

The filehook Package

Martin Scharrer

martin@scharrer-online.de

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

Version v0.7 – 2020/02/02

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

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

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

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

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

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

2 Usage

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

Every File

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

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

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

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

All Files

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

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

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

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

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

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

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

`\InputIfFileExists:`

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

Include Files

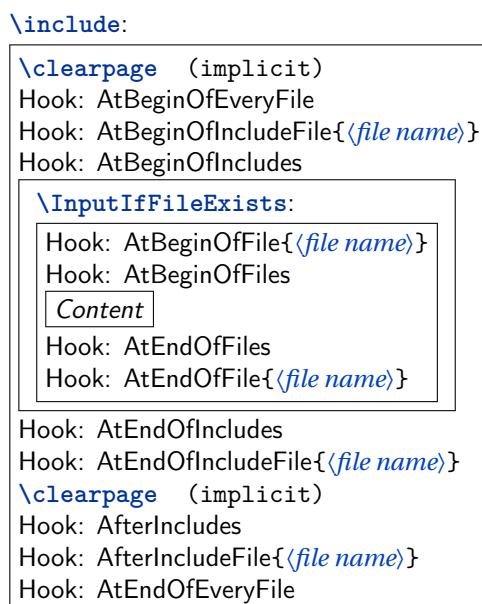
```
\AtBeginOfIncludes{<TEX code>}
\AtEndOfIncludes{<TEX code>}
\AfterIncludes{<TEX code>}
```

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

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

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

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



Input Files

```

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

```

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

```

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

```

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

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

```

\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 \TeX macro `\AtEndOfPackage`. Note that it is therefore executed after the ‘AtEndOfEveryFile’ hook. If the starred version is used and the package is already loaded the code is executed right away.

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

```

\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 ( $\TeX$  hook)
  Hook: AtEndOfPackageFile{<file name>}

```

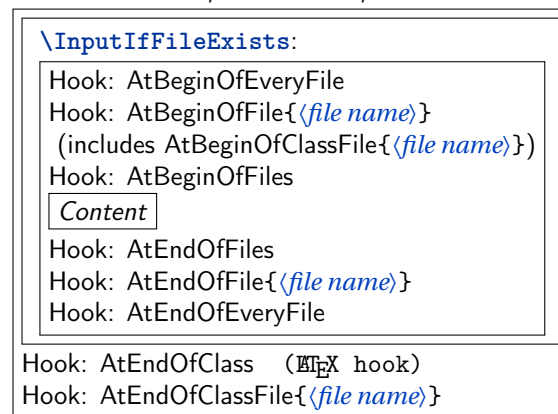
Class Files

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

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

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

`\documentclass/\LoadClass/\LoadClassWithOptions:`



2.1 Clearing Hooks

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

New in v0.5
2011/01/09

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

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

3 PGF Key Interface

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

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

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

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

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

```
\pgfkeys{/filehook/Inputs/AtBegin={<code>},/filehook/Inputs/AtEnd={<code>}}
```

can be shorten as

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

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

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

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

4 Compatibility Issues with Classes and other Packages

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

scrfile

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

fink

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

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

listings

The `listings` package uses `\input` inside `\lstinputlisting`. Therefore the `InputFile(s)` and `File(s)` hooks are also triggered for these files. Please note that this hooks are executing inside a verbatim environment. While the code in the hook is not affected (because it was added outside the verbatim environment), any further code read using any input macro (`\input`, `\@input`, `\@@input` (T_EX'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.

L^AT_EX's `\bibliography`

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

5 Upgrade Guide

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

Upgrade to v0.4 - 2011/01/03

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

6 Implementation

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

6.1 Options

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

6.2 General stuff

`\iffilehook@newfmt`

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

`\filehook@let`

```
    #1: <macro name 1>
    #2: <macro name 2>
15 \def\filehook@let#1#2{%
16     \expandafter\ifx\csname #2\space\endcsname\relax
17     \expandafter\let\csname #1\expandafter\endcsname\
        csname #2\endcsname
18     \else
19     \expandafter\def\csname #1\expandafter\endcsname\
        expandafter{\expandafter\protect\csname #1\
        space\endcsname}%
20     \expandafter\let\csname #1\space\expandafter\
        endcsname\csname #2\space\endcsname
21 \fi
22 }
```

`\filehook@glet`

#1: <macro name 1>
#2: <macro name 2>

```
23 \def\filehook@glet#1#2{%  
24   \expandafter\ifx\csname #2\space\endcsname\relax  
25     \expandafter\global\expandafter\let\csname #1\  
       expandafter\endcsname\csname #2\endcsname  
26   \else  
27     \expandafter\global\expandafter\def\csname #1\  
       expandafter\endcsname\expandafter{\expandafter\  
       \protect\csname #1\space\endcsname}%  
28     \expandafter\global\expandafter\let\csname #1\  
       space\expandafter\endcsname\csname #2\space\  
       endcsname  
29   \fi  
30 }
```

`\filehook@cmp`

#1: <macro name 1>
#2: <macro name 2>

Compare two macros definition including its space form in case of robust macros.

