The filehook Package

Martin Scharrer
martin@scharrer-online.de

http://www.ctan.org/pkg/filehook

Version v0.5b – 2011/07/18

Abstract

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

1 Introduction

These package changes some internal LaTeX macros used to load input files so that they include 'hooks'. A hook is an (internal) macro executed at specific points. Normally it is initially empty, but can be extended using an user level macro. The most common hook in LaTeX is the 'At-Begin-Document' hook. Code can be added to this hook using `\AtBeginDocument{`⟨*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 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 5 for information how to upgrade older documents.

2 Usage

The below macros can be used to add material (TeX code) to the related hooks. All 'AtBegin' macros will *append* the code to the hooks, but the 'AtEnd' and 'After' macros will *prefix* the code instead. This ensures that two different packages adding material in 'AtBegin'/'AtEnd' pairs do not overlap each other. Instead the later used package adds the code closer to the file content, 'inside' the material added by the first package. Therefore it is safely possible to surround the content of a file with multiple LaTeX 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:

`\include`:

```
\clearpage  (implicit)
Hook: AtBeginOfEveryFile
Hook: AtBeginOfIncludeFile{⟨file name⟩}
Hook: AtBeginOfIncludes
    \InputIfFileExists:
        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
    \InputIfFileExists:
        Hook: AtBeginOfFile{⟨file name⟩}
        Hook: AtBeginOfFiles
        Content
        Hook: AtEndOfFiles
        Hook: AtEndOfFile{⟨file name⟩}
Hook: AtEndOfInputs
Hook: AtEndOfInputFile{⟨file name⟩}
Hook: AtEndOfEveryFile
```

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 LaTeX 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:

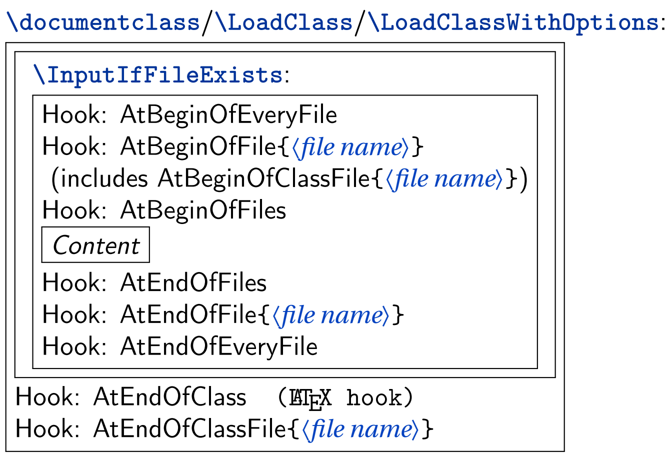
```
\usepackage/\RequirePackage/\RequirePackageWithOptions:
┌──────────────────────────────────────────────────────┐
│ ┌──────────────────────────────────────────────────┐ │
│ │ \InputIfFileExists:                              │ │
│ │ ┌──────────────────────────────────────────────┐ │ │
│ │ │ Hook: AtBeginOfEveryFile                     │ │ │
│ │ │ Hook: AtBeginOfFile{⟨file name⟩}             │ │ │
│ │ │  (includes AtBeginOfPackageFile{⟨file name⟩})│ │ │
│ │ │ Hook: AtBeginOfFiles                         │ │ │
│ │ │ ┌─────────┐                                  │ │ │
│ │ │ │ Content │                                  │ │ │
│ │ │ └─────────┘                                  │ │ │
│ │ │ Hook: AtEndOfFiles                           │ │ │
│ │ │ Hook: AtEndOfFile{⟨file name⟩}               │ │ │
│ │ │ Hook: AtEndOfEveryFile                       │ │ │
│ │ └──────────────────────────────────────────────┘ │ │
│ └──────────────────────────────────────────────────┘ │
│ Hook: AtEndOfPackage   (LaTeX hook)                  │
│ Hook: AtEndOfPackageFile{⟨file name⟩}                │
└──────────────────────────────────────────────────────┘
```

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 LaTeX 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*⟩}

2.1 Clearing Hooks

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

New in v0.5 2011/01/09

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

`\ClearHook\AtBeginOfInputFile{`⟨*file name*⟩`}{`⟨*ignored*⟩`}`
`\ClearHook\AtBeginOfFiles{`⟨*ignored*⟩`}`

3 PGF Key Interface

An auxiliary package `pgf-filehook` is provided which adds support for the versatile `pgfkeys` interface. This interface is heavily used by `pgf` (portable graphics format) and its higher level format Ti*k*Z. 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*⟩`,...}`. Ti*k*Z provides the similar macro `\tikzstyle` which defaults to the main path '`/tikz`'. More detailed information can be found in the official `pgfmanual`.

