  
480 }
```

\AtEndOfIncludes

```
481 \newcommand*\AtEndOfIncludes{%  
482   \filehook@prefix\filehook@include@@atend  
483 }
```

\AfterIncludes

```
484 \newcommand*\AfterIncludes{%  
485   \filehook@prefix\filehook@include@@after  
486 }
```

\AtBeginOfIncludeFile

```
487 \newcommand*\AtBeginOfIncludeFile [1]{%  
488   \filehook@addtohook\filehook@append\  
         filehook@include@atbegin@{\filehook@ensuretex/  
         {#1}}%  
489 }
```

\AtEndOfIncludeFile

```
490 \newcommand*\AtEndOfIncludeFile [1]{%  
491   \filehook@addtohook\filehook@prefix\  
         filehook@include@atend@{\filehook@ensuretex{#1}}\  
         %  
492 }
```

\AfterIncludeFile

```
493 \newcommand*\AfterIncludeFile [1]{%  
494   \filehook@addtohook\filehook@prefix\  
         filehook@include@after@{\filehook@ensuretex{#1}}\  
         %  
495 }
```

\AtBeginOfInputs

```
496 \newcommand*\AtBeginOfInputs{%  
497   \filehook@append\filehook@input@@atbegin  
498 }
```

\AtEndOfInputs

```
499 \newcommand*\AtEndOfInputs{%  
500   \filehook@prefix\filehook@input@@atend  
501 }
```

\AtBeginOfInputFile

```
502 \newcommand*\AtBeginOfInputFile{%  
503   \filehook@addtohook\filehook@append\  
    filehook@input@atbegin@  
504 }
```

\AtEndOfInputFile

```
505 \newcommand*\AtEndOfInputFile{%  
506   \filehook@addtohook\filehook@prefix\  
    filehook@input@atend@  
507 }
```

\AtBeginOfFiles

```
508 \newcommand*\AtBeginOfFiles{%  
509   \filehook@append\filehook@@atbegin  
510 }
```

\AtEndOfFiles

```
511 \newcommand*\AtEndOfFiles{%  
512   \filehook@prefix\filehook@@atend  
513 }
```

`\AtBeginOfEveryFile`

```
514 \newcommand*\AtBeginOfEveryFile{%  
515   \filehook@append\filehook@every@@atbegin  
516 }
```

`\AtEndOfEveryFile`

```
517 \newcommand*\AtEndOfEveryFile{%  
518   \filehook@prefix\filehook@every@@atend  
519 }
```

`\AtBeginOfFile`

```
520 \newcommand*\AtBeginOfFile{%  
521   \filehook@addtohook\filehook@append\  
522     filehook@atbegin@  
523 }
```

`\AtEndOfFile`

```
524 \newcommand*\AtEndOfFile{%  
525   \filehook@addtohook\filehook@prefix\filehook@atend@  
526 }
```

`\AtBeginOfClassFile`

```
527 \newcommand*\AtBeginOfClassFile{%  
528   \@ifnextchar*  
529     {\AtBeginOfXFile@star\@clsextension}%  
530     {\AtBeginOfXFile@normal\@clsextension}%  
531 }
```

`\AtBeginOfPackageFile`

```
532 \newcommand*\AtBeginOfPackageFile{%  
533   \@ifnextchar*  
534     {\AtBeginOfXFile@star\@pkgextension}%  
535     {\AtBeginOfXFile@normal\@pkgextension}%  
536 }
```

`\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.

```
536 \def\AtBeginOfXFile@star#1*#2{%  
537   \@ifl@aded{#1}{#2}%  
538     {\@firstofone}%  
539     {\AtBeginOfXFile@normal{#1}{#2}}%  
540 }
```

`\AtBeginOfXFile@normal`

#1: extension
#2: name

```
541 \def\AtBeginOfXFile@normal#1#2{%  
542   \AtBeginOfFile{#2.#1}%  
543 }
```

`\AtEndOfClassFile`

```
544 \newcommand*\AtEndOfClassFile{%  
545   \@ifnextchar*  
546     {\AtEndOfXFile@star\@clsextension}%  
547     {\AtEndOfXFile@normal\@clsextension}%  
548 }
```

`\AtEndOfPackageFile`

```
549 \newcommand*\AtEndOfPackageFile{%  
550   \@ifnextchar*  
551     {\AtEndOfXFile@star\@pkgextension}%  
552     {\AtEndOfXFile@normal\@pkgextension}%  
553 }
```

`\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.