```
31 \def\filehook@cmp#1#2{%  
32   \expandafter\ifx\csname #2\space\endcsname\relax  
33     \expandafter\ifx\csname #1\expandafter\endcsname\  
       csname #2\endcsname  
34     \expandafter\expandafter\expandafter\  
       @firstoftwo  
35   \else  
36     \expandafter\expandafter\expandafter\  
       @secondoftwo  
37   \fi  
38 \else  
39   \expandafter\ifx\csname #1\space\expandafter\  
       endcsname\csname #2\space\endcsname  
40   \expandafter\expandafter\expandafter\  
       @firstoftwo  
41   \else  
42     \expandafter\expandafter\expandafter\  
       @secondoftwo  
43   \fi  
44 \fi  
45 }
```

6.3 Initialisation of Hooks

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

`\filehook@csuse`

```
46 \begingroup
47 \gdef\filehook@csuse#1{\ifcsname #1\endcsname\csname /
    #1\expandafter\endcsname\fi}
48 \expandafter\ifx\csname csuse\endcsname\relax
49     \expandafter\ifx\csname ifcsname\endcsname\relax
50         \gdef\filehook@csuse#1{\expandafter\ifx\
            csname #1\endcsname\relax\else\csname #1\
            expandafter\endcsname\fi}
51     \fi
52 \else
53     \global\let\filehook@csuse\csuse
54 \fi
55 \endgroup
```

`\filehook@include@atbegin`

```
56 \def\filehook@include@atbegin#1{%
57     \filehook@let{InputIfFileExists}{\
        filehook@@InputIfFileExists}%
58     \filehook@csuse{\filehook@include@atbegin@#1}%
59     \filehook@include@@@atbegin
60 }
```

`\filehook@include@@@atbegin`

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

`\filehook@include@atend`

```
62 \def\filehook@include@atend#1{%
63     \filehook@include@@@atend
64     \filehook@csuse{\filehook@include@atend@#1}%
65 }
```

`\filehook@include@@@atend`

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

`\filehook@include@after`

```
67 \def\filehook@include@after#1{%  
68   \filehook@include@@after  
69   \filehook@csuse{\filehook@include@after@#1}%  
70 }
```

`\filehook@include@@after`

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

`\filehook@input@atbegin`

```
72 \def\filehook@input@atbegin#1{%  
73   \filehook@let{InputIfFileExists}{/  
       filehook@@InputIfFileExists}%  
74   \filehook@csuse{\filehook@input@atbegin@/  
       filehook@ensureext{#1}}%  
75   \filehook@input@@atbegin  
76 }
```

`\filehook@input@@atbegin`

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

`\filehook@input@atend`

```
78 \def\filehook@input@atend#1{%  
79   \filehook@input@@atend  
80   \filehook@csuse{\filehook@input@atend@/  
       filehook@ensureext{#1}}%  
81 }
```

`\filehook@input@@atend`

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

`\filehook@atbegin`

```
83 \def\filehook@atbegin#1{%  
84   \filehook@csuse{\filehook@atbegin@\  
      filehook@ensureext{#1}}%  
85   \filehook@@atbegin  
86 }
```

`\filehook@@atbegin`

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

`\filehook@atend`

```
88 \def\filehook@atend#1{%  
89   \filehook@@atend  
90   \filehook@csuse{\filehook@atend@\filehook@ensureext/  
      {#1}}%  
91 }
```

`\filehook@@atend`

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

`\filehook@every@atbegin`

```
93 \def\filehook@every@atbegin#1{%  
94   \filehook@every@@atbegin  
95 }
```

`\filehook@every@@atbegin`

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

`\filehook@every@atend`

```
97 \def\filehook@every@atend#1{%  
98   \filehook@every@@atend  
99 }
```

`\filehook@every@@atend`

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

6.4 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.

```
101 \def\filehook@include@atbegin@{/  
    filehook@include@atbegin@}  
102 \def\filehook@include@atend@{filehook@include@atend@}  
103 \def\filehook@include@after@{filehook@include@after@}  
104 \def\filehook@input@atbegin@{filehook@input@atbegin@}  
105 \def\filehook@input@atend@{filehook@input@atend@}  
106 \def\filehook@input@after@{filehook@input@after@}  
107 \def\filehook@atbegin@{filehook@atbegin@}  
108 \def\filehook@atend@{filehook@atend@}  
109 \def\filehook@after@{filehook@after@}
```

`\filehook@append`

Uses default \TeX macro.