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

`\pgfkeys{/filehook/Inputs/AtBegin={`⟨*code*⟩`}}`

or `\AfterIncludeFile{`⟨*file name*⟩`}{`⟨*code*⟩`}` as

`\pgfkeys{/filehook/IncludeFile/After={`⟨*file name*⟩`}{`⟨*code*⟩`}}`

as well as `\AtEndOfClassFile*{`⟨*file name*⟩`}{`⟨*code*⟩`}` as

`\pgfkeys{/filehook/ClassFile/AtEnd=*{`⟨*file name*⟩`}{`⟨*code*⟩`}}`.

`\pgffilehook{`⟨*key*⟩`=`⟨*value*⟩`,...}`

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

`\pgfkeys{/filehook/Inputs/AtBegin={`⟨*code*⟩`},/filehook/Inputs/AtEnd={`⟨*code*⟩`}}`

can be shorten as

`\pgffilehook{Inputs,AtBegin={`⟨*code*⟩`},AtEnd={`⟨*code*⟩`}}`.

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

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

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

4 Compatibility Issues with Classes and other Packages

The `filehook` package might clash with other packages or classes which also redefine `\InputIfFileExists` or internal macros used by `\include` and `\input` (which are `\@input@` and `\@iinput`). Special compatibility code is in place for the packages listed below (in their current implementation). If any other unknown definition of `\InputIfFileExists` is found an error will be raised. The package option '`force`' can be used to prevent this and to force the redefinition of this macro. Then any previous modifications will be lost, which will most likely break the other package. Table 1 lists all packages and classes which 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.

4.1 Supported Classes and Packages

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

memoir

The `memoir` class redefines `\InputIfFileExists` to add own hooks identical to the 'At...OfFiles' hooks (there called `\AtBeginFile` and `\AtEndFile`). 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 `\include`'d files
`journal`	class		All hooks for `\include`'d files
`gmparts`	package		`\include` hooks
`newclude`	package	formally `includex`	All hooks for `\include`'d files

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

listings

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

4.2 Other Classes and Packages

jmlrbook

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

LaTeX's \bibliography

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

5 Upgrade Guide

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

Upgrade to v0.4 - 2011/01/03

- The macro `\AfterIncludeFile` was misspelled as `\AfterOfIncludeFile` in the implementation of earlier versions, but not in the documentation. This has now be corrected. Please adjust your code to use the correct name and to require the `filehook` package from 2011/01/03.

- All general hooks (the one not taking a file argument) used to have an implicit argument `#1` which was expanded to the file name (i.e. the argument of `\input` etc.). This has now be changed, so that macro arguments are not handled special in hook code, which e.g. simplifies macro definitions. Older hook code might need to change `##` to `#` to compensate for this change. If the file name is required the macros (e.g. `\currfilename`) of the partner package `currfile` should be used. These macros are available everywhere including in all hocks.

6 Implementation

6.1 Options

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

6.2 Initialisation of Hooks

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

```
\@ifpackageloaded{etoolbox}{%
    \let\filehook@csuse\csuse
}{%
    \def\filehook@csuse#1{\ifcsname #1\endcsname\
        csname #1\expandafter\endcsname\fi}
}
```

\filehook@include@atbegin

```
\def\filehook@include@atbegin#1{%
  \let\InputIfFileExists\filehook@@InputIfFileExists
  \filehook@csuse{\filehook@include@atbegin@#1}%
  \filehook@include@@atbegin
}
```

\filehook@include@@atbegin

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

\filehook@include@atend

```
\def\filehook@include@atend#1{%
  \filehook@include@@atend
  \filehook@csuse{\filehook@include@atend@#1}%
}
```

