The *filehook* Package

Martin Scharrer

martin@scharrer-online.de

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

Version v0.1 – 2010/04/08

Abstract

This small 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{`⟨*code*⟩`}`.

2 Usage

This package provides three groups of hooks: for file read using `\input`, for files read using `\include` and for all read files (i.e. all files read using `\InputIfFileExists`, which includes package and class files and files falling into the first two groups). All groups include a 'AtBegin' and a 'AtEnd' macro. The `\include` group has 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.

Each group includes general and file specific hooks. The general hooks are executed for every file of this group and provide the file name as argument `#1`. The file specific ones are only executed for a certain file.

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.

Include Files

\AtBeginOfIncludes
\AtEndOfIncludes
\AfterIncludes

All these macro take one argument (some TEX code) which is added to the specific hook for files read using `\include`. The code can use the macro argument `#1` which will be expanded to the include file name, i.e. the hooks are macros with one argument which will be the file name. As described above the 'AtEnd' hook is executed before and the 'After' hook is executed after the trailing `\clearpage`. Material which should be (still) valid in the page header or footer of the last page of such an file should therefore use the 'After' hook.

\AtBeginOfIncludeFile
\AtEndOfIncludeFile
\AfterIncludeFile

These file-specific macros take the two arguments `{`⟨*file name*⟩`}{`⟨*code*⟩`}`. The ⟨*code*⟩ is only executed for the file with the given ⟨*file name*⟩ and only if it is read using `\include`. It is not allowed to use macro arguments inside the code. The ⟨*file name*⟩ should be identical to the name used for `\include` and not include the '`.tex`' extension.

Input Files

\AtBeginOfInputs
\AtEndOfInputs

Like the `\...OfIncludes{`⟨*code*⟩`}` macros above, just for file read using `\input`. Again, the macro argument `#1` can be used inside the ⟨*code*⟩ and will be expanded to the file name.

\AtBeginOfInputFile
\AtEndOfInputFile

Like the `\...OfIncludeFile{`⟨*file name*⟩`}{`⟨*code*⟩`}` macros above, just for file read using `\input`. Here the ⟨*file name*⟩ should include the file extension! The ⟨*code*⟩ must not include any macro arguments (`#1`).

All Files

\AtBeginOfFiles
\AtEndOfFiles

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 `\IfFileExists{`⟨*file name*⟩`}{\@input\@filef@und}{}` instead.

\AtBeginOfFile
\AtEndOfFile

Like the `\...OfIncludeFile{`⟨*file name*⟩`}{`⟨*code*⟩`}` macros above, just for 'all' read files. Here the ⟨*file name*⟩ should include the file extension! The ⟨*code*⟩ must not include any macro arguments (`#1`).

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.

3 Implementation

3.1 Installation of Hooks

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

First the original definitions are saved away.

```
\let\filehook@orig@@input@\@input@
\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@include@atbegin{#1}%
     \filehook@orig@@input@{#1}%
     \filehook@include@atend{#1}%
     \clearpage
     \filehook@include@after{#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\@iinput#1{%
  \filehook@input@atbegin{#1}%
  \filehook@orig@@iinput{#1}%
  \filehook@input@atend{#1}%
}
```

`\InputIfFileExists` This macro is redefined for the general file hooks. The original definition is not saved away and called by the new definition, because of the existing complexity. The hooks must be places around the actual input macro (`\@@input`). To be compatible with the similar `fink` package its code inserted into this macro is kept if the package was loaded beforehand.

```
\ifcase0%
  \ifx\fink@input\@undefined 1\else
  \ifx\fink@input\relax 1\fi\fi
\relax
\long\def\InputIfFileExists#1#2{%
  \IfFileExists{#1}{%
    #2\@addtofilelist{#1}%
    \fink@prepare{#1}%
    \filehook@atbegin{#1}%
    \expandafter\fink@input%
    \expandafter\fink@restore\expandafter{\finkpath}%
    \filehook@atend{#1}%
  }%
}
\else
\long\def\InputIfFileExists#1#2{%
```

```
  \IfFileExists{#1}%
    {#2\@addtofilelist{#1}%
     \filehook@atbegin{#1}%
     \@@input\@filef@und
     \filehook@atend{#1}%
    }%
}
\fi
```

3.2 Initialisation of Hooks

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

\filehook@include@atbegin
\filehook@include@atend
\filehook@include@after

```
\def\filehook@include@atbegin#1{%
  \@nameuse{\filehook@include@atbegin@#1}%
}
\def\filehook@include@atend#1{%
  \@nameuse{\filehook@include@atend@#1}%
}
\def\filehook@include@after#1{%
  \@nameuse{\filehook@include@after@#1}%
}
```

\filehook@input@atbegin
\filehook@input@atend

```
\def\filehook@input@atbegin#1{%
  \@nameuse{\filehook@input@atbegin@#1}%
}
\def\filehook@input@atend#1{%
  \@nameuse{\filehook@input@atend@#1}%
}
```

\filehook@atbegin
\filehook@atend

```
\def\filehook@atbegin#1{%
  \@nameuse{\filehook@atbegin@#1}%
}
\def\filehook@atend#1{%
  \@nameuse{\filehook@atend@#1}%
}
```

3.3 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 without an argument 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
}
```

User Level Macros

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

`\AtBeginOfIncludes`

```
\newcommand*\AtBeginOfIncludes{%
```

```
97    \filehook@appendwarg\filehook@include@atbegin
98  }

    \AtEndOfIncludes
99  \newcommand*\AtEndOfIncludes{%
100   \filehook@prefixwarg\filehook@include@atend
101 }

    \AfterOfIncludes
102 \newcommand*\AfterIncludes{%
103   \filehook@prefixwarg\filehook@include@after
104 }

\AtBeginOfIncludeFile
105 \newcommand*\AtBeginOfIncludeFile[1]{%
106   \@ifundefined{\filehook@include@atbegin@#1.tex}%
107     {\long\global\@namedef{\filehook@include@atbegin@#1.tex}}%
108     {\expandafter\filehook@append\csname\filehook@include@atbegin@#1.tex\endcsname}%
109 }

  \AtEndOfIncludeFile
110 \newcommand*\AtEndOfIncludeFile[1]{%
111   \@ifundefined{\filehook@include@atend@#1.tex}%
112     {\long\global\@namedef{\filehook@include@atend@#1.tex}}%
113     {\expandafter\filehook@prefix\csname\filehook@include@atend@#1.tex\endcsname}%
114 }

  \AfterOfIncludeFile
115 \newcommand*\AfterOfIncludeFile[1]{%
116   \@ifundefined{\filehook@include@after@#1.tex}%
117     {\long\global\@namedef{\filehook@include@after@#1.tex}}%
118     {\expandafter\filehook@prefix\csname\filehook@include@after@#1.tex\endcsname}%
119 }

    \AtBeginOfInputs
120 \newcommand*\AtBeginOfInputs{%
121   \filehook@appendwarg\filehook@input@atbegin
122 }

      \AtEndOfInputs
123 \newcommand*\AtEndOfInputs{%
124   \filehook@prefixwarg\filehook@input@atend
125 }

  \AtBeginOfInputFile
126 \newcommand*\AtBeginOfInputFile[1]{%
127   \@ifundefined{\filehook@input@atbegin@#1}%
128     {\long\global\@namedef{\filehook@input@atbegin@#1}}%
129     {\expandafter\filehook@append\csname\filehook@input@atbegin@#1\endcsname}%
130 }
```

\AtEndOfInputFile

```
\newcommand*\AtEndOfInputFile[1]{%
  \@ifundefined{\filehook@input@atend@#1}%
    {\long\global\@namedef{\filehook@input@atend@#1}}%
    {\expandafter\filehook@prefix\csname\filehook@input@atend@#1\endcsname}%
}
```

\AtBeginOfFiles

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

\AtEndOfFiles

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

\AtBeginOfFile

```
\newcommand*\AtBeginOfFile[1]{%
  \@ifundefined{\filehook@atbegin@#1}%
    {\long\global\@namedef{\filehook@atbegin@#1}}%
    {\expandafter\filehook@append\csname\filehook@atbegin@#1\endcsname}%
}
```

\AtEndOfFile

```
\newcommand*\AtEndOfFile[1]{%
  \@ifundefined{\filehook@atend@#1}%
    {\long\global\@namedef{\filehook@atend@#1}}%
    {\expandafter\filehook@prefix\csname\filehook@atend@#1\endcsname}%
}
```