```
110 \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.

```
111 \long\def\filehook@appendwarg#1#2{%  
112   \begingroup  
113     \toks@\expandafter{#1{##1}#2}%  
114     \edef\@tempa{\the\toks@}%  
115     \expandafter\gdef\expandafter#1\expandafter##\/  
        expandafter1\expandafter{\@tempa}%  
116   \endgroup  
117 }
```


`\filehook@prefix`

Prefixes code to a hook.

```
118 \long\def\filehook@prefix#1#2{%
119   \begingroup
120     \@temptokena{#2}%
121     \toks@\expandafter{#1}%
122     \xdef#1{\the\@temptokena\the\toks@}%
123   \endgroup
124 }
```

`\filehook@prefixwarg`

Prefixes code with an argument to a hook.

```
125 \long\def\filehook@prefixwarg#1#2{%
126   \begingroup
127     \@temptokena{#2}%
128     \toks@\expandafter{#1{##1}}%
129     \edef\@tempa{\the\@temptokena\the\toks@}%
130     \expandafter\gdef\expandafter#1\expandafter##\
131       expandafter1\expandafter{\@tempa}%
132   \endgroup
133 }
```

`\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.

```
133 \def\filehook@addtohook#1#2#3{%
134   \begingroup
135   \edef\@tempa{#3}%
136   \edef\@tempa{#2\filehook@ensureext{\@tempa}}%
137   \ifundefined{\@tempa}{\global\@namedef{\@tempa/
138     }{}}{}%
139   \expandafter\endgroup
140   \expandafter#1\curname\@tempa\endcurname
141 }
```

User Level Macros

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

`\AtBeginOfIncludes`

```
141 \newcommand*\AtBeginOfIncludes{%  
142   \filehook@append\filehook@include@@atbegin  
143 }
```

`\AtEndOfIncludes`

```
144 \newcommand*\AtEndOfIncludes{%  
145   \filehook@prefix\filehook@include@@atend  
146 }
```

`\AfterIncludes`

```
147 \newcommand*\AfterIncludes{%  
148   \filehook@prefix\filehook@include@@after  
149 }
```

`\AtBeginOfIncludeFile`

```
150 \newcommand*\AtBeginOfIncludeFile [1]{%  
151   \filehook@addtohook\filehook@append\  
       filehook@include@atbegin@{\filehook@ensuretex/  
       {#1}}}%  
152 }
```

`\AtEndOfIncludeFile`

```
153 \newcommand*\AtEndOfIncludeFile [1]{%  
154   \filehook@addtohook\filehook@prefix\  
       filehook@include@atend@{\filehook@ensuretex{#1}}/  
       %  
155 }
```

`\AfterIncludeFile`

```
156 \newcommand*\AfterIncludeFile [1]{%  
157   \filehook@addtohook\filehook@prefix\  
       filehook@include@after@{\filehook@ensuretex{#1}}/  
       %  
158 }
```

\AtBeginOfInputs

```
159 \newcommand*\AtBeginOfInputs{%  
160   \filehook@append\filehook@input@@atbegin  
161 }
```

\AtEndOfInputs

```
162 \newcommand*\AtEndOfInputs{%  
163   \filehook@prefix\filehook@input@@atend  
164 }
```

\AtBeginOfInputFile

```
165 \newcommand*\AtBeginOfInputFile{%  
166   \filehook@addtohook\filehook@append\/  
    filehook@input@atbegin@  
167 }
```

\AtEndOfInputFile

```
168 \newcommand*\AtEndOfInputFile{%  
169   \filehook@addtohook\filehook@prefix\/  
    filehook@input@atend@  
170 }
```

\AtBeginOfFiles

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

\AtEndOfFiles

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

`\AtBeginOfEveryFile`

```
177 \newcommand*\AtBeginOfEveryFile{%
178   \filehook@append\filehook@every@@atbegin
179 }
```

`\AtEndOfEveryFile`

```
180 \newcommand*\AtEndOfEveryFile{%
181   \filehook@prefix\filehook@every@@atend
182 }
```

`\AtBeginOfFile`

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

`\AtEndOfFile`

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

`\AtBeginOfClassFile`

```
189 \newcommand*\AtBeginOfClassFile{%
190   \@ifnextchar*
191     {\AtBeginOfXFile@star\@clsextension}%
192     {\AtBeginOfXFile@normal\@clsextension}%
193 }
```

`\AtBeginOfPackageFile`

```
194 \newcommand*\AtBeginOfPackageFile{%
195   \@ifnextchar*
196     {\AtBeginOfXFile@star\@pkgextension}%
197     {\AtBeginOfXFile@normal\@pkgextension}%
198 }
```

`\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.

```
199 \def\AtBeginOfXFile@star#1*#2{%  
200     \@ifl@aded{#1}{#2}%  
201     {\@firstofone}%  
202     {\AtBeginOfXFile@normal{#1}{#2}}%  
203 }
```