\filehook@include@@atend

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

\filehook@include@after

```
\def\filehook@include@after#1{%
  \filehook@include@@after
  \filehook@csuse{\filehook@include@after@#1}%
}
```

\filehook@include@@after

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

\filehook@input@atbegin

```
\def\filehook@input@atbegin#1{%
  \let\InputIfFileExists\filehook@@InputIfFileExists
  \filehook@csuse{\filehook@input@atbegin@\filehook@ensureext{#1}}%
  \filehook@input@@atbegin
}
```

\filehook@input@@atbegin

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

\filehook@input@atend

```
\def\filehook@input@atend#1{%
  \filehook@input@@atend
  \filehook@csuse{\filehook@input@atend@\filehook@ensureext{#1}}%
}
```

\filehook@input@@atend

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

\filehook@atbegin

```
\def\filehook@atbegin#1{%
  \filehook@csuse{\filehook@atbegin@\filehook@ensureext{#1}}%
  \filehook@@atbegin
}
```

\filehook@@atbegin

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

\filehook@atend

```
\def\filehook@atend#1{%
  \filehook@@atend
  \filehook@csuse{\filehook@atend@\filehook@ensureext{#1}}%
}
```

\filehook@@atend

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

\filehook@every@atbegin

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

\filehook@every@@atbegin

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

\filehook@every@atend

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

`\filehook@every@@atend`

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

6.3 Hook Modification Macros

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

Internal Macros

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

```
\def\filehook@include@atbegin@{filehook@include@atbegin@}
\def\filehook@include@atend@{filehook@include@atend@}
\def\filehook@include@after@{filehook@include@after@}
\def\filehook@input@atbegin@{filehook@input@atbegin@}
\def\filehook@input@atend@{filehook@input@atend@}
\def\filehook@input@after@{filehook@input@after@}
\def\filehook@atbegin@{filehook@atbegin@}
\def\filehook@atend@{filehook@atend@}
\def\filehook@after@{filehook@after@}
```

`\filehook@append`

Uses default LaTeX macro.

```
\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.

```
\long\def\filehook@appendwarg#1#2{%
  \begingroup
    \toks@\expandafter{#1{##1}#2}%
    \edef\@tempa{\the\toks@}%
    \expandafter\gdef\expandafter#1\expandafter##\expandafter1\expandafter{\@tempa}%
  \endgroup
}
```

`\filehook@prefix`

Prefixes code to a hook.

```
\long\def\filehook@prefix#1#2{%
  \begingroup
    \@temptokena{#2}%
    \toks@\expandafter{#1}%
    \xdef#1{\the\@temptokena\the\toks@}%
  \endgroup
}
```

`\filehook@prefixwarg`

Prefixes code with an argument to a hook.

```
\long\def\filehook@prefixwarg#1#2{%
  \begingroup
    \@temptokena{#2}%
    \toks@\expandafter{#1{##1}}%
    \edef\@tempa{\the\@temptokena\the\toks@}%
    \expandafter\gdef\expandafter#1\expandafter##\expandafter1\expandafter{\@tempa}%
  \endgroup
}
```

`\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.

```
\def\filehook@addtohook#1#2#3{%
  \begingroup
  \edef\@tempa{#3}%
  \edef\@tempa{#2\filehook@ensureext{\@tempa}}%
  \@ifundefined{\@tempa}{\global\@namedef{\@tempa}{}}{}%
  \expandafter\endgroup
  \expandafter#1\csname\@tempa\endcsname
}
```

User Level Macros

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

\AtBeginOfIncludes

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

\AtEndOfIncludes

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

\AfterIncludes

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

\AtBeginOfIncludeFile

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

\AtEndOfIncludeFile

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

\AfterIncludeFile

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

\AtBeginOfInputs

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

\AtEndOfInputs

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

\AtBeginOfInputFile

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

\AtEndOfInputFile

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

\AtBeginOfFiles

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

\AtEndOfFiles

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

\AtBeginOfEveryFile

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

\AtEndOfEveryFile

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

\AtBeginOfFile

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

\AtEndOfFile

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

\AtBeginOfClassFile

```
\newcommand*\AtBeginOfClassFile{%
    \@ifnextchar*
        {\AtBeginOfXFile@star\@clsextension}%
        {\AtBeginOfXFile@normal\@clsextension}%
}
```

\AtBeginOfPackageFile

```
\newcommand*\AtBeginOfPackageFile{%
    \@ifnextchar*
        {\AtBeginOfXFile@star\@pkgextension}%
        {\AtBeginOfXFile@normal\@pkgextension}%
}
```

`\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.

