The `ltfilehook` package*

Frank Mittelbach

July 21, 2020

Contents

1 Introduction **1**
1.1 Provided hooks . . . 1
1.2 General hooks for file reading . . . 2
1.3 Hooks for package and class files . . . 3
1.4 Hooks for `\include` files . . . 4
1.5 High-level interfaces for LaTeX . . . 4
1.6 A sample package for structuring the log output . . . 4

2 The Implementation **5**
2.1 expl3 helpers . . . 5
2.2 Declaring the file-related hooks . . . 7
2.3 Patching LaTeX commands (need proper integration later) . . . 7
2.4 High-level interfaces for LaTeX . . . 13

3 Package emulation for compatibility **14**
3.1 Package filehook emulation . . . 14
3.2 Package scrlfile emulation . . . 15

4 A sample package for structuring the log output **16**

Index **17**

1 Introduction

1.1 Provided hooks

The code offers a number of hooks into which packages (or the user) can add code to support different use cases. Many hooks are offered as pairs (i.e., the second hook is reversed. Also important to know is that these pairs are properly nested with respect to other pairs of hooks.

There are hooks that are executed for all files of a certain type (if they contain code), e.g., for all "include files" or all "packages", and there are also hooks that are specific to a single file, e.g., do something after the package `foo.sty` has been loaded.

*This package has version v0.9a dated 2020/07/19, © LaTeX Project.

1.2 General hooks for file reading

There are four hooks that are called for each file that is read using document-level commands such as `\input`, `\include`, `\usepackage`, etc. They are not called for files read using internal low-level methods, such as `\@input` or `\openin`.

`file/before`
`file/before/...`
`file/after/...`
`file/after`

These are:

`file/before`, `file/before/`⟨*file-name*⟩ These hooks are executed in that order just before the file is loaded for reading. The code of the first hook is used is used with every file, while the second is executed only for the file with matching ⟨*file-name*⟩ allowing you to specify code that only applies to one file.

`file/after/`⟨*file-name*⟩`, file/after` These hooks are after the file with name ⟨*file-name*⟩ has been fully consumed. The order is swapped (the specific one comes first) so that the `before` and `after` hooks nest properly, which is important if any of them involve grouping (e.g., contain environments, for example). furthermore both hooks are reversed hooks to support correct nesting of different packages add code to both `/before` and `/after` hooks.

So the overall sequence of hook processing for any file read through the user interface commands of LaTeX is:

```
\UseHook{⟨file/before⟩}
\UseHook{⟨file/before/⟨file name⟩⟩}
   ⟨file contents⟩
\UseHook{⟨file/after/⟨file name⟩⟩}
\UseHook{⟨file/after⟩}
```

Todo: With a higher-level interface that doesn't matter, but it is a bit weird, if you use `\AddToHook` *or* `\hook_gput:nnn` *directly, so I guess that has to be done differently!*

The file hooks only refer to the file by its name and extension, so the ⟨*file name*⟩ should be the file name as it is on the filesystem with extension (if any) and without paths. Different from `\input` and similar commands, for hooks the `.tex` extension is not assumed, so `.tex` files must also be given with their extension. Files within subfolders should also be addressed by their name and extension only.

Extensionless files also work, and should then be given without extension. Note however that TeX prioritizes `.tex` files, so if two files `foo` and `foo.tex` exist in the search path, only the latter will be seen.

When a file is input, the ⟨*file name*⟩ is available in `\CurrentFile`, which is then used when accessing the `file/before/`⟨*file name*⟩ and `file/after/`⟨*file name*⟩.

`\CurrentFile`

The name of the file about to be read (or just finished) is available to the hooks through `\CurrentFile` (there is no `expl3` name for it for now). The file is always provided with its extension, i.e., how it appears on your hard drive, but without any specified path to it. For example, `\input{sample}` and `\input{app/sample.tex}` would both have `\CurrentFile` being `sample.tex`.