`\AtBeginOfXFile@normal`

#1: extension

#2: name

```
204 \def\AtBeginOfXFile@normal#1#2{%  
205     \AtBeginOfFile{#2.#1}%  
206 }
```

`\AtEndOfClassFile`

```
207 \newcommand*\AtEndOfClassFile{%  
208     \@ifnextchar*  
209     {\AtEndOfXFile@star\@clsextension}%  
210     {\AtEndOfXFile@normal\@clsextension}%  
211 }
```

`\AtEndOfPackageFile`

```
212 \newcommand*\AtEndOfPackageFile{%  
213     \@ifnextchar*  
214     {\AtEndOfXFile@star\@pkgextension}%  
215     {\AtEndOfXFile@normal\@pkgextension}%  
216 }
```

`\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.

```

217 \def\AtEndOfXFile@star#1*#2{%
218     \@ifl@aded{#1}{#2}%
219     {\@firstofone}%
220     {\AtEndOfXFile@normal{#1}{#2}}%
221 }

```

`\AtEndOfXFile@normal`

#1: extension

#2: name

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

```

222 \long\def\AtEndOfXFile@normal#1#2#3{%
223     \AtEndOfFile{#2.#1}{\AtEndOfPackage{#3}}%
224 }

```

`\ClearHook`

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

```

225 \newcommand*\ClearHook{%
226     \begingroup
227     \def\filehook@prefix##1#2{%
228         \gdef##1{}%
229         \endgroup
230     }%
231     \let\filehook@append\filehook@prefix
232 }

```

6.5 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@`

```

233 \let\filehook@orig@@input@\@input@

```

`\filehook@orig@@iinput`

```

234 \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.

```
235 \def\@input@#1{%
236   \@ifnextchar\clearpage
237   {%
238     \filehook@every@atbegin{#1}%
239     \filehook@include@atbegin{#1}%
240     \filehook@orig@@input@{#1}%
241     \filehook@include@atend{#1}%
242     \clearpage
243     \filehook@include@after{#1}%
244     \filehook@every@atend{#1}%
245     \@gobble
246   }%
247   {\filehook@orig@@input@{#1}}%
248 }
```

`\@iinput`

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

```
249 \def\filehook@@iinput#1{%
250   \filehook@every@atbegin{#1}%
251   \filehook@input@atbegin{#1}%
252   \filehook@orig@@iinput{#1}%
253   \filehook@input@atend{#1}%
254   \filehook@every@atend{#1}%
255 }
256 \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.

```
257 \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.

```

258 \def\filehook@ensureext#1{%
259     \expandafter\filehook@@ensureext#1\empty.tex\
        empty\empty
260 }

```

`\filehook@@ensureext`

```

261 \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.

```

262 \def\filehook@ensuretex#1{%
263     \expandafter\filehook@@ensuretex#1\empty.tex\
        empty\empty
264 }

```

`\filehook@@ensuretex`

```

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

```

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

`\latex@InputIfFileExists`

Standard \TeX definition of `\InputIfFileExists`.

```

266 \iffilehook@newfmt
267 \expandafter\def\expandafter\latex@InputIfFileExists\
    \expandafter{%
268     \expandafter\protect\csname InputIfFileExists\space\
        \endcsname
269 }
270 \expandafter\long\expandafter\def\csname \
    latex@InputIfFileExists\space\endcsname#1#2{%
271     \IfFileExists{#1}%
272     {%
273         \expandafter\@swaptwoargs\expandafter
274         {\@filef@und}{#2\@addtofilelist{#1}\@@input}}
275     \else
276     \long\def\latex@InputIfFileExists#1#2{%

```



```

277 \IfFileExists{#1}%
278   {#2\@addtofilelist{#1}%
279   \@@input\@filef@und
280   }%
281 }
282 \fi

```

`\filehook@default@InputIfFileExists`

```

283 \DeclareRobustCommand\
284   filehook@default@InputIfFileExists [2]{%
285   \IfFileExists{#1}%
286     {\expandafter\filehook@swap
287     \expandafter{\@filef@und}%
288     {#2\@addtofilelist{#1}%
289     \filehook@every@atbegin{#1}%
290     \filehook@atbegin{#1}%
291     \@@input}%
292     \filehook@atend{#1}%
293     \filehook@every@atend{#1}%
294   }%

```

Make sure definition is global:

```

295 \filehook@glet{filehook@default@InputIfFileExists}{\
296   filehook@default@InputIfFileExists}%

```

`\filehook@@default@InputIfFileExists`

```

296 \DeclareRobustCommand\
297   filehook@@default@InputIfFileExists [2]{%
298   \filehook@let{InputIfFileExists}{\
299   filehook@InputIfFileExists}%
300   \IfFileExists{#1}%
301     {\expandafter\filehook@swap
302     \expandafter{\@filef@und}%
303     {#2\@addtofilelist{#1}%
304     \filehook@atbegin{#1}%
305     \@@input}%
306     \filehook@atend{#1}%
307   }%