```
\def\AtBeginOfXFile@star#1*#2{%
    \@ifl@aded{#1}{#2}%
        {\@firstofone}%
        {\AtBeginOfXFile@normal{#1}{#2}}%
}
```

`\AtBeginOfXFile@normal`

#1: extension
#2: name

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

`\AtEndOfClassFile`

```
\newcommand*\AtEndOfClassFile{%
    \@ifnextchar*
        {\AtEndOfXFile@star\@clsextension}%
        {\AtEndOfXFile@normal\@clsextension}%
}
```

`\AtEndOfPackageFile`

```
\newcommand*\AtEndOfPackageFile{%
    \@ifnextchar*
        {\AtEndOfXFile@star\@pkgextension}%
        {\AtEndOfXFile@normal\@pkgextension}%
}
```

`\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.

```
\def\AtEndOfXFile@star#1*#2{%
    \@ifl@aded{#1}{#2}%
        {\@firstofone}%
        {\AtEndOfXFile@normal{#1}{#2}}%
}
```

`\AtEndOfXFile@normal`

#1: extension
#2: name

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

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

`\ClearHook`

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

```
\newcommand*\ClearHook{%
    \begingroup
    \def\filehook@prefix##1##2{%
        \gdef##1{}%
        \endgroup
    }%
    \let\filehook@append\filehook@prefix
}
```

6.4 Installation of Hooks

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

First the original definitions are saved away.

`\filehook@orig@@input@`

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

`\filehook@orig@@iinput`

```
\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.

```
\def\@input@#1{%
  \@ifnextchar\clearpage
    {%
     \filehook@every@atbegin{#1}%
     \filehook@include@atbegin{#1}%
     \filehook@orig@@input@{#1}%
     \filehook@include@atend{#1}%
     \clearpage
     \filehook@include@after{#1}%
     \filehook@every@atend{#1}%
     \@gobble
    }%
    {\filehook@orig@@input@{#1}}%
}
```

`\@iinput`

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

```
\def\filehook@@iinput#1{%
  \filehook@every@atbegin{#1}%
  \filehook@input@atbegin{#1}%
  \filehook@orig@@iinput{#1}%
  \filehook@input@atend{#1}%
  \filehook@every@atend{#1}%
}
\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.

```
\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.

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

`\filehook@@ensureext`

```
\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.

```
\def\filehook@ensuretex#1{%
    \expandafter\filehook@@ensuretex#1\empty.tex\empty\empty
}
```

`\filehook@@ensuretex`

```
\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 LaTeX definition of `\InputIfFileExists`.

```
\long\def\latex@InputIfFileExists#1#2{%
  \IfFileExists{#1}%
    {#2\@addtofilelist{#1}%
     \@@input\@filef@und
    }%
}
```

`\filehook@default@InputIfFileExists`

```
\long\gdef\filehook@default@InputIfFileExists#1#2{%
  \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}%
    }%
}
```

\filehook@@default@InputIfFileExists

```
\long\gdef\filehook@@default@InputIfFileExists#1#2{%
  \let\InputIfFileExists\filehook@InputIfFileExists
  \IfFileExists{#1}%
    {\expandafter\filehook@swap
     \expandafter{\@filef@und}%
     {#2\@addtofilelist{#1}%
     \filehook@atbegin{#1}%
     \@@input}%
     \filehook@atend{#1}%
    }%
}
```

\scrlfile@InputIfFileExists

```
\long\def\scrlfile@InputIfFileExists#1#2{%
  \begingroup\expandafter\expandafter\expandafter\
      endgroup
  \expandafter\ifx\csname #1-@alias\endcsname\relax
    \expandafter\@secondoftwo
  \else
    \scr@replacefile@msg{\csname #1-@alias\endcsname
        }{#1}%
    \expandafter\@firstoftwo
  \fi
  {%
    \expandafter\InputIfFileExists\expandafter{\
        csname
      #1-@alias\endcsname}{#2}%
  }%
  {\IfFileExists{#1}{%
      \scr@load@hook{before}{#1}%
      #2\@addtofilelist{#1}%
      \@@input \@filef@und
      \scr@load@hook{after}{#1}%
    }}%
}
```