\CurrentFilePath

The path to the current file (complement to \CurrentFile) is available in \CurrentFilePath if needed. The paths returned in \CurrentFilePath are only user paths, given through \input@path (or expl3's equivalent \l_file_search_path_seq) or by directly typing in the path in the \input command or equivalent. Files located by kpsewhich get the path added internally by the TEX implementation, so at the macro level it looks as if the file were in the current folder, so the path in \CurrentFilePath is empty in these cases (package and class files, mostly).

1.3 Hooks for package and class files

Commands to load package and class files (e.g., \usepackage, \RequirePackage, \LoadPackageWithOptions, etc.) offer the hooks from section 1.2 when they are used to load a package or class file, e.g., file/after/array.sty would be called after the array package got loaded. But as packages and classes form as special group of files, there are some additional hooks available that only apply when a package or class is loaded.

package/before
package/after
package/before/...
package/after/...
class/before
class/after
class/before/...
class/after/...

These are:

package/before, package/after These hooks are called for each package being loaded.

package/before/⟨*name*⟩, package/after/⟨*name*⟩ These hooks are additionally called if the package name is ⟨*name*⟩ (without extension).

class/before, class/after These hooks are called for each class being loaded.

class/before/⟨*name*⟩, class/after/⟨*name*⟩ These hooks are additionally called if the class name is ⟨*name*⟩ (without extension).

All /after hooks are implemented as reversed hooks.
The overall sequence of execution for \usepackage and friends is therefore:

```
\UseHook{⟨package/before⟩}
\UseHook{⟨package/before/⟨package name⟩⟩}
    \UseHook{⟨file/before⟩}
    \UseHook{⟨file/before/⟨package name⟩.sty⟩}
        ⟨package contents⟩
    \UseHook{⟨file/after/⟨package name⟩.sty⟩}
    \UseHook{⟨file/after⟩}
    code from \AtEndOfPackage if used inside the package
\UseHook{⟨package/after/⟨package name⟩⟩}
\UseHook{⟨package/after⟩}
```

and similar for class file loading, except that package/ is replaced by class/ and \AtEndOfPackage by \AtEndOfClass.

If a package or class is not loaded (or it was loaded before the hooks were set) none of the hooks are executed!

1.4 Hooks for \include files

To manage \include files, LaTeX issues a \clearpage before and after loading such a file. Depending on the use case one may want to execute code before or after these \clearpages especially for the one that is issued at the end.

Executing code before the final \clearpage, means that the code is processed while the last page of the included material is still under construction. Executing code after it means that all floats from inside the include file are placed (which might have added further pages) and the final page has finished.

Because of these different scenarios we offer hooks in three places.[1] None of the hooks are executed when an \include file is bypassed because of an \includeonly declaration. They are, however, all executed if LaTeX makes an attempt to load the \include file (even if it doesn't exist and all that happens is "No file ⟨*filename*⟩.tex").

include/before
include/before/...
include/end
include/end/...
include/after
include/after/...

These are:

include/before, include/before/⟨*name*⟩ These hooks are executed one after another after the initial \clearpage and after .aux file is changed to use ⟨*name*⟩.aux, but before the ⟨*name*⟩.tex file is loaded. In other words they are executed at the very beginning of the first page of the \include file.

include/end/⟨*name*⟩, include/end These hooks are executed (in that order) after LaTeX has stopped reading from the \include file, but before it has issued a \clearpage to output any deferred floats.

include/after/⟨*name*⟩, include/after These hooks are executed (in that order) after LaTeX has issued the \clearpage but before is has switched back writing to the main .aux file. Thus technically we are still inside the \include and if the hooks generate any further typeset material including anything that writes to the .aux file, then it would be considered part of the included material and bypassed if it is not loaded because of some \includeonly statement.[2]

1.5 High-level interfaces for LaTeX

We do not provide any high-level LaTeX commands (like filehook or scrlfile do) but think that for package writers the commands from for hook management are sufficient.

1.6 A sample package for structuring the log output

As an application we provide the package structuredlog that adds lines to the .log when a file is opened and closed for reading keeping track of nesting level es well. For example, for the current document it adds the lines

```
= (LEVEL 1 START) t1lmr.fd
= (LEVEL 1 STOP) t1lmr.fd
= (LEVEL 1 START) supp-pdf.mkii
```

[1] If you want to execute code before the first \clearpage there is no need to use a hook—you can write it directly in front of the \include.