```

Make sure definition is global:

```

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

```

`\InputIfFileExists`

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

```
308 \AtBeginOfPackageFile{scrfile}{%
309   \filehook@let{InputIfFileExists}{%
      latex@InputIfFileExists}%
310 }%
311 \AtEndOfPackageFile*{scrfile}{%
312   \RequirePackage{filehook-scrfile}%
313 }%
```

Fink:

```
314 \AtBeginOfPackageFile*{fink}{%
315   \RequirePackage{kvoptions}%
316   \begingroup
317   \filehook@let{InputIfFileExists}{%
      latex@InputIfFileExists}%
318 }%
319 \AtEndOfPackageFile*{fink}{%
320   \edef\@tempa{\noexpand\PassOptionsToPackage{
      mainext=\fink@mainext,maindir=\fink@maindir}{%
      currfile}}%
321   \expandafter\endgroup\@tempa
322   \RequirePackage{filehook-fink}%
323 }%
```

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

```
324 \AtBeginOfClassFile{memoir}{%
325   \filehook@let{InputIfFileExists}{%
      latex@InputIfFileExists}%
326   \let\@input\filehook@orig@@input
327 }%
328 \AtEndOfClassFile*{memoir}{%
329   \let\@input\filehook@@input
330   \RequirePackage{filehook-memoir}%
331 }%
```

Finally, if no specific alternate definition is detected the original `TeX` definition is checked for and an 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.

```
332 \filehook@cmp{InputIfFileExists}{%
      filehook@InputIfFileExists}%
333 }% already set up
334 {%
```

```

335 \filehook@cmp{InputIfFileExists}{\
      latex@InputIfFileExists}%
336 {%
337   \filehook@let{filehook@InputIfFileExists}{\
      filehook@default@InputIfFileExists}%
338   \filehook@let{filehook@@InputIfFileExists}{\
      filehook@@@default@InputIfFileExists}%
339   \filehook@let{InputIfFileExists}{\
      filehook@InputIfFileExists}%
340 }%
341 {%
342   \iffilehook@force
343     \filehook@let{filehook@InputIfFileExists}{\
      filehook@default@InputIfFileExists}%
344     \filehook@let{filehook@@InputIfFileExists}{\
      filehook@@@default@InputIfFileExists}%
345     \filehook@let{InputIfFileExists}{\
      filehook@InputIfFileExists}%
346     \PackageWarning{filehook}{Detected unknown /
      definition of \string\InputIfFileExists/
      .^^J%
347                                     The 'force' /
                                     option of ' /
                                     filehook' is /
                                     in effect. /
                                     Macro is /
                                     overwritten /
                                     with default!}%
                                     %
348   \else
349     \PackageError{filehook}{Detected unknown /
      definition of \string\InputIfFileExists/
      .^^J%
350                                     Use the 'force' /
                                     option of ' /
                                     filehook' to /
                                     overwrite it.}}}%
                                     %
351   \fi
352 }%
353 }%

354 \AtBeginDocument{%
355   % Check if definition got changed again. For the /
      new LaTeX format we check again \ /
      InputIfFileExists<space>,
356   % for the old format to \InputIfFileExists /
      directly.
357   \filehook@cmp{InputIfFileExists}{\
      filehook@InputIfFileExists}{}%

```

```

358     \PackageWarning{filehook}{Macro \string\
        InputIfFileExists\space got redefined /
        after 'filehook' was loaded.^^J%
359                                     Certain file hooks /
                                                might now be /
                                                dysfunctional!}%
360     }%
361 }

362 %<!COPYRIGHT>
363 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
364 \ProvidesPackage{filehook-memoir}[2020/02/02 v0.2 /
        filehook patch for memoir class]

365 \RequirePackage{filehook}
366 \begingroup

```

\memoir@InputIfFileExists

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

```

367 \ifcsname InputIfFileExists\space\endcsname
368   \DeclareRobustCommand \memoir@InputIfFileExists [2]{/
        %
369   \IfFileExists{#1}%
370   {%
371     \expandafter\@swaptwoargs\expandafter
372     {\@filef@und\m@matendf{#1}\killm@matf{#1}}{%
373       #2\@addtofilelist{#1}\m@matbeginf{#1}\@@input/
        %
374     }%
375   }%
376 }
377 \else
378   % Old definition
379 \renewcommand{\memoir@InputIfFileExists}[2]{%
380   \IfFileExists{#1}%
381     {#2\@addtofilelist{#1}\m@matbeginf{#1}%
382       \@@input \@filef@und
383       \m@matendf{#1}%
384       \killm@matf{#1}}%
385 }
386 \fi

387 \@tempswafalse
388 \filehook@cmp{InputIfFileExists}{/
        filehook@InputIfFileExists}%
389 {\@tempswatrue}%
390 {%