```

554 \def\AtEndOfXFile@star#1*#2{%
555     \@ifl@aded{#1}{#2}%
556     {\@firstofone}%
557     {\AtEndOfXFile@normal{#1}{#2}}%
558 }

```

\AtEndOfXFile@normal

#1: extension

#2: name

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

```

559 \long\def\AtEndOfXFile@normal#1#2#3{%
560     \AtEndOfFile{#2.#1}{\AtEndOfPackage{#3}}%
561 }

```

\ClearHook

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

```

562 \newcommand*\ClearHook{%
563     \begingroup
564     \def\filehook@prefix##1##2{%
565         \gdef##1{}%
566         \endgroup
567     }%
568     \let\filehook@append\filehook@prefix
569 }

```

7.12 Installation of Hooks

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

\@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.

```

570 \def\DEPRECATED@input@#1{%
571     \@ifnextchar\clearpage
572     {%
573         \filehook@every@atbegin{#1}%
574         \filehook@include@atbegin{#1}%

```

```

575     \filehook@orig@@input@{#1}%
576     \filehook@include@atend{#1}%
577     \clearpage
578     \filehook@include@after{#1}%
579     \filehook@every@atend{#1}%
580     \@gobble
581   }%
582   {\filehook@orig@@input@{#1}}%
583 }

```

`\@iinput`

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

```

584 \def\filehook@@iinput#1{%
585   \filehook@every@atbegin{#1}%
586   \filehook@input@atbegin{#1}%
587   \filehook@orig@@iinput{#1}%
588   \filehook@input@atend{#1}%
589   \filehook@every@atend{#1}%
590 }
591 %\let\@iinput\filehook@@iinput

```

`\filehook@ensureext`

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

```

592 \def\filehook@ensureext#1{%
593   \expandafter\filehook@@ensureext#1\empty.tex\
594     empty\empty
595 }

```

`\filehook@@ensureext`

```

595 \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.

```

596 \def\filehook@ensuret看#1{%
597   \expandafter\filehook@@ensuret看#1\empty.tex\
598     empty\empty
599 }

```

`\filehook@@ensuretex`

```
599 \def\filehook@@ensuretex#1.tex\empty#2\empty{#1.tex}
```

`\filehook@set@CurrentFile`

```
600 \def\filehook@set@CurrentFile{%  
601   \edef\filehook@CurrentFile{%  
602     \ifx\CurrentFilePath\empty  
603     \else  
604       \CurrentFilePath/%  
605     \fi  
606     \CurrentFile  
607   }%  
608 }
```

`\Hook`

```
609 \AddToHook{include/before}{%  
610   \filehook@set@CurrentFile  
611   \filehook@include@atbegin{\filehook@CurrentFile}%  
612 }
```

`\Hook`

```
613 \AddToHook{include/end}{%  
614   \filehook@set@CurrentFile  
615   \filehook@include@atend{\filehook@CurrentFile}%  
616 }
```

`\Hook`

```
617 \AddToHook{include/after}{%  
618   \filehook@set@CurrentFile  
619   \filehook@include@after{\filehook@CurrentFile}%  
620 }
```


`\filehook@istexfile`

```
621 \begingroup
622 \edef\dottex{\expandafter\expandafter\expandafter\
  @gobble\expandafter\string\csname.tex\endcsname}
623 \expandafter
624 \gdef\expandafter\filehook@istexfile\expandafter#\
  expandafter1\expandafter{%
625   \expandafter\expandafter\expandafter\
    filehook@istexfile@\expandafter#\expandafter1\
    expandafter\empty\dottex\empty\empty\@nil
626 }
```

`\filehook@istexfile@`

```
627 \expandafter\gdef\expandafter\filehook@istexfile@\
  expandafter#\expandafter1\dottex\empty#2\empty#3\
  @nil{%
628   \begingroup
629   \def\@tempa{#2}%
630   \ifx\@tempa\empty
631     \endgroup
632     \expandafter\@secondoftwo
633   \else
634     \endgroup
635     \expandafter\@firstoftwo
636   \fi
637 }
638 \endgroup
```

`\Hook`

```
639 \AddToHook{file/before}{%
640   \filehook@set@CurrentFile
641   \filehook@every@atbegin{\filehook@CurrentFile}%
642   \filehook@istexfile\filehook@CurrentFile{\
    filehook@input@atbegin{\filehook@CurrentFile\
    }}}%
643   \filehook@atbegin{\filehook@CurrentFile}%
644 }
```

`\Hook`

```

645 \AddToHook{file/after}{%
646     \filehook@set@CurrentFile
647     \filehook@atend{\filehook@CurrentFile}%
648     \filehook@istexfile\filehook@CurrentFile{\filehook@input@atend{\filehook@CurrentFile}}{}/
649     \filehook@every@atend{\filehook@CurrentFile}%
650 }

651 %<!COPYRIGHT>
652 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
653 \ProvidesPackage{filehook-memoir}[2020/02/02 v0.2 /
        filehook patch for memoir class]

654 \RequirePackage{filehook}
655 \begingroup

```

\memoir@InputIfFileExists

The definition taken from memoir.cls. Copyright see there.

```

656 \ifcsname InputIfFileExists\space\endcsname
657   \DeclareRobustCommand \memoir@InputIfFileExists [2]{/
        %
658   \IfFileExists{#1}%
659   {%
660     \expandafter\@swaptwoargs\expandafter
661     {\@filef@und\m@matendf{#1}\killm@matf{#1}}{%
662       #2\@addtofilelist{#1}\m@matbeginf{#1}\@input/
663       }%
664     }%
665   }
666 \else
667   % Old definition
668 \renewcommand{\memoir@InputIfFileExists}[2]{%
669   \IfFileExists{#1}%
670   {#2\@addtofilelist{#1}\m@matbeginf{#1}%
671     \@input \@filef@und
672     \m@matendf{#1}%
673     \killm@matf{#1}}%
674 }
675 \fi

676 \@tempwafalse
677 \filehook@cmp{InputIfFileExists}{/
        filehook@InputIfFileExists}%
678 {\@tempwatrue}%
679 {%

```

```

680     \filehook@cmp{InputIfFileExists}{\
        memoir@InputIfFileExists}%
681     {\@tempwatrue}%
682     {}%
683 }%
684
685 \if@tempswa
686 \filehook@glet{filehook@InputIfFileExists}{\
        filehook@default@InputIfFileExists}%
687 \filehook@glet{filehook@@InputIfFileExists}{\
        filehook@@default@InputIfFileExists}%
688 \filehook@glet{InputIfFileExists}{\
        filehook@InputIfFileExists}%
689 \filehook@appendwarg\filehook@atbegin{\m@matbeginf\
        {#1}}%
690 \filehook@prefixwarg\filehook@atend{\m@matendf{#1}\
        killm@matf{#1}}%
691 \PackageInfo{filehook}{Detected 'memoir' class: the
        memoir hooks will be moved to the 'At...OfFiles\
        ' hooks}
692 \else
693 \iffilehook@force
694 \filehook@glet{filehook@InputIfFileExists}{\
        filehook@default@InputIfFileExists}%
695 \filehook@glet{filehook@@InputIfFileExists}{\
        filehook@@default@InputIfFileExists}%
696 \filehook@glet{InputIfFileExists}{\
        filehook@InputIfFileExists}%
697 \PackageWarning{filehook}{Detected 'memoir' class\
        with unknown definition of \string\
        InputIfFileExists.^~J%
698                                     The 'force' option of '\
        filehook' is in \
        effect. Macro is \
        overwritten with \
        default!}%
699 \else
700 \PackageError{filehook}{Detected 'memoir' class \
        with unknown definition of \string\
        InputIfFileExists.^~J%
701                                     Use the 'force' option of\
        'filehook' to \
        overwrite it.}}%
702 \fi
703 \fi
704 \endgroup
705 %<!COPYRIGHT>
706 \NeedsTeXFormat{LaTeX2e}[1999/12/01]

```

```

707 \ProvidesPackage{filehook-listings}[2011/01/02 v0.1 /
      Patch for listings to avoid hooks for verbatim /
      input files]

708 \begingroup
709
710 \long\def\patch#1\def\lst@next#2#3\endpatch{%
711   \toks@{#2}%
712   \edef\@tempa{\the\toks@}%
713   \def\@tempb{\input{###1}}%
714   \ifx\@tempa\@tempb
715     \gdef\lst@InputListing##1{#1\def\lst@next{\
716       @input{##1}}#3}%
717   \else
718     \PackageWarning{filehook-listings}{To-be-
719       patched code in macro \string\
720       lst@InputListing was not found!}%
721   \fi
722 }
723
724 \@ifundefined{lst@InputListing}{%
725   \PackageWarning{filehook-listings}{To-be-patched /
726     Macro \string\lst@InputListing not found!}%
727 }{}
728
729 \expandafter\patch\lst@InputListing{#1}\endpatch
730
731 \endgroup
732
733 %<!COPYRIGHT>
734 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
735 \ProvidesPackage{filehook-scrfile}[2020/02/02 v0.2 /
736   filehook patch for scrfile package]
737 \RequirePackage{filehook}
738 \begingroup

```

<code>\scrfile@InputIfFileExists</code>

```

733 \expandafter\def\expandafter\
734   scrfile@InputIfFileExists\expandafter{%
735   \expandafter\protect\csname InputIfFileExists\space/
736     \endcsname
737 }
738 \expandafter\long\expandafter\def\csname /
739   scrfile@InputIfFileExists\space\endcsname#1#2{%
740   \begingroup\expandafter\expandafter\expandafter\
741     endgroup
742   \expandafter\ifx\csname #1-@alias\endcsname\relax

```

```

739     \expandafter \@secondoftwo
740 \else
741     \scr@replacefile@msg{\csname #1-@alias\endcsname/
        }{#1}%
742     \expandafter \@firstoftwo
743 \fi
744 {%
745     \expandafter\InputIfFileExists\expandafter{\
        csname
746 #1-@alias\endcsname}{#2}%
747     }%
748     {\IfFileExists{#1}{%
749         \expandafter\scr@input@withhook\expandafter{\
            @filef@und}{#1}{#2}}%
750     }%
751 }

```

<pre>\filehook@scrfile@InputIfFileExists</pre>
--

```

752 \DeclareRobustCommand\
        filehook@scrfile@InputIfFileExists [2]{%
753     \begingroup\expandafter\expandafter\expandafter\
        endgroup
754     \expandafter\ifx\csname #1-@alias\endcsname\relax
755     \expandafter \@secondoftwo
756     \else
757     \scr@replacefile@msg{\csname #1-@alias\endcsname/
        }{#1}%
758     \expandafter \@firstoftwo
759 \fi
760 {%
761     \expandafter\InputIfFileExists\expandafter{\
        csname
762     #1-@alias\endcsname}{#2}%
763     }%
764     {\IfFileExists{#1}{%
765         \expandafter\filehook@swap
766         \expandafter{\@filef@und}%
767         {\scr@load@hook{before}{#1}%
768         #2\@addtofilelist{#1}%
769         \filehook@every@atbegin{#1}%
770         \filehook@atbegin{#1}%
771         \@@input}%
772         \filehook@atend{#1}%
773         \filehook@every@atend{#1}%
774         \scr@load@hook{after}{#1}%
775     }}%
776 }

```

```

777 \filehook@glet{filehook@scrfile@InputIfFileExists}{\
      filehook@scrfile@InputIfFileExists}%

```

`\filehook@@scrfile@InputIfFileExists`

```

778 \DeclareRobustCommand\
      filehook@@scrfile@InputIfFileExists [2]{%
779 \filehook@let{InputIfFileExists}{\
      filehook@InputIfFileExists}%
780 \begingroup\expandafter\expandafter\expandafter\
      endgroup
781 \expandafter\ifx\cename #1-@alias\endcename\relax
782 \expandafter\@secondoftwo
783 \else
784 \scr@replacefile@msg{\cename #1-@alias\endcename\
      }{#1}%
785 \expandafter\@firstoftwo
786 \fi
787 {%
788 \expandafter\InputIfFileExists\expandafter{\
      cename
789 #1-@alias\endcename}{#2}%
790 }%
791 {\IfFileExists{#1}{%
792 \expandafter\filehook@swap
793 \expandafter{\@filef@und}%
794 {\scr@load@hook{before}{#1}%
795 #2\@addtofilelist{#1}%
796 \filehook@atbegin{#1}%
797 \@@input}%
798 \filehook@atend{#1}%
799 \scr@load@hook{after}{#1}%
800 }}%
801 }
802 \filehook@glet{filehook@@scrfile@InputIfFileExists}{\
      filehook@@scrfile@InputIfFileExists}%

```

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.

```

803 \@tempwafalse
804 \filehook@cmp{InputIfFileExists}{\
      filehook@InputIfFileExists}%
805 {\@tempwattrue}%
806 {%
807 \filehook@cmp{InputIfFileExists}{\
      scrfile@InputIfFileExists}%

```

```

808     {\@tempswatrue}%
809     {}%
810 }%
811
812 \if@tempswa
813   \filehook@glet{filehook@InputIfFileExists}{\
814     filehook@scrlfile@InputIfFileExists}%
815   \filehook@glet{filehook@@InputIfFileExists}{\
816     filehook@@scrlfile@InputIfFileExists}%
817   \filehook@glet{InputIfFileExists}{\
818     filehook@InputIfFileExists}%
819   \PackageInfo{filehook}{Package 'scrlfile' detected \
820     and compensated for}%
821 \else
822   \iffilehook@force
823     \filehook@glet{filehook@InputIfFileExists}{\
824       filehook@scrlfile@InputIfFileExists}%
825     \filehook@glet{filehook@@InputIfFileExists}{\
826       filehook@@scrlfile@InputIfFileExists}%
827     \filehook@glet{InputIfFileExists}{\
828       filehook@InputIfFileExists}%
829     \PackageWarning{filehook}{Detected 'scrlfile' \
830       package with unknown definition of \string\
831       InputIfFileExists.^~J%
832
833       The 'force' option of '
834       filehook' is in \
835       effect. Macro is \
836       overwritten with \
837       default!}%
838   \else
839     \PackageError{filehook}{Detected 'scrlfile' \
840       package with unknown definition of \string\
841       InputIfFileExists.^~J%
842
843       Use the 'force' option of \
844       'filehook' to \
845       overwrite it.}}%
846 \fi
847 \fi
848
849 \endgroup
850
851 %<!COPYRIGHT>
852 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
853 \ProvidesPackage{filehook-fink}[011/01/03 v0.1 \
854   filehook compatibility code for fink package]
855
856 \RequirePackage{filehook}
857 \RequirePackage{currfile}%
858
859 \begingroup

```

```

837
838 \long\def\fink@old@InputIfFileExists#1#2{%
839   \IfFileExists{#1}{%
840     #2\@addtofilelist{#1}%
841     \fink@prepare{#1}%
842     \expandafter\fink@input%
843     \expandafter\fink@restore\expandafter{\finkpath}}%
      %
844   }
845
846 \long\def\fink@new@InputIfFileExists#1#2{%
847   \IfFileExists{#1}{%
848     #2\@addtofilelist{#1}%
849     \edef\fink@before{\noexpand\fink@input{#1}}%
850     \edef\fink@after{\noexpand\fink@restore{\finkpath/
      }}%
851     \expandafter\fink@before\fink@after}%
852   }
853
854 \ifcase
855   \ifx\InputIfFileExists\filehook@InputIfFileExists/
      0\else
856   \ifx\InputIfFileExists\latex@InputIfFileExists /
      1\else
857   \ifx\InputIfFileExists\fink@new@InputIfFileExists/
      1\else
858   \ifx\InputIfFileExists\fink@old@InputIfFileExists/
      1\else
859     1%
860     \fi\fi\fi\fi
861 \relax
862 \or
863   \global\let\filehook@InputIfFileExists\
      filehook@default@InputIfFileExists
864   \global\let\filehook@@InputIfFileExists\
      filehook@@default@InputIfFileExists
865   \global\let\InputIfFileExists\
      filehook@InputIfFileExists
866   \PackageInfo{filehook-fink}{Package 'fink' detected/
      and replaced by 'currfile'}%
867 \else
868   \iffilehook@force
869   \global\let\filehook@InputIfFileExists\
      filehook@default@InputIfFileExists
870   \global\let\filehook@@InputIfFileExists\
      filehook@@default@InputIfFileExists
871   \global\let\InputIfFileExists\
      filehook@InputIfFileExists
872   \PackageWarning{filehook-fink}{Detected 'fink' /
      package with unknown definition of \string\

```



```

      InputIfFileExists.^^J%
873           The 'force' option of '
              filehook' is in /
              effect. Macro is /
              overwritten with /
              default!}%
874 \else
875   \PackageError{filehook-fink}{Detected 'fink' /
      package with unknown definition of \string\
      InputIfFileExists.^^J%
876           Use the 'force' /
              option of '
              filehook' to /
              overwrite it.}{}%
877 \fi
878 \fi
879
880 \endgroup

```

7.13 Support for PGF Keys

```

881 \ProvidesPackage{pgf-filehook}[2010/01/07 v1.0 PGF /
      keys for the filehook package]
882 \RequirePackage{filehook}
883 \RequirePackage{pgfkeys}
884
885 \pgfkeys{%
886   /filehook/.is family,
887   /filehook,
888   %
889   EveryFile/.is family,
890   EveryFile/AtBegin/.code={\AtBeginOfEveryFile/
      {#1}},
891   EveryFile/AtBegin/.value required,
892   EveryFile/AtEnd/.code={\AtEndOfEveryFile{#1}},
893   EveryFile/AtEnd/.value required,
894   %
895   Files/.is family,
896   Files/AtBegin/.code={\AtBeginOfFiles{#1}},
897   Files/AtBegin/.value required,
898   Files/AtEnd/.code={\AtEndOfFiles{#1}},
899   Files/AtEnd/.value required,
900   %
901   File/.is family,
902   File/AtBegin/.code 2 args={\AtBeginOfFile/
      {#1}{#2}},
903   File/AtBegin/.value required,
904   File/AtEnd/.code 2 args={\AtEndOfFile{#1}{#2}},
905   File/AtEnd/.value required,

```

```

906 %
907 Inputs/.is family,
908 Inputs/AtBegin/.code={\AtBeginOfInputs{#1}},
909 Inputs/AtBegin/.value required,
910 Inputs/AtEnd/.code={\AtEndOfInputs{#1}},
911 Inputs/AtEnd/.value required,
912 %
913 InputFile/.is family,
914 InputFile/AtBegin/.code 2 args={\
    AtBeginOfInputFile{#1}{#2}},
915 InputFile/AtBegin/.value required,
916 InputFile/AtEnd/.code 2 args={\AtEndOfInputFile/
    {#1}{#2}},
917 InputFile/AtEnd/.value required,
918 %
919 Includes/.is family,
920 Includes/AtBegin/.code={\AtBeginOfIncludes{#1}},
921 Includes/AtBegin/.value required,
922 Includes/AtEnd/.code={\AtEndOfIncludes{#1}},
923 Includes/AtEnd/.value required,
924 Includes/After/.code={\AfterIncludes{#1}},
925 Includes/After/.value required,
926 %
927 IncludeFile/.is family,
928 IncludeFile/AtBegin/.code 2 args={\
    AtBeginOfIncludeFile{#1}{#2}},
929 IncludeFile/AtBegin/.value required,
930 IncludeFile/AtEnd/.code 2 args={\
    AtEndOfIncludeFile{#1}{#2}},
931 IncludeFile/AtEnd/.value required,
932 IncludeFile/After/.code 2 args={\AfterIncludeFile/
    {#1}{#2}},
933 IncludeFile/After/.value required,
934 %
935 ClassFile/.is family,
936 ClassFile/AtBegin/.code={\AtBeginOfClassFile#1},
937 ClassFile/AtBegin/.value required,
938 ClassFile/AtEnd/.code={\AtEndOfClassFile#1},
939 ClassFile/AtEnd/.value required,
940 %
941 PackageFile/.is family,
942 PackageFile/AtBegin/.code={\AtBeginOfPackageFile/
    #1},
943 PackageFile/AtBegin/.value required,
944 PackageFile/AtEnd/.code={\AtEndOfPackageFile#1},
945 PackageFile/AtEnd/.value required,
946 }
947
948 \newcommand{\pgffilehook}{\pgfqkeys{/filehook}}

```