[2] For that reason another \clearpage is executed after these hooks which normally does nothing, but starts a new page if further material got added this way.

```
= (LEVEL 1 STOP) supp-pdf.mkii
= (LEVEL 1 START) nameref.sty
== (LEVEL 2 START) refcount.sty
== (LEVEL 2 STOP) refcount.sty
== (LEVEL 2 START) gettitlestring.sty
== (LEVEL 2 STOP) gettitlestring.sty
= (LEVEL 1 STOP) nameref.sty
= (LEVEL 1 START) ltfilehook-doc.out
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.out
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.hd
= (LEVEL 1 STOP) ltfilehook-doc.hd
= (LEVEL 1 START) ltfilehook.dtx
== (LEVEL 2 START) ot1lmr.fd
== (LEVEL 2 STOP) ot1lmr.fd
== (LEVEL 2 START) omllmm.fd
== (LEVEL 2 STOP) omllmm.fd
== (LEVEL 2 START) omslmsy.fd
== (LEVEL 2 STOP) omslmsy.fd
== (LEVEL 2 START) omxlmex.fd
== (LEVEL 2 STOP) omxlmex.fd
== (LEVEL 2 START) umsa.fd
== (LEVEL 2 STOP) umsa.fd
== (LEVEL 2 START) umsb.fd
== (LEVEL 2 STOP) umsb.fd
== (LEVEL 2 START) ts1lmr.fd
== (LEVEL 2 STOP) ts1lmr.fd
== (LEVEL 2 START) t1lmss.fd
== (LEVEL 2 STOP) t1lmss.fd
= (LEVEL 1 STOP) ltfilehook.dtx
```

Thus if you inspect an issue in the `.log` it is easy to figure out in which file it occurred, simply by searching back for `LEVEL` and if it is a `STOP` then remove 1 from the level value and search further for `LEVEL` with that value which should then be the `START` level of the file you are in.

2 The Implementation

```
1 ⟨*2ekernel⟩
```

2.1 expl3 helpers

```
2 ⟨@@=filehook⟩
```

\CurrentFile
\CurrentFilePath

User-level macros that hold the current file name and file path. These are used internally as well because the code takes care to protect against a possible redefinition of these macros in the loaded file (it's necessary anyway to make hooks work with nested `\input`).

```
3 \def\CurrentFile{}
4 \def\CurrentFilePath{}
```

(*End definition for* `\CurrentFile` *and* `\CurrentFilePath`*. These functions are documented on page 2.*)

\l__filehook_internal_tl
\@filehook@set@curr@file
__filehook_normalise_file_name:n

When inputting a file, \@filehook@set@curr@file does a file lookup (in \input@path and \l_file_search_path_seq) and returns the actual file name (⟨*base*⟩ plus ⟨*ext*⟩) in \CurrentFile. Only the base and extension are returned, regardless of the input (both path/to/file.tex and file.tex end up as file.tex in \CurrentFile). The path is returned in \CurrentFilePath, in case it's needed. \CurrentFile is then used to run the file hooks with file/before/\CurrentFile and file/after/\CurrentFile.

```
\ExplSyntaxOn
\tl_new:N \l__filehook_internal_tl
\cs_new_protected:Npn \@filehook@set@curr@file #1
  { \exp_args:NV \__filehook_normalise_file_name:n #1 }
\cs_new_protected:Npn \__filehook_normalise_file_name:n #1
  {
    \file_if_exist:nTF {#1}
      {
        \exp_args:Nx \file_parse_full_name:nNNN
          { \file_full_name:n {#1} }
      }
      { \file_parse_full_name:nNNN {#1} }
      \CurrentFilePath \CurrentFile \l__filehook_internal_tl
    \tl_set:Nx \CurrentFile { \CurrentFile \l__filehook_internal_tl }
  }
```

(*End definition for* \l__filehook_internal_tl*,* \@filehook@set@curr@file*, and* __filehook_normalise_file_name:n*. This function is documented on page* ??.)