```

```

391 \filehook@cmp{InputIfFileExists}{\
      memoir@InputIfFileExists}%
392 {\@tempwattrue}%
393 {}%
394 }%
395
396 \if@tempswa
397 \filehook@glet{filehook@InputIfFileExists}{\
      filehook@default@InputIfFileExists}%
398 \filehook@glet{filehook@@InputIfFileExists}{\
      filehook@@default@InputIfFileExists}%
399 \filehook@glet{InputIfFileExists}{\
      filehook@InputIfFileExists}%
400 \filehook@appendwarg\filehook@atbegin{\m@matbeginf/
      {#1}}}%
401 \filehook@prefixwarg\filehook@atend{\m@matendf{#1}\
      killm@matf{#1}}}%
402 \PackageInfo{filehook}{Detected 'memoir' class: the
      memoir hooks will be moved to the 'At...OfFiles\
      ' hooks}
403 \else
404 \iffilehook@force
405 \filehook@glet{filehook@InputIfFileExists}{\
      filehook@default@InputIfFileExists}%
406 \filehook@glet{filehook@@InputIfFileExists}{\
      filehook@@default@InputIfFileExists}%
407 \filehook@glet{InputIfFileExists}{\
      filehook@InputIfFileExists}%
408 \PackageWarning{filehook}{Detected 'memoir' class\
      with unknown definition of \string\
      InputIfFileExists.^^J%
409                                     The 'force' option of '\
                                     filehook' is in \
                                     effect. Macro is \
                                     overwritten with \
                                     default!}%
410 \else
411 \PackageError{filehook}{Detected 'memoir' class \
      with unknown definition of \string\
      InputIfFileExists.^^J%
412                                     Use the 'force' option of\
                                     'filehook' to \
                                     overwrite it.}}}%
413 \fi
414 \fi
415 \endgroup
416 %<!COPYRIGHT>
417 \NeedsTeXFormat{LaTeX2e}[1999/12/01]

```

```

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

419 \begingroup
420
421 \long\def\patch#1\def\lst@next#2#3\endpatch{%
422     \toks@{#2}%
423     \edef\@tempa{\the\toks@}%
424     \def\@tempb{\input{####1}}%
425     \ifx\@tempa\@tempb
426         \gdef\lst@InputListing##1{#1\def\lst@next{\%/
            @input{##1}}#3}%
427     \else
428         \PackageWarning{filehook-listings}{To-be- /
            patched code in macro \string\ /
            lst@InputListing was not found!}%
429     \fi
430 }
431
432 \@ifundefined{lst@InputListing}{%
433     \PackageWarning{filehook-listings}{To-be-patched /
        Macro \string\lst@InputListing not found!}%
434 }{}
435
436 \expandafter\patch\lst@InputListing{#1}\endpatch
437
438 \endgroup

439 %<!COPYRIGHT>
440 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
441 \ProvidesPackage{filehook-scrfile}[2020/02/02 v0.2 /
    filehook patch for scrfile package]
442 \RequirePackage{filehook}
443 \begingroup

```

<code>\scrfile@InputIfFileExists</code>

```

444 \expandafter\def\expandafter\ /
    scrfile@InputIfFileExists\expandafter{%
445     \expandafter\protect\csname InputIfFileExists\space /
        \endcsname
446 }
447 \expandafter\long\expandafter\def\csname /
    scrfile@InputIfFileExists\space\endcsname#1#2{%
448     \begingroup\expandafter\expandafter\expandafter\ /
        endgroup
449     \expandafter\ifx\csname #1-@alias\endcsname\relax

```

```

450     \expandafter \@secondoftwo
451 \else
452     \scr@replacefile@msg{\csname #1-@alias\endcsname/
        }{#1}%
453     \expandafter \@firstoftwo
454 \fi
455 {%
456     \expandafter\InputIfFileExists\expandafter{\
        csname
457 #1-@alias\endcsname}{#2}%
458     }%
459     {\IfFileExists{#1}{%
460         \expandafter\scr@input@withhook\expandafter{\
            @filef@und}{#1}{#2}}%
461     }%
462 }

```

`\filehook@scrfile@InputIfFileExists`

```

463 \DeclareRobustCommand\
    filehook@scrfile@InputIfFileExists [2]{%
464     \begingroup\expandafter\expandafter\expandafter\
        endgroup
465     \expandafter\ifx\csname #1-@alias\endcsname\relax
466         \expandafter \@secondoftwo
467     \else
468         \scr@replacefile@msg{\csname #1-@alias\endcsname/
            }{#1}%
469         \expandafter \@firstoftwo
470     \fi
471     {%
472         \expandafter\InputIfFileExists\expandafter{\
            csname
473             #1-@alias\endcsname}{#2}%
474         }%
475         {\IfFileExists{#1}{%
476             \expandafter\filehook@swap
477             \expandafter{\@filef@und}%
478             {\scr@load@hook{before}{#1}%
479             #2\@addtofilelist{#1}%
480             \filehook@every@atbegin{#1}%
481             \filehook@atbegin{#1}%
482             \@@input}%
483             \filehook@atend{#1}%
484             \filehook@every@atend{#1}%
485             \scr@load@hook{after}{#1}%
486             }}%
487     }

```

```

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

```

`\filehook@@scrfile@InputIfFileExists`

```

489 \DeclareRobustCommand\
      filehook@@scrfile@InputIfFileExists [2]{%
490 \filehook@let{InputIfFileExists}{\
      filehook@InputIfFileExists}%
491 \begingroup\expandafter\expandafter\expandafter\
      endgroup
492 \expandafter\ifx\csname #1-@alias\endcsname\relax
493 \expandafter\@secondoftwo
494 \else
495 \scr@replacefile@msg{\csname #1-@alias\endcsname\
      }{#1}%
496 \expandafter\@firstoftwo
497 \fi
498 {%
499 \expandafter\InputIfFileExists\expandafter{\
      csname
500 #1-@alias\endcsname}{#2}%
501 }%
502 {\IfFileExists{#1}{%
503 \expandafter\filehook@swap
504 \expandafter{\@filef@und}%
505 {\scr@load@hook{before}{#1}%
506 #2\@addtofilelist{#1}%
507 \filehook@atbegin{#1}%
508 \@@input}%
509 \filehook@atend{#1}%
510 \scr@load@hook{after}{#1}%
511 }}%
512 }
513 \filehook@glet{filehook@@scrfile@InputIfFileExists}{\
      filehook@@scrfile@InputIfFileExists}%

```

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

```

514 \@tempswafalse
515 \filehook@cmp{InputIfFileExists}{\
      filehook@InputIfFileExists}%
516 {\@tempswatrue}%
517 {%
518 \filehook@cmp{InputIfFileExists}{\
      scrfile@InputIfFileExists}%

```



```

519         {\@tempswatrue}%
520         {}%
521     }%
522
523 \if@tempswa
524     \filehook@glet{filehook@InputIfFileExists}{/
525         filehook@scrfile@InputIfFileExists}%
526     \filehook@glet{filehook@@InputIfFileExists}{/
527         filehook@@scrfile@InputIfFileExists}%
528     \filehook@glet{InputIfFileExists}{/
529         filehook@InputIfFileExists}%
530     \PackageInfo{filehook}{Package 'scrfile' detected /
531         and compensated for}%
532 \else
533     \iffilehook@force
534         \filehook@glet{filehook@InputIfFileExists}{/
535             filehook@scrfile@InputIfFileExists}%
536         \filehook@glet{filehook@@InputIfFileExists}{/
537             filehook@@scrfile@InputIfFileExists}%
538         \filehook@glet{InputIfFileExists}{/
539             filehook@InputIfFileExists}%
540         \PackageWarning{filehook}{Detected 'scrfile' /
541             package with unknown definition of \string\
542             InputIfFileExists.^~J%
543
544             The 'force' option of '
545             filehook' is in /
546             effect. Macro is /
547             overwritten with /
548             default!}%
549     \else
550         \PackageError{filehook}{Detected 'scrfile' /
551             package with unknown definition of \string\
552             InputIfFileExists.^~J%
553
554             Use the 'force' option of /
555             'filehook' to /
556             overwrite it.}{}%
557
558 \fi
559 \fi
560
561 \endgroup
562
563 %<!COPYRIGHT>
564 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
565 \ProvidesPackage{filehook-fink}[011/01/03 v0.1 /
566     filehook compatibility code for fink package]
567
568 \RequirePackage{filehook}
569 \RequirePackage{currfile}%
570
571 \begingroup

```

```

548
549 \long\def\fink@old@InputIfFileExists#1#2{%
550   \IfFileExists{#1}{%
551     #2\@addtofilelist{#1}%
552     \fink@prepare{#1}%
553     \expandafter\fink@input%
554     \expandafter\fink@restore\expandafter{\finkpath}}%
555   }
556
557 \long\def\fink@new@InputIfFileExists#1#2{%
558   \IfFileExists{#1}{%
559     #2\@addtofilelist{#1}%
560     \edef\fink@before{\noexpand\fink@input{#1}}%
561     \edef\fink@after{\noexpand\fink@restore{\finkpath}%
562       }}%
563     \expandafter\fink@before\fink@after}%
564   }
565 \ifcase
566   \ifx\InputIfFileExists\filehook@InputIfFileExists/
567     0\else
568     \ifx\InputIfFileExists\latex@InputIfFileExists /
569       1\else
570     \ifx\InputIfFileExists\fink@new@InputIfFileExists/
571       1\else
572     \ifx\InputIfFileExists\fink@old@InputIfFileExists/
573       1\else
574       1%
575     \fi\fi\fi\fi
576 \relax
577 \or
578   \global\let\filehook@InputIfFileExists\
579     filehook@default@InputIfFileExists
580   \global\let\filehook@@InputIfFileExists\
581     filehook@@default@InputIfFileExists
582   \global\let\InputIfFileExists\
583     filehook@InputIfFileExists
584   \PackageInfo{filehook-fink}{Package 'fink' detected/
585     and replaced by 'currfile'}%
586 \else
587   \iffilehook@force
588     \global\let\filehook@InputIfFileExists\
589       filehook@default@InputIfFileExists
590     \global\let\filehook@@InputIfFileExists\
591       filehook@@default@InputIfFileExists
592     \global\let\InputIfFileExists\
593       filehook@InputIfFileExists
594     \PackageWarning{filehook-fink}{Detected 'fink' /
595       package with unknown definition of \string\

```

```

InputIfFileExists.^~J%
584                                     The 'force' option of '/'
                                     filehook' is in /
                                     effect. Macro is /
                                     overwritten with /
                                     default!}%
585 \else
586   \PackageError{filehook-fink}{Detected 'fink' /
   package with unknown definition of \string\
   InputIfFileExists.^~J%
587                                     Use the 'force' /
                                     option of '/'
                                     filehook' to /
                                     overwrite it.}{}%
588 \fi
589 \fi
590
591 \endgroup

```

6.6 Support for PGF Keys

```

592 \ProvidesPackage{pgf-filehook}[2010/01/07 v1.0 PGF /
   keys for the filehook package]
593 \RequirePackage{filehook}
594 \RequirePackage{pgfkeys}
595
596 \pgfkeys{%
597   /filehook/.is family,
598   /filehook,
599   %
600   EveryFile/.is family,
601   EveryFile/AtBegin/.code={\AtBeginOfEveryFile/
   {#1}},
602   EveryFile/AtBegin/.value required,
603   EveryFile/AtEnd/.code={\AtEndOfEveryFile{#1}},
604   EveryFile/AtEnd/.value required,
605   %
606   Files/.is family,
607   Files/AtBegin/.code={\AtBeginOfFiles{#1}},
608   Files/AtBegin/.value required,
609   Files/AtEnd/.code={\AtEndOfFiles{#1}},
610   Files/AtEnd/.value required,
611   %
612   File/.is family,
613   File/AtBegin/.code 2 args={\AtBeginOfFile/
   {#1}{#2}},
614   File/AtBegin/.value required,
615   File/AtEnd/.code 2 args={\AtEndOfFile{#1}{#2}},
616   File/AtEnd/.value required,

```

```

617 %
618 Inputs/.is family,
619 Inputs/AtBegin/.code={\AtBeginOfInputs{#1}},
620 Inputs/AtBegin/.value required,
621 Inputs/AtEnd/.code={\AtEndOfInputs{#1}},
622 Inputs/AtEnd/.value required,
623 %
624 InputFile/.is family,
625 InputFile/AtBegin/.code 2 args={\
        AtBeginOfInputFile{#1}{#2}},
626 InputFile/AtBegin/.value required,
627 InputFile/AtEnd/.code 2 args={\AtEndOfInputFile/
        {#1}{#2}},
628 InputFile/AtEnd/.value required,
629 %
630 Includes/.is family,
631 Includes/AtBegin/.code={\AtBeginOfIncludes{#1}},
632 Includes/AtBegin/.value required,
633 Includes/AtEnd/.code={\AtEndOfIncludes{#1}},
634 Includes/AtEnd/.value required,
635 Includes/After/.code={\AfterIncludes{#1}},
636 Includes/After/.value required,
637 %
638 IncludeFile/.is family,
639 IncludeFile/AtBegin/.code 2 args={\
        AtBeginOfIncludeFile{#1}{#2}},
640 IncludeFile/AtBegin/.value required,
641 IncludeFile/AtEnd/.code 2 args={\
        AtEndOfIncludeFile{#1}{#2}},
642 IncludeFile/AtEnd/.value required,
643 IncludeFile/After/.code 2 args={\AfterIncludeFile/
        {#1}{#2}},
644 IncludeFile/After/.value required,
645 %
646 ClassFile/.is family,
647 ClassFile/AtBegin/.code={\AtBeginOfClassFile#1},
648 ClassFile/AtBegin/.value required,
649 ClassFile/AtEnd/.code={\AtEndOfClassFile#1},
650 ClassFile/AtEnd/.value required,
651 %
652 PackageFile/.is family,
653 PackageFile/AtBegin/.code={\AtBeginOfPackageFile/
        #1},
654 PackageFile/AtBegin/.value required,
655 PackageFile/AtEnd/.code={\AtEndOfPackageFile#1},
656 PackageFile/AtEnd/.value required,
657 }
658
659 \newcommand{\pgffilehook}{\pgfqkeys{/filehook}}

```