\filehook@scrlfile@InputIfFileExists

```
\long\def\filehook@scrlfile@InputIfFileExists#1#2{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname #1-@alias\endcsname\relax
    \expandafter\@secondoftwo
  \else
    \scr@replacefile@msg{\csname #1-@alias\endcsname}{#1}%
    \expandafter\@firstoftwo
  \fi
  {%
    \expandafter\InputIfFileExists\expandafter{\csname
      #1-@alias\endcsname}{#2}%
  }%
  {\IfFileExists{#1}{%
      \expandafter\filehook@swap
      \expandafter{\@filef@und}%
      {\scr@load@hook{before}{#1}%
      #2\@addtofilelist{#1}%
      \filehook@every@atbegin{#1}%
      \filehook@atbegin{#1}%
      \@@input}%
      \filehook@atend{#1}%
      \filehook@every@atend{#1}%
      \scr@load@hook{after}{#1}%
    }}%
}
```

\filehook@@scrlfile@InputIfFileExists

```
\long\def\filehook@@scrlfile@InputIfFileExists#1#2{%
  \let\InputIfFileExists\filehook@InputIfFileExists
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname #1-@alias\endcsname\relax
    \expandafter\@secondoftwo
  \else
    \scr@replacefile@msg{\csname #1-@alias\endcsname}{#1}%
```

```
      \expandafter\@firstoftwo
    \fi
    {%
      \expandafter\InputIfFileExists\expandafter{\csname
        #1-@alias\endcsname}{#2}%
    }%
    {\IfFileExists{#1}{%
        \expandafter\filehook@swap
        \expandafter{\@filef@und}%
        {\scr@load@hook{before}{#1}%
        #2\@addtofilelist{#1}%
        \filehook@atbegin{#1}%
        \@@input}%
        \filehook@atend{#1}%
        \scr@load@hook{after}{#1}%
      }}%
}
```

```
\ProvidesPackage{filehook-memoir}[2011/01/03 v0.1 filehook patch for memoir class]
\RequirePackage{filehook}
\begingroup
```

\memoir@InputIfFileExists

```
\long\def\memoir@InputIfFileExists#1#2{%
  \IfFileExists{#1}%
    {#2\@addtofilelist{#1}\m@matbeginf{#1}%
     \@@input \@filef@und
     \m@matendf{#1}%
     \killm@matf{#1}}%
}
```

```
\ifcase
      \ifx\InputIfFileExists\latex@InputIfFileExists 0\else
      \ifx\InputIfFileExists\memoir@InputIfFileExists 0\else
       1%
      \fi\fi
\relax
  \global\let\filehook@InputIfFileExists\filehook@default@InputIfFileExists
  \global\let\filehook@@InputIfFileExists\filehook@@default@InputIfFileExists
  \global\let\InputIfFileExists\filehook@InputIfFileExists
  \filehook@appendwarg\filehook@atbegin{\m@matbeginf{#1}}%
  \filehook@prefixwarg\filehook@atend{\m@matendf{#1}\killm@matf{#1}}%
  \PackageInfo{filehook}{Detected 'memoir' class: the memoir hooks will be moved to the `At...OfFiles' hooks}
\else
  \iffilehook@force
    \global\let\filehook@InputIfFileExists\filehook@default@InputIfFileExists
    \global\let\filehook@@InputIfFileExists\filehook@@default@InputIfFileExists
    \global\let\InputIfFileExists\filehook@InputIfFileExists
    \PackageWarning{filehook}{Detected 'memoir' class with unknown definition of \string\InputIfFileExists.^^J%
                              The 'force' option of 'filehook' is in effect. Macro is overwritten with default!}%
  \else
    \PackageError{filehook}{Detected 'memoir' class with unknown definition of \string\InputIfFileExists.^^J%
                              Use the 'force' option of 'filehook' to overwrite it.}{}%
  \fi
\fi

\endgroup

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

\long\def\patch#1\def\lst@next#2#3\endpatch{%
    \toks@{#2}%
    \edef\@tempa{\the\toks@}%
    \def\@tempb{\input{####1}}%
    \ifx\@tempa\@tempb
        \gdef\lst@InputListing##1{#1\def\lst@next{\@input{##1}}#3}%
    \else
        \PackageWarning{filehook-listings}{To-be-patched code in macro \string\
            lst@InputListing was not found!}%
    \fi
}

\@ifundefined{lst@InputListing}{%
    \PackageWarning{filehook-listings}{To-be-patched ⁄
       Macro \string\lst@InputListing not found!}%
}{}

\expandafter\patch\lst@InputListing{#1}\endpatch

\endgroup
```

```
\ProvidesPackage{filehook-scrlfile}[2011/01/03 v0.1 ⁄
   filehook patch for scrlfile package]
\RequirePackage{filehook}
\begingroup
```

\scrlfile@InputIfFileExists

```
\long\def\scrlfile@InputIfFileExists#1#2{%
  \begingroup\expandafter\expandafter\expandafter\⁄
     endgroup
  \expandafter\ifx\csname #1-@alias\endcsname\relax
    \expandafter\@secondoftwo
  \else
    \scr@replacefile@msg{\csname #1-@alias\endcsname⁄
       }{#1}%
    \expandafter\@firstoftwo
  \fi
  {%
    \expandafter\InputIfFileExists\expandafter{\⁄
       csname
      #1-@alias\endcsname}{#2}%
  }%
  {\IfFileExists{#1}{%
      \scr@load@hook{before}{#1}%
      #2\@addtofilelist{#1}%
      \@@input \@filef@und
      \scr@load@hook{after}{#1}%
    }}%
}
```

\filehook@scrlfile@InputIfFileExists

```
\long\def\filehook@scrlfile@InputIfFileExists#1#2{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname #1-@alias\endcsname\relax
    \expandafter\@secondoftwo
  \else
    \scr@replacefile@msg{\csname #1-@alias\endcsname}{#1}%
    \expandafter\@firstoftwo
  \fi
  {%
    \expandafter\InputIfFileExists\expandafter{\csname
      #1-@alias\endcsname}{#2}%
  }%
  {\IfFileExists{#1}{%
      \expandafter\filehook@swap
      \expandafter{\@filef@und}%
      {\scr@load@hook{before}{#1}%
      #2\@addtofilelist{#1}%
      \filehook@every@atbegin{#1}%
      \filehook@atbegin{#1}%
      \@@input}%
      \filehook@atend{#1}%
      \filehook@every@atend{#1}%
      \scr@load@hook{after}{#1}%
    }}%
}
```

\filehook@@scrlfile@InputIfFileExists

```
\long\def\filehook@@scrlfile@InputIfFileExists#1#2{%
  \let\InputIfFileExists\filehook@InputIfFileExists
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname #1-@alias\endcsname\relax
    \expandafter\@secondoftwo
  \else
    \scr@replacefile@msg{\csname #1-@alias\endcsname}{#1}%
    \expandafter\@firstoftwo
  \fi
  {%
    \expandafter\InputIfFileExists\expandafter{\csname
      #1-@alias\endcsname}{#2}%
  }%
  {\IfFileExists{#1}{%
```

```
        \expandafter\filehook@swap
        \expandafter{\@filef@und}%
        {\scr@load@hook{before}{#1}%
        #2\@addtofilelist{#1}%
        \filehook@atbegin{#1}%
        \@@input}%
        \filehook@atend{#1}%
        \scr@load@hook{after}{#1}%
    }}%
}
```