\g__filehook_input_file_seq
\@filehook@file@push
\@filehook@file@pop
__filehook_file_pop_assign:nn

Yet another stack, to keep track of \CurrentFile and \CurrentFilePath with nested \inputs. At the beginning of \InputIfFileExists, the current value of \CurrentFilePath and \CurrentFile is pushed to \g__filehook_input_file_seq, and at the end, it is popped and the value reassigned. \IfFileExists does \set@curr@file internally, which changes \CurrentFile, so \@filehook@file@push has to be executed before \IfFileExists.

```
\seq_new:N \g__filehook_input_file_seq
\cs_new_protected:Npn \@filehook@file@push
  {
    \seq_gpush:Nx \g__filehook_input_file_seq
      { { \CurrentFilePath } { \CurrentFile } }
  }
\cs_new_protected:Npn \@filehook@file@pop
  {
    \seq_gpop:NNTF \g__filehook_input_file_seq \l__filehook_internal_tl
      { \exp_after:wN \__filehook_file_pop_assign:nn \l__filehook_internal_tl }
      { \ERROR_should_not_happen }
  }
\cs_new_protected:Npn \__filehook_file_pop_assign:nn #1 #2
  {
    \tl_set:Nn \CurrentFilePath {#1}
    \tl_set:Nn \CurrentFile {#2}
  }
\ExplSyntaxOff
```

(*End definition for* \g__filehook_input_file_seq *and others. These functions are documented on page* ??.)

2.2 Declaring the file-related hooks

All hooks starting with `file/` `include/`, `class/` or `package/` are generic and will be allocated if code is added to them. Thus there is no need to explicitly declare any hook in the code below.

Furthermore, those named `.../after` or `.../end` are automatically declared as reversed hooks if filled with code, so this is also automatically taken care of.

2.3 Patching LaTeX commands (need proper integration later)

Most of what we have to do is adding `\UseHook` into several LaTeX 2_ε core commands, which is done for now by patching them.

```
38 ⟨@@=⟩
```

`\InputIfFileExists` `\InputIfFileExists` loads any file if it is available so we have to add the hooks `file/before` and `file/after` in the right places. If the file doesn't exist no hooks should be executed.

```
39 \let\InputIfFileExists\@undefined
40 \DeclareRobustCommand \InputIfFileExists[3]{%
41   \@filehook@file@push
42   \IfFileExists{#1}%
43     {%
```

If the file exists then `\@curr@file` holds its name. But we can't rely on that still being true after the file has been processed. Thus for using the name in the file hooks we need to preserve the name and then restored it for the `file/after/...` hook.

The hook always refers to the *actual* file that will be operated on, regardless of how the user had it written in the document. expl3's `\file_full_name:n` normalizes the file name (to factor out differences in the `.tex` extension), and then does a file lookup to take into account a possible path from `\l_file_search_path_seq` and `\input@path`. However only the file name and extension are returned so that file hooks can refer to the file by their name only. The path to the file is returned in `\CurrentFilePath`.

```
44       \edef\reserved@a{\@filef@und
45         \def\noexpand\CurrentFile{\CurrentFile}%
46         \def\noexpand\CurrentFilePath{\CurrentFilePath}%
47       }%
48       \expandafter\@swaptwoargs\expandafter
49         {\reserved@a}%
50         {%
51           #2%
52           \@addtofilelist{#1}%
53           \UseHook{file/before}%
```

The current file name is available in `\@curr@file` so we use that in the specific hook.

```
54           \UseHook{file/before/\CurrentFile}%
55           \@@input
56         }%
```

And it is restored here so we can use it once more.

```
57       \UseHook{file/after/\CurrentFile}%
58       \UseHook{file/after}%
59       \@filehook@file@pop
60     }%
61     {\@filehook@file@pop #3}%
```

```
}
```

(*End definition for* `\InputIfFileExists`*. This function is documented on page* **??**.)

`\set@curr@file` Now we just hook into `\set@curr@file` to add `\@filehook@set@curr@file` at the end, after `\@curr@file` is set.

```
\def\set@curr@file#1{%
  \begingroup
    \escapechar\m@ne
    \xdef\@curr@file{%
      \expandafter\expandafter\expandafter\unquote@name
      \expandafter\expandafter\expandafter{%
      \expandafter\string
        \csname\@firstofone#1\@empty\endcsname}}%
  \endgroup
  \@filehook@set@curr@file{\@curr@file}%
}
```

(*End definition for* `\set@curr@file`*. This function is documented on page* **??**.)

`\load@onefilewithoptions` This macro is used when loading packages or classes.

```
\def\load@onefilewithoptions#1[#2][#3]#4{%
  \@pushfilename
  \xdef\@currname{#1}%
  \global\let\@currext#4%
  \let\CurrentOption\@empty
  \@reset@ptions
  \makeatletter
  \def\reserved@a{%
    \@ifl@aded\@currext{#1}%
      {\@if@ptions\@currext{#1}{#2}{}%
        {\@latex@error
            {Option clash for \@cls@pkg\space #1}%
            {The package #1 has already been loaded
             with options:\MessageBreak
             \space\space[\@ptionlist{#1.\@currext}]\MessageBreak
             There has now been an attempt to load it
              with options\MessageBreak
             \space\space[#2]\MessageBreak
             Adding the global options:\MessageBreak
             \space\space
                  \@ptionlist{#1.\@currext},#2\MessageBreak
             to your \noexpand\documentclass declaration may fix this.%
             \MessageBreak
             Try typing \space <return> \space to proceed.}}}%
      {\@pass@ptions\@currext{#2}{#1}%
       \global\expandafter
       \let\csname ver@\@currname.\@currext\endcsname\@empty
       \expandafter\let\csname\@currname.\@currext-h@@k\endcsname\@empty
       \InputIfFileExists
         {\@currname.\@currext}%
         {%
%-----------------------------------------
```

When the current extension is `\@pkgextension` we are loading a package otherwise, if it is `\@clsextension`, a class, so depending on that we execute different hooks. If the extension is neither, then it is another type of file without special hooks.

```
            \ifx\@currext\@pkgextension
              \UseHook{package/before}%
              \UseHook{package/before/\@currname}%
            \else
              \ifx\@currext\@clsextension
                \UseHook{class/before}%
                \UseHook{class/before/\@currname}%
              \fi
            \fi
```

The value of `\CurrentFile` holds during `\InputIfFileExists`, so the hooks above have that available for using. However at this point `\CurrentFile` is reset to its previous value by `\@filehook@file@pop` because it doesn't know that we'll have more hooks ahead. So here (still in the ⟨*true*⟩ branch of `\InputIfFileExists`, right after actually reading the file in) we'll cheat: use `\@filehook@file@push` once more, so there are two entries for the current file in the name stack, so that when `\InputIfFileExists` pops it, there's still one (identical) left.

```
            \@filehook@file@push
%----------------------------------------
          }%
          {\@missingfileerror\@currname\@currext}%
    \let\@unprocessedoptions\@@unprocessedoptions
    \csname\@currname.\@currext-h@@k\endcsname
    \expandafter\let\csname\@currname.\@currext-h@@k\endcsname
                \@undefined
%----------------------------------------
```

And same procedure, James, when we are finished loading, except that the hook order is now reversed.

```
    \ifx\@currext\@pkgextension
      \UseHook{package/after/\@currname}%
      \UseHook{package/after}%
    \else
      \ifx\@currext\@clsextension
        \UseHook{class/after/\@currname}%
        \UseHook{class/after}%
      \fi
    \fi
```

Now here we do `\@filehook@file@pop` to restore the `\CurrentFile` before this file being loaded and fix what we've done in the stack right above.

```
    \@filehook@file@pop
%----------------------------------------
    \@unprocessedoptions}%
    \@ifl@ter\@currext{#1}{#3}{}%
      {\@latex@warning@no@line
         {You have requested,\on@line,
          version\MessageBreak
            `#3' of \@cls@pkg\space #1,\MessageBreak
          but only version\MessageBreak
           `\csname ver@#1.\@currext\endcsname'\MessageBreak
```

```
          is available}}%
    \ifx\@currext\@clsextension\let\LoadClass\@twoloadclasserror\fi
    \@popfilename
    \@reset@ptions}%
  \reserved@a}
```

The code for this macro has changed between 2020/02/02 and 2020/10/01 so the never version is this:

```
\@ifl@t@r\fmtversion{2020/10/01}
{%
\def\load@onefilewithoptions#1[#2][#3]#4{%
  \@pushfilename
  \xdef\@currname{#1}%
  \global\let\@currext#4%
  \let\CurrentOption\@empty
  \@reset@ptions
  \makeatletter
  \def\reserved@a{%
    \@ifl@aded\@currext{#1}%
      {\@if@ptions\@currext{#1}{#2}{}%
        {\@latex@error
            {Option clash for \@cls@pkg\space #1}%
            {The package #1 has already been loaded
             with options:\MessageBreak
             \space\space[\@ptionlist{#1.\@currext}]\MessageBreak
             There has now been an attempt to load it
              with options\MessageBreak
             \space\space[#2]\MessageBreak
             Adding the global options:\MessageBreak
             \space\space
                  \@ptionlist{#1.\@currext},#2\MessageBreak
             to your \noexpand\documentclass declaration may fix this.%
             \MessageBreak
             Try typing \space <return> \space to proceed.}}}%
      {\@pass@ptions\@currext{#2}{#1}%
       \global\expandafter
       \let\csname ver@\@currname.\@currext\endcsname\@empty
       \expandafter\let\csname\@currname.\@currext-h@@k\endcsname\@empty
       \InputIfFileExists
         {\@currname.\@currext}%
         {%
%----------------------------------------
           \ifx\@currext\@pkgextension
             \UseHook{package/before}%
             \UseHook{package/before/\@currname}%
           \else
             \ifx\@currext\@clsextension
               \UseHook{class/before}%
               \UseHook{class/before/\@currname}%
             \fi
           \fi
           \@filehook@file@push
%----------------------------------------
         }%
```

```
          {\@missingfileerror\@currname\@currext}%
    \expandafter\let\csname unprocessedoptions-\@currname.\@currext\endcsname
                    \@@unprocessedoptions
    \csname\@currname.\@currext-h@@k\endcsname
    \expandafter\let\csname\@currname.\@currext-h@@k\endcsname
              \@undefined
    \ifx\@unprocessedoptions\relax
      \let\@unprocessedoptions\@undefined
    \else
      \csname unprocessedoptions-\@currname.\@currext\endcsname
    \fi
    \expandafter\let
        \csname unprocessedoptions-\@currname.\@currext\endcsname
       \@undefined
%-----------------------------------------
    \ifx\@currext\@pkgextension
      \UseHook{package/after/\@currname}%
      \UseHook{package/after}%
    \else
      \ifx\@currext\@clsextension
        \UseHook{class/after/\@currname}%
        \UseHook{class/after}%
      \fi
    \fi
    \@filehook@file@pop
%-----------------------------------------
    }%
    \@ifl@ter\@currext{#1}{#3}{}%
      {\@latex@warning@no@line
         {You have requested,\on@line,
          version\MessageBreak
            `#3' of \@cls@pkg\space #1,\MessageBreak
          but only version\MessageBreak
           `\csname ver@#1.\@currext\endcsname'\MessageBreak
          is available}}%
    \ifx\@currext\@clsextension\let\LoadClass\@twoloadclasserror\fi
    \@popfilename
    \@reset@ptions}%
  \reserved@a}
}{}%
```

(*End definition for* `\load@onefilewithoptions`. *This function is documented on page* **??**.)

`\@include`

```
\def\@include#1 {%
  \clearpage
  \if@filesw
    \immediate\write\@mainaux{\string\@input{#1.aux}}%
  \fi
  \@tempswatrue
  \if@partsw
    \@tempswafalse
    \edef\reserved@b{#1}%
    \@for\reserved@a:=\@partlist\do
```

```
      {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
  \fi
  \if@tempswa
    \let\@auxout\@partaux
    \if@filesw
      \immediate\openout\@partaux #1.aux
      \immediate\write\@partaux{\relax}%
    \fi
%----------------------------------------
```

First we need to fix `\CurrentFile`, because due the pesky space-delimited `\@include`, `\CurrentFile` contains the `\include`'d file with a space. To fix that, we re-do `\set@curr@file`. The 2020/10/01 release doesn't need this as `\include` was changed to do `\set@curr@file` on the correct file name, rather than one with a trailing space.

```
    \set@curr@file{#1}%
```

Execute the `before` hooks just after we switched the `.aux` file ...

```
    \UseHook{include/before}%
    \UseHook{include/before/#1}%
%----------------------------------------
    \@input@{#1.tex}%
```

... then `end` hooks ...

```
%----------------------------------------
    \UseHook{include/end/#1}%
    \UseHook{include/end}%
%----------------------------------------
    \clearpage
```

... and after the `\clearpage` the `after` hooks followed by another `\clearpage` just in case new material got added (after all we need to be in well defined state after the `\include`).

```
%----------------------------------------
    \UseHook{include/after/#1}%
    \UseHook{include/after}%
```

The additional `\clearpage` is needed to ensure that switching the `.aux` files happen at a defined point even if the above hooks add further material.

```
    \clearpage
%----------------------------------------
    \@writeckpt{#1}%
    \if@filesw
      \immediate\closeout\@partaux
    \fi
  \else
    \deadcycles\z@
    \@nameuse{cp@#1}%
  \fi
  \let\@auxout\@mainaux
}
```

The code for this macro has changed between 2020/02/02 and 2020/10/01 so the never version is this:

```
\@ifl@t@r\fmtversion{2020/10/01}
{%
\def\@include#1 {%
```

```
  \clearpage
  \if@filesw
    \immediate\write\@mainaux{\string\@input{"#1.aux"}}%
  \fi
  \@tempswatrue
  \if@partsw
    \@tempswafalse
    \edef\reserved@b{#1}%
    \@for\reserved@a:=\@partlist\do
      {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
  \fi
  \if@tempswa
    \let\@auxout\@partaux
    \if@filesw
      \immediate\openout\@partaux "#1.aux"
      \immediate\write\@partaux{\relax}%
    \fi
%----------------------------------------
    \UseHook{include/before}%
    \UseHook{include/before/#1}%
%----------------------------------------
    \@input@{#1.tex}%
%----------------------------------------
    \UseHook{include/end/#1}%
    \UseHook{include/end}%
%----------------------------------------
    \clearpage
%----------------------------------------
    \UseHook{include/after/#1}%
    \UseHook{include/after}%
    \clearpage
%----------------------------------------
    \@writeckpt{#1}%
    \if@filesw
      \immediate\closeout\@partaux
    \fi
  \else
    \deadcycles\z@
    \@nameuse{cp@#1}%
  \fi
  \let\@auxout\@mainaux}
}{}
```

(*End definition for* `\@include`. *This function is documented on page* ??.)

2.4 High-level interfaces for LaTeX

None so far and the general feeling for now is that the hooks are enough. Packages like filehook, etc., may use them to set up their interfaces (samples are given below) but for the now the kernel will not provide any.

```
⟨/2ekernel⟩
```

3 Package emulation for compatibility

3.1 Package filehook emulation

This is a partial implementation of the filehook interfaces. It is only meant for guidance in case that package gets updated to use the hook management.

Not implemented are:

```
\AtBeginOfFiles
\AtEndOfFiles
\AtBeginOfInputs
\AtEndOfInputs
\AtBeginOfInputFile
\AtEndOfInputFile
```

```
⟨*filehook-draft⟩
\newcommand\AtBeginOfEveryFile [1]
  {\AddToHook{file/before}{#1}}
\newcommand\AtEndOfEveryFile [1]
  {\AddToHook{file/after}{#1}}

\newcommand\AtBeginOfIncludes [1]
  {\AddToHook{include/before}{#1}}
\newcommand\AtEndOfIncludes [1]
  {\AddToHook{include/end}{#1}}
\newcommand\AfterIncludes [1]
  {\AddToHook{include/after}{#1}}

\newcommand\AtBeginOfPackages [1]
  {\AddToHook{package/before}{#1}}
\newcommand\AtEndOfPackages [1]
  {\AddToHook{package/after}{#1}}

\newcommand\AