

The filehook Package

Martin Scharrer
martin.scharrer@web.de

Version v0.8b – 2022/10/25

License: [LPPL v1.3c or later](#)
CTAN: <https://www.ctan.org/pkg/filehook>
Texdoc: <https://texdoc.org/pkg/filehook>
Homepage: <https://github.com/MartinScharrer/filehook>
Repository: <https://github.com/MartinScharrer/filehook.git>
Issue tracker: <https://github.com/MartinScharrer/filehook/issues>

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 L^AT_EX is the ‘At-Begin-Document’ hook. Code can be added to this hook using `\AtBeginDocument{\(T\!E\!X\ code)}`.

This package provides hooks for files read by the L^AT_EX 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 where 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 it 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{\(T\!E\!X\ code)}
\AtEndOfEveryFile{\(T\!E\!X\ 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 `\InputIfExists`. Special care is taken to suppress them for the `\InputIfExists` 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 `\InputIfExists` (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 `\InputIfExists` 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:

```
\include:
\clearpage (implicit)
Hook: AtBeginOfEveryFile
Hook: AtBeginOfIncludeFile{\file name}
Hook: AtBeginOfIncludes
\InputIfExists:
Hook: AtBeginOfFile{\file name}
Hook: AtBeginOfFiles
Content
Hook: AtEndOfFiles
Hook: AtEndOfFile{\file name}
Hook: AtEndOfIncludes
Hook: AtEndOfIncludeFile{\file name}
\clearpage (implicit)
Hook: AfterIncludes
Hook: AfterIncludeFile{\file name}
Hook: AtEndOfEveryFile
```

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:

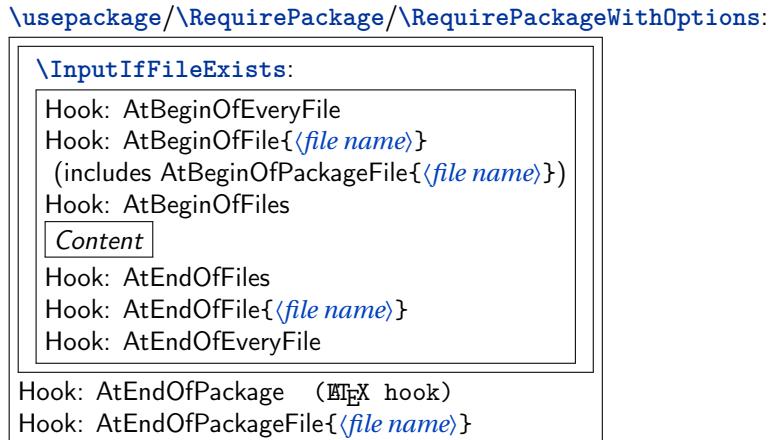
```
\input:
Hook: AtBeginOfEveryFile
Hook: AtBeginOfInputFile{\file name}
Hook: AtBeginOfInputs
\InputIfExists:
Hook: AtBeginOfFile{\file name}
Hook: AtBeginOfFiles
Content
Hook: AtEndOfFiles
Hook: AtEndOfFile{\file name}
Hook: AtEndOfInputs
Hook: AtEndOfInputFile{\file name}
Hook: AtEndOfEveryFile
```

Package Files

```
\AtBeginOfPackageFile*{\langle package name\rangle}{\langle TEX code\rangle}  
\AtEndOfPackageFile*{\langle package name\rangle}{\langle TEX code\rangle}
```

This macros install the given $\langle T_{E}X \text{ code} \rangle$ in the ‘AtBegin’ and ‘AtEnd’ hooks of the given package file. The $\text{\AtBeginOfPackageFile}$ simply executes $\text{\AtBeginOfFile}\{\langle \text{package name} \rangle.\text{sty}\}\{\langle T_{E}X \text{ code} \rangle\}$. Special care is taken to ensure that the ‘AtEnd’ code is executed *after* any code installed by the package itself using the L^AT_EX 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*{\langle class name\rangle}{\langle TEX code\rangle}  
\AtEndOfClassFile*{\langle class name\rangle}{\langle TEX code\rangle}
```

This macros install the given $\langle T_{E}X \text{ code} \rangle$ in the ‘AtBegin’ and ‘AtEnd’ hooks of the given class file. They work with classes loaded using \LoadClass , $\text{\LoadClassWithOptions}$ and also \documentclass . However, in the latter case filehook must be loaded using \RequirePackage beforehand. The macro $\text{\AtBeginOfClassFile}$ simply executes $\text{\AtBeginOfFile}\{\langle \text{class name} \rangle.\text{cls}\}\{\dots\}$. Special care is taken to ensure that the ‘AtEnd’ code is executed *after* any code installed by the class itself using the L^AT_EX macro \AtEndOfClass . Note that it is therefore executed after the ‘AtEndOfEveryFile’ 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:
```

```
\InputIfFileExists:
```

```
Hook: AtBeginOfEveryFile  
Hook: AtBeginOfFile{\file name}  
(includes AtBeginOfClassFile{\file name})  
Hook: AtBeginOfFiles
```

```
Content
```

```
Hook: AtEndOfFiles  
Hook: AtEndOfFile{\file name}  
Hook: AtEndOfEveryFile
```

```
Hook: AtEndOfClass (LATEX hook)  
Hook: AtEndOfClassFile{\file name}
```

3.1 Clearing Hooks

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

New in v0.5 Using this macro existing hooks can be globally cleared, i.e. set to empty. This should
2011/01/09 be used with care because it will also remove all (user level) hook code set by 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 `\pgfkeys{<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 where found do 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`). This 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.

scrlfile

The `scrlfile` 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 `scrlfile` package is loaded after `filehook`.

fink

The `filehook` and `currfile` packages where 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 hooks for <code>\include</code> 'd files
journal	class		All hooks for <code>\include</code> 'd files
gmparts	package		<code>\include</code> hooks
newclude	package	formally includex	All hooks 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 hooks.

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}{%
11   filehook-2019}}
12 \ProcessOptions\relax
```

7.2 Load actual package

```
12 \@ifl@t@r\fmtversion{2020/10/01}{\RequirePackage{%
13   filehook-2020}}{\RequirePackage{filehook-2019}}
14
15  %<! COPYRIGHT >
16  \NeedsTeXFormat{LaTeX2e}[1999/12/01]
17  \ProvidesPackage{filehook-2019}[% filehook subpackage%
18  , do not load directly
19  %<! DATE >
20  %<! VERSION >
21  %<* DRIVER >
22  2099/01/01 develop
23  %</ DRIVER >
24  Hooks for input files]
```

7.3 Options

```
22 \newif\iffil@hook@force
23 \DeclareOption{force}{\fil@hook@forcetrue}
24 \ProcessOptions\relax
```

7.4 General stuff

```
\iffilehook@newfmt
```

```
25 \newif\iffilehook@newfmt
26 \@ifl@t@r\fmtversion{2019/10/01}{\filehook@newfmttrue,}
27 }{\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\
30       \csname #2\endcsname
31   \else
32     \expandafter\def\csname #1\expandafter\endcsname\
33       \expandafter{\expandafter\protect\csname #1\,
34         space\endcsname}%
35     \expandafter\let\csname #1\space\expandafter\,
36       endcsname\csname #2\space\endcsname
37   \fi
38 }
```

```
\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\,
38       \expandafter\endcsname\csname #2\endcsname
39   \else
40     \expandafter\global\expandafter\def\csname #1\,
41       \expandafter\endcsname\expandafter{\expandafter\,
42         \protect\csname #1\space\endcsname}%
43     \expandafter\global\expandafter\let\csname #1\,
44       space\expandafter\endcsname\csname #2\space\,
45       endcsname
46   \fi
47 }
```

```
\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\
46       csname #2\endcsname
47     \expandafter\expandafter\expandafter\expandafter\/
48       @firstoftwo
49   \else
50     \expandafter\expandafter\expandafter\expandafter\/
51       @secondoftwo
52   \fi
53 \else
54   \expandafter\ifx\csname #1\space\expandafter\/
55     endcsname\csname #2\space\endcsname
56   \expandafter\expandafter\expandafter\expandafter\/
57     @firstoftwo
58 \else
59   \expandafter\expandafter\expandafter\expandafter\/
60     @secondoftwo
61   \fi
62 \fi
63 }

```

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

```

`\filehook@include@atbegin`

```

71 \def\filehook@include@atbegin#1{%
72   \filehook@let{InputIfFileExists}{%
73     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{InputIfExists}{%
86         \filehook@InputIfExists}%
87     \filehook@csuse{\filehook@input@atbegin@%
88         \filehook@ensureext{#1}}%
89     \filehook@input@@atbegin
90 }
```

```
\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@\%
93     filehook@ensureext{#1}}%
}
```

```
\filehook@input@@atend
```