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\@addtofilelist{#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.

```
\ifcase
    \ifx\InputIfFileExists\latex@InputIfFileExists 0\else
    \ifx\InputIfFileExists\scrlfile@InputIfFileExists 0\else
     1%
    \fi\fi
\relax
  \global\let\filehook@InputIfFileExists\filehook@scrlfile@InputIfFileExists
  \global\let\filehook@@InputIfFileExists\filehook@@scrlfile@InputIfFileExists
  \global\let\InputIfFileExists\filehook@InputIfFileExists
  \PackageInfo{filehook}{Package 'scrlfile' detected and compensated for}%
\else
  \iffilehook@force
    \global\let\filehook@InputIfFileExists\filehook@default@InputIfFileExists
    \global\let\filehook@@InputIfFileExists\filehook@@default@InputIfFileExists
    \global\let\InputIfFileExists\filehook@InputIfFileExists
    \PackageWarning{filehook}{Detected 'scrlfile' package with unknown definition of \string\InputIfFileExists.^^J%
                              The 'force' option of 'filehook' is in effect. Macro is overwritten with default!}%
  \else
    \PackageError{filehook}{Detected 'scrlfile'
```

```
package with unknown definition of \string\InputIfFileExists.^^J%
                           Use the 'force' option of 'filehook' to overwrite it.}{}%
  \fi
\fi

\endgroup

\ProvidesPackage{filehook-fink}[2011/01/03 v0.1 filehook compatibility code for fink package]
\RequirePackage{filehook}
\RequirePackage{currfile}%

\begingroup

\long\def\fink@old@InputIfFileExists#1#2{%
  \IfFileExists{#1}{%
    #2\@addtofilelist{#1}%
    \fink@prepare{#1}%
    \expandafter\fink@input%
    \expandafter\fink@restore\expandafter{\finkpath}%
}

\long\def\fink@new@InputIfFileExists#1#2{%
  \IfFileExists{#1}{%
    #2\@addtofilelist{#1}%
    \edef\fink@before{\noexpand\fink@input{#1}}%
    \edef\fink@after{\noexpand\fink@restore{\finkpath}}%
    \expandafter\fink@before\fink@after}%
}

\ifcase
    \ifx\InputIfFileExists\filehook@InputIfFileExists 0\else
    \ifx\InputIfFileExists\latex@InputIfFileExists 1\else
    \ifx\InputIfFileExists\fink@new@InputIfFileExists 1\else
    \ifx\InputIfFileExists\fink@old@InputIfFileExists 1\else
     1%
    \fi\fi\fi\fi
\relax
\or
  \global\let\filehook@InputIfFileExists\filehook@default@InputIfFileExists
  \global\let\filehook@@InputIfFileExists\filehook@@default@InputIfFileExists
  \global\let\InputIfFileExists\filehook@InputIfFileExists
  \PackageInfo{filehook-fink}{Package 'fink' detected and replaced by 'currfile'}%
\else
  \iffilehook@force
    \global\let\filehook@InputIfFileExists\filehook@default@InputIfFileExists
    \global\let\filehook@@InputIfFileExists\filehook@@default@InputIfFileExists
    \global\let\InputIfFileExists\filehook@InputIfFileExists
    \PackageWarning{filehook-fink}{Detected 'fink' package with unknown definition of \string\InputIfFileExists.^^J%
                              The 'force' option of 'filehook' is in effect. Macro is overwritten with default!}%
  \else
    \PackageError{filehook-fink}{Detected 'fink' package with unknown definition of \string\InputIfFileExists.^^J%
                                 Use the 'force' option of 'filehook' to overwrite it.}{}%
  \fi
\fi

\endgroup
```

`\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.

```
\AtBeginOfPackageFile*{scrlfile}{%
   \let\InputIfFileExists\latex@InputIfFileExists
}%
\AtEndOfPackageFile*{scrlfile}{%
   \RequirePackage{filehook-scrlfile}%
}%
```

Fink:

```
\AtBeginOfPackageFile*{fink}{%
   \RequirePackage{kvoptions}%
   \begingroup
   \let\InputIfFileExists\latex@InputIfFileExists
}%
\AtEndOfPackageFile*{fink}{%
   \edef\@tempa{\noexpand\PassOptionsToPackage{mainext=\fnk@mainext,maindir=\fnk@maindir}{currfile}}%
   \expandafter\endgroup\@tempa
   \RequirePackage{filehook-fink}%
}%
```

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.

```
\AtBeginOfClassFile*{memoir}{%
  \let\filehook@@InputIfFileExists\latex@InputIfFileExists
  \let\InputIfFileExists\latex@InputIfFileExists
  \let\@iinput\filehook@orig@@iinput
}%
\AtEndOfClassFile*{memoir}{%
  \let\@iinput\filehook@@iinput
  \RequirePackage{filehook-memoir}%
}%
```

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.

```
\ifcase
    \ifx\InputIfFileExists\filehook@InputIfFileExists 0\else
    \ifx\InputIfFileExists\latex@InputIfFileExists 1\else
    \iffilehook@force 1\else
    9%
    \fi\fi\fi
\relax% 0
\or% 1
    \let\filehook@InputIfFileExists\filehook@default@InputIfFileExists
    \let\filehook@@InputIfFileExists\filehook@@default@InputIfFileExists
    \let\InputIfFileExists\filehook@InputIfFileExists
    \iffilehook@force
      \PackageWarning{filehook}{Detected unknown definition of \string\InputIfFileExists.^^J%
```

```
                                        The 'force' option of 'filehook' is in effect. Macro is overwritten with default!}%
    \fi
\else
    \PackageError{filehook}{Detected unknown definition of \string\InputIfFileExists.^^J%
                                Use the 'force' option of 'filehook' to overwrite it.}{}%
\fi

\AtBeginDocument{%
    \ifx\InputIfFileExists\filehook@InputIfFileExists\else
     \PackageWarning{filehook}{Macro \string\InputIfFileExists\space got redefined after 'filehook' was loaded.^^J%
                                      Certain file hooks might now be dysfunctional!}
    \fi
}
```

6.5 Support for PGF Keys

```
\ProvidesPackage{pgf-filehook}[2010/01/07 v1.0 PGF keys for the filehook package]
\RequirePackage{filehook}
\RequirePackage{pgfkeys}

\pgfkeys{%
    /filehook/.is family,
    /filehook,
%
    EveryFile/.is family,
    EveryFile/AtBegin/.code={\AtBeginOfEveryFile{#1}},
    EveryFile/AtBegin/.value required,
    EveryFile/AtEnd/.code={\AtEndOfEveryFile{#1}},
    EveryFile/AtEnd/.value required,
%
    Files/.is family,
    Files/AtBegin/.code={\AtBeginOfFiles{#1}},
    Files/AtBegin/.value required,
    Files/AtEnd/.code={\AtEndOfFiles{#1}},
    Files/AtEnd/.value required,
%
    File/.is family,
    File/AtBegin/.code 2 args={\AtBeginOfFile{#1}{#2}},
    File/AtBegin/.value required,
    File/AtEnd/.code 2 args={\AtEndOfFile{#1}{#2}},
    File/AtEnd/.value required,
%
    Inputs/.is family,
    Inputs/AtBegin/.code={\AtBeginOfInputs{#1}},
    Inputs/AtBegin/.value required,
    Inputs/AtEnd/.code={\AtEndOfInputs{#1}},
    Inputs/AtEnd/.value required,
%
    InputFile/.is family,
    InputFile/AtBegin/.code 2 args={\AtBeginOfInputFile{#1}{#2}},
    InputFile/AtBegin/.value required,
    InputFile/AtEnd/.code 2 args={\AtEndOfInputFile{#1}{#2}},
    InputFile/AtEnd/.value required,
%
    Includes/.is family,
    Includes/AtBegin/.code={\AtBeginOfIncludes{#1}},
    Includes/AtBegin/.value required,
    Includes/AtEnd/.code={\AtEndOfIncludes{#1}},
    Includes/AtEnd/.value required,
    Includes/After/.code={\AfterIncludes{#1}},
    Includes/After/.value required,
%
    IncludeFile/.is family,
    IncludeFile/AtBegin/.code 2 args={\AtBeginOfIncludeFile{#1}{#2}},
    IncludeFile/AtBegin/.value required,
    IncludeFile/AtEnd/.code 2 args={\AtEndOfIncludeFile{#1}{#2}},
    IncludeFile/AtEnd/.value required,
    IncludeFile/After/.code 2 args={\AfterIncludeFile{#1}{#2}},
    IncludeFile/After/.value required,
%
    ClassFile/.is family,
    ClassFile/AtBegin/.code={\AtBeginOfClassFile#1},
    ClassFile/AtBegin/.value required,
    ClassFile/AtEnd/.code={\AtEndOfClassFile#1},
    ClassFile/AtEnd/.value required,
%
    PackageFile/.is family,
    PackageFile/AtBegin/.code={\AtBeginOfPackageFile#1},
    PackageFile/AtBegin/.value required,
    PackageFile/AtEnd/.code={\AtEndOfPackageFile#1},
    PackageFile/AtEnd/.value required,
}

\newcommand{\pgffilehook}{\pgfqkeys{/filehook}}
```