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

```
\filehook@atbegin
```

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

```
\filehook@@atbegin
```

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

```
\filehook@atend
```

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

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

\filehook@append

Uses default L^AT_EX 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##\expandafter{%
128       \expandafter1\expandafter{\tempa}%
129     \endgroup
130 }
```

\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##\expandafter{%
143       \expandafter1\expandafter{\tempa}%
144     \endgroup
145 }
```

\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 \atempa are made inside a group to keep them local.

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

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\%
164      filehook@include@atbegin@\{\filehook@ensuretex{%
165      \#1}\}\%
166  }
```

\AtEndOfIncludeFile

```
165  \newcommand*\AtEndOfIncludeFile[1]{%
166      \filehook@addtohook\filehook@prefix\%
167      filehook@include@atend@\{\filehook@ensuretex{\#1}\}\%
168  }
```

\AfterIncludeFile

```
168  \newcommand*\AfterIncludeFile[1]{%
169      \filehook@addtohook\filehook@prefix\%
170      filehook@include@after@\{\filehook@ensuretex{\#1}\}\%
171  }
```

\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\%
179      filehook@input@atbegin@
}
```

\AtEndOfFile

```
180 \newcommand*\AtEndOfFile{%
181     \filehook@addtohook\filehook@prefix\/
182         filehook@input@atend@
183 }
```

\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\/
197         filehook@atbegin@
198 }
```

\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     \@ifl@aded{#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@nd`, 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\/
272   empty\empty
273 }
```

\filehook@@ensureext

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

\filehook@ensuretex

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

```
274 \def\filehook@ensuretex#1{%
275   \expandafter\filehook@ensuretex#1\empty.tex\/
276   empty\empty
277 }
```

```
\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 L^AT_EX definition of `\InputIfFileExists`.

```
278 \iffilehook@newfmt
279 \expandafter\def\expandafter\latex@InputIfFileExists\/
  \expandafter{%
280   \expandafter\protect\csname InputIfFileExists\space\%
    \endcsname
}
281 \expandafter\long\expandafter\def\csname /
  \IfFileExists{#1}%
  {%
    \expandafter\@swaptwoargs\expandafter
      {\@filef@und}{#2\@addtofilelist{#1}\@input}}}
283 \else
284 \long\def\latex@InputIfFileExists#1#2{%
285   \IfFileExists{#1}%
  {#2\@addtofilelist{#1}%
    \@@input\@filef@und
  }%
}
286 \fi
```

```
\filehook@default@InputIfFileExists
```

```
295 \DeclareRobustCommand\/
  filehook@default@InputIfFileExists[2]{%
296   \IfFileExists{#1}%
  {\expandafter\filehook@swap
    \expandafter{\@filef@und}%
  {#2\@addtofilelist{#1}%
    \filehook@every@atbegin{#1}%
    \filehook@atbegin{#1}%
    \@@input}%
    \filehook@atend{#1}%
    \filehook@every@atend{#1}%
  }}
```

```
305     }%
306 }
```

Make sure definition is global:

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

```
\filehook@@default@InputIfFileExists
```

```
308 \DeclareRobustCommand{%
309   filehook@@default@InputIfFileExists [2]{%
310     \filehook@let{InputIfFileExists}{%
311       filehook@InputIfFileExists}%
312     \IfFileExists{#1}{%
313       {\expandafter\filehook@swap
314        \expandafter{\@filef@und}}%
315       {#2\@addtofilelist{#1}%
316        \filehook@atbegin{#1}%
317        \@@input}%
318        \filehook@atend{#1}%
319      }%
320    }}
```

Make sure definition is global:

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

```
\InputIfFileExists
```

First we test for the `scrlfile` 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{scrlfile}{%
321   \filehook@glet{InputIfFileExists}{%
322     latex@InputIfFileExists}%
323 }%
324 \AtEndOfPackageFile*{scrlfile}{%
325   \RequirePackage{filehook-scrlfile}%
326 }
```

Fink:

```
326 \AtBeginOfPackageFile*{fink}{%
327   \RequirePackage{kvoptions}%
328   \begingroup
329   \filehook@let{InputIfFileExists}{%
330     latex@InputIfFileExists}%
331 }
```

```

331 \AtEndOfPackageFile*{fink}{%
332   \edef\@tempa{\noexpand\PassOptionsToPackage{%
333     mainext=\fnk@mainext ,maindir=\fnk@maindir}{%
334       currfile}}%
335   \expandafter\endgroup\@tempa
336   \RequirePackage{filehook-fink}%
337 }%

```

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 `scrlfile` is not explicitly covered. In this case the `scrlfile` package will overwrite `memoirs` definition.

```

338 \AtBeginOfClassFile{memoir}{%
339   \filehook@let{InputIfFileExists}{%
340     latex@InputIfFileExists}%
341   \let\@iinput\filehook@orig@@iinput
342 }%
343 \AtEndOfClassFile*{memoir}{%
344   \let\@iinput\filehook@orig@@iinput
345   \RequirePackage{filehook-memoir}%
346 }%

```

Finally, if no specific alternate definition is detected the original L^AT_EX 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.

```

347 \filehook@cmp{InputIfFileExists}{%
348   filehook@InputIfFileExists}%
349 {}% already set up
350 {}%
351 \filehook@cmp{InputIfFileExists}{%
352   latex@InputIfFileExists}%
353 {}%
354 \filehook@let{filehook@InputIfFileExists}{%
355   filehook@default@InputIfFileExists}%
356 \filehook@let{filehook@@InputIfFileExists}{%
357   filehook@@default@InputIfFileExists}%
358 \filehook@let{InputIfFileExists}{%
359   filehook@InputIfFileExists}%
360 {}%
361 {}%
362 \iffilehook@force
363   \filehook@let{filehook@InputIfFileExists}{%
364     filehook@default@InputIfFileExists}%
365   \filehook@let{filehook@@InputIfFileExists}{%
366     filehook@@default@InputIfFileExists}%
367   \filehook@let{InputIfFileExists}{%
368     filehook@InputIfFileExists}%
369   \PackageWarning{filehook}{Detected unknown /
370     definition of \string\InputIfFileExists/
371     .^^J}%

```

```

359
360     \else
361         \PackageError{filehook}{Detected unknown \
362                         definition of \string\InputIfFileExists/
363                         .^^J%
364
365             \fi
366         }%
367     }%
368
369 \AtBeginDocument{%
370     % Check if definition got changed again. For the \
371     % new LaTeX format we check again \
372     % InputIfExists<space>,
373     % for the old format to \InputIfExists \
374     % directly.
375     \filehook@cmp{\InputIfExists}{%
376         \filehook@InputIfExists{}{}%
377         \PackageWarning{filehook}{Macro \string\
378             InputIfExists\space got redefined \
379             after 'filehook' was loaded.^^J%
380
381             Certain file hooks \
382             might now be \
383             dysfunctional!}%
384     }%
385 }
386
387
388 %<! COPYRIGHT >
389 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
390 \ProvidesPackage{filehook-2020}[% filehook subpackage,
391     , do not load directly
392 %<! DATE >
393 %<! VERSION >
394 %<*DRIVER>
395     2099/01/01 develop
396 %</DRIVER>
397     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  
    \expandafter\ifx\csname ifcsname\endcsname\relax  
        \gdef\filehook@csuse#1{\expandafter\ifx\/  
            \csname #1\endcsname\relax\else\csname #1\  
                \expandafter\endcsname\fi}  
    \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@\%
412   filehook@ensureext{#1}}%
413   \filehook@input@@atbegin
414 }
```

```
\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@\%
418   filehook@ensureext{#1}}%
419 }
```

```
\filehook@input@@atend
```

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

```
\filehook@atbegin
```

```
420 \def\filehook@atbegin#1{%
421   \filehook@csuse{\filehook@atbegin@\%
422     filehook@ensureext{#1}}%
423   \filehook@atbegin
424 }
```

```
\filehook@@atbegin
```

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

```
\filehook@atend
```

```
425 \def\filehook@atend#1{%
426   \filehook@@atend
427   \filehook@csuse{\filehook@atend@\filehook@ensureext%
428     {#1}}%
429 }
```

```
\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
500     \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##\expandafter1\expandafter{\@tempa}%
468     \endgroup
469 }
```

\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     \expandafter\endgroup
476     \expandafter#1\csname\@tempa\endcsname
477 }
```

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\/
504         filehook@input@atbegin@
504 }
```

\AtEndOfInputFile

```
505 \newcommand*\AtEndOfInputFile{%
506     \filehook@addtohook\filehook@prefix\/
507         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

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

\AtBeginOfClassFile

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

\AtBeginOfPackageFile

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

\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 `\@iinput` 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\/
      empty\empty
594 }

```

\filehook@@ensureext

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

\filehook@ensuretex

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

```

596 \def\filehook@ensuretex#1{%
597   \expandafter\filehook@@ensuretex#1\empty.tex\/
      empty\empty
598 }

```

```
\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\v
623   @gobble\expandafter\string\csname.tex\endcsname}
624 \expandafter
625 \gdef\expandafter\filehook@istexfile\expandafter#\v
626   \expandafter1\expandafter{%
627     \expandafter\expandafter\expandafter\expandafter\filehook@istexfile@\expandafter#\expandafter1\v
628     \expandafter\empty\dotex\empty\empty\@nil
629   }
```

```
\filehook@istexfile@
```

```
627 \expandafter\gdef\expandafter\filehook@istexfile@\v
628   \expandafter#\expandafter1\dotex\empty#2\empty#3\v
629   @nil{%
630     \begingroup
631     \def\@tempa{#2}%
632     \ifx\@tempa\empty
633       \endgroup
634       \expandafter\@secondoftwo
635     \else
636       \endgroup
637       \expandafter\@firstoftwo
638     \fi
639   }
640 \endgroup
```

```
\Hook
```

```
639 \AddToHook{file/before}{%
640   \filehook@set@CurrentFile
641   \filehook@every@atbegin{\filehook@CurrentFile}%
642   \filehook@istexfile\filehook@CurrentFile{\
643     \filehook@input@atbegin{\filehook@CurrentFile,
644     }}{}%
645   \filehook@atbegin{\filehook@CurrentFile}%
646 }
```

```
\Hook
```

```

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

651  %<! COPYRIGHT>
652  \NeedsTeXFormat{LaTeX2e}[1999/12/01]
653  \ProvidesPackage{filehook-memoir}[2020/02/02 v0.2 /
654      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      %
659      \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  %@tempswafalse
677  \filehook@cmp{InputIfFileExists}{%
678      filehook@InputIfFileExists}%
679  {\@tempswatrue}%
679  {%

```

```

680     \filehook@cmp{InputIfFileExists}{%
681         memoir@InputIfFileExists}%
682         {\@tempsw@true}%
683         {}%
684     }%
685
686 \if@tempswa
687   \filehook@glet{\filehook@InputIfFileExists}{%
688     \filehook@default@InputIfFileExists}%
689   \filehook@glet{\filehook@@InputIfFileExists}{%
690     \filehook@@default@InputIfFileExists}%
691   \filehook@glet{\InputIfFileExists}{%
692     \filehook@InputIfFileExists}%
693   \filehook@appendwarg{\filehook@atbegin{\m@matbeginf{%
694     {#1}}}}%
695   \filehook@prefixwarg{\filehook@atend{\m@matendf{#1}\%
696     killm@matf{#1}}}}%
697   \PackageInfo{\filehook}{Detected 'memoir' class: the%
698     memoir hooks will be moved to the 'At...OffFiles,%
699     ' hooks}
700 \else
701   \iffilehook@force
702     \filehook@glet{\filehook@InputIfFileExists}{%
703       \filehook@default@InputIfFileExists}%
704     \filehook@glet{\filehook@@InputIfFileExists}{%
705       \filehook@@default@InputIfFileExists}%
706     \filehook@glet{\InputIfFileExists}{%
707       \filehook@InputIfFileExists}%
708     \PackageWarning{\filehook}{Detected 'memoir' class,%
709       with unknown definition of \string\%
710       InputIfFileExists.^~J}%
711     The 'force' option of '\%
712       filehook' is in /%
713       effect. Macro is /%
714       overwritten with /%
715       default!}%
716 \else
717   \PackageError{\filehook}{Detected 'memoir' class,%
718     with unknown definition of \string\%
719     InputIfFileExists.^~J}%
720     Use the 'force' option of,%
721     'filehook' to /%
722     overwrite it.}{}%
723 \fi
724 \fi
725
726 \endgroup
727
728 %<! COPYRIGHT>
729 \NeedsTeXFormat{LaTeX2e}[1999/12/01]

```

```

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

710  \begingroup
711
712  \long\def\patch#1\def\lst@next#2#3\endpatch{%
713    \toks@{\#2}%
714    \edef\@tempa{\the\toks@}%
715    \def\@tempb{\input{####1}}%
716    \ifx\@tempa\@tempb
717      \gdef\lst@InputListing##1{\#1\def\lst@next{\%
718        \input{\#1}}#3}%
719    \else
720      \PackageWarning{filehook-listings}{To-be-
721        patched code in macro \string\%
722        lst@InputListing was not found!}%
723    \fi
724  }
725
726  \expandafter\patch\lst@InputListing{\#1}\endpatch
727
728  \endgroup

729  %<! COPYRIGHT>
730  \NeedsTeXFormat{LaTeX2e}[1999/12/01]
731  \ProvidesPackage{filehook-scrlfile}[2020/02/02 v0.2 %
732    filehook patch for scrlfile package]
733  \RequirePackage{filehook}
734  \begingroup

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

```

```

739     \expandafter\@secondoftwo
740 \else
741     \scr@replacefile@msg{\csname #1-@alias\endcsname,
742     }{#1}%
743     \expandafter\@firstoftwo
744 \fi
745 {\%
746     \expandafter\InputIfFileExists\expandafter{\
747         \csname
748     #1-@alias\endcsname}{#2}%
749 }%
750 {\Iffileexists{#1}{%
751     \expandafter\scr@input@withhook\expandafter{\
752         @filef@und}{#1}{#2}}%
753 }%
754 }

```

\filehook@scrlfile@InputIfFileExists

```

752 \DeclareRobustCommand\  

753     \filehook@scrlfile@InputIfFileExists [2]{%
754     \begingroup\expandafter\expandafter\expandafter\  

755         endgroup
756     \expandafter\ifx\csname #1-@alias\endcsname\relax
757         \expandafter\@secondoftwo
758     \else
759         \scr@replacefile@msg{\csname #1-@alias\endcsname,
760         }{#1}%
761         \expandafter\@firstoftwo
762 \fi
763 {\%
764     \expandafter\InputIfFileExists\expandafter{\
765         \csname
766     #1-@alias\endcsname}{#2}%
767 }%
768 {\Iffileexists{#1}{%
769     \expandafter\filehook@swap
770     \expandafter{\@filef@und}%
771     {\scr@load@hook{before}{#1}%
772     #2\@addtolist{#1}%
773     \filehook@every@atbegin{#1}%
774     \filehook@atbegin{#1}%
775     \@@input}%
776     \filehook@atend{#1}%
777     \filehook@every@atend{#1}%
778     \scr@load@hook{after}{#1}%
779 }{%
780 }

```

```

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



\filehook@@scrlfile@InputIfFileExists



778 \DeclareRobustCommand\%
779   filehook@@scrlfile@InputIfFileExists[2]{%
780   \filehook@let{InputIfFileExists}{%
781     filehook@InputIfFileExists}%
782   \begingroup\expandafter\expandafter\expandafter\%
783     endgroup
784   \expandafter\ifx\csname #1-@alias\endcsname\relax
785     \expandafter\@secondoftwo
786   \else
787     \scr@replacefile@msg{\csname #1-@alias\endcsname%
788       }{\#1}%
789     \expandafter\@firstoftwo
790   \fi
791   {%
792     \expandafter\InputIfFileExists\expandafter{%
793       \csname
794         #1-@alias\endcsname}{#2}%
795   }%
796   {\IfFileExists{#1}{%
797     \expandafter\filehook@swap
798     \expandafter{\@filef@und}%
799     {\scr@load@hook{before}{#1}%
800       #2\@addtolist{#1}%
801       \filehook@atbegin{#1}%
802       \@@input}%
803       \filehook@atend{#1}%
804       \scr@load@hook{after}{#1}%
805     }{%
806       \tempswafalse
807       \filehook@cmp{InputIfFileExists}{%
808         filehook@InputIfFileExists}%
809       {\tempswatrue}%
810     }%
811     \filehook@cmp{InputIfFileExists}{%
812       scrlfile@InputIfFileExists}%
813   }%
814 }
```

If the `scrlfile` 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\@addtolist{#1}` code. Otherwise the `filehooks` could simply be added to these hooks. Note that this will stop working if `scrlfile` ever changes its definition of the `\InputIfFileExists` macro.

```

803 \tempswafalse
804 \filehook@cmp{InputIfFileExists}{%
805   filehook@InputIfFileExists}%
806 {\tempswatrue}%
807 {%
808   \filehook@cmp{InputIfFileExists}{%
809     scrlfile@InputIfFileExists}%
810 }
```

```

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

```

```

837
838 \long\def\fink@old@InputIfFileExists#1#2{%
839   \IfExists{#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   \IfExists{#1}{%
848     #2\@addtofilelist{#1}%
849     \edef\fink@before{\noexpand\fink@input{#1}}%
850     \edef\fink@after{\noexpand\fink@restore{\finkpath}%
851       }%
852     \expandafter\fink@before\fink@after}%
853 }
854 \ifcase
855   \ifx\InputIfFileExists\filehook@InputIfFileExists,
856   0\else
857   \ifx\InputIfFileExists\latex@InputIfFileExists ,
858   1\else
859   \ifx\InputIfFileExists\fink@new@InputIfFileExists,
860   1\else
861   \ifx\InputIfFileExists\fink@old@InputIfFileExists,
862   1\else
863     1%
864     \fi\fi\fi\fi
865 \relax
866 \or
867   \global\let\filehook@InputIfFileExists\/
868     filehook@default@InputIfFileExists
869   \global\let\filehook@@InputIfFileExists\/
870     filehook@@default@InputIfFileExists
871   \global\let\InputIfFileExists\/
872     filehook@InputIfFileExists
873   \PackageInfo{filehook-fink}{Package 'fink' detected,%
874     and replaced by 'currfile'}%
875 \else
876   \iffilehook@force
877     \global\let\filehook@InputIfFileExists\/
878       filehook@default@InputIfFileExists
879     \global\let\filehook@@InputIfFileExists\/
880       filehook@@default@InputIfFileExists
881     \global\let\InputIfFileExists\/
882       filehook@InputIfFileExists
883   \PackageWarning{filehook-fink}{Detected 'fink' ,%
884     package with unknown definition of \string\/

```

```

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

```

7.13 Support for PGF Keys

```

881 \ProvidesPackage{pgf-filehook}[2010/01/07 v1.0 PGF /
882   keys for the filehook package]
883 \RequirePackage{filehook}
884 \RequirePackage{pgfkeys}
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={\/
915     AtBeginOfFile{#1}{#2}},
916     inputFile/AtBegin/.value required,
917     inputFile/AtEnd/.code 2 args={\AtEndOfFile{#
918         #1}{#2}},
919     inputFile/AtEnd/.value required,
920 %
921 Includes/.is family,
922 Includes/AtBegin/.code={\AtBeginOfIncludes{#1}},
923 Includes/AtBegin/.value required,
924 Includes/AtEnd/.code={\AtEndOfIncludes{#1}},
925 Includes/AtEnd/.value required,
926 Includes/After/.code={\AfterIncludes{#1}},
927 Includes/After/.value required,
928 %
929 IncludeFile/.is family,
930 IncludeFile/AtBegin/.code 2 args={\/
931     AtBeginOfIncludeFile{#1}{#2}},
932     IncludeFile/AtBegin/.value required,
933     IncludeFile/AtEnd/.code 2 args={\/
934         AtEndOfIncludeFile{#1}{#2}},
935     IncludeFile/AtEnd/.value required,
936     IncludeFile/After/.code 2 args={\AfterIncludeFile{#
937         #1}{#2}},
938     IncludeFile/After/.value required,
939 %
940 ClassFile/.is family,
941 ClassFile/AtBegin/.code={\AtBeginOfClassFile{#1}},
942 ClassFile/AtBegin/.value required,
943 ClassFile/AtEnd/.code={\AtEndOfClassFile{#1}},
944 ClassFile/AtEnd/.value required,
945 %
946 PackageFile/.is family,
947 PackageFile/AtBegin/.code={\AtBeginOfPackageFile{#
948     #1}},
949     PackageFile/AtBegin/.value required,
950     PackageFile/AtEnd/.code={\AtEndOfPackageFile{#1}},
951     PackageFile/AtEnd/.value required,
952 }
953 %
954 \newcommand{\pgffilehook}{\pgfqkeys{/filehook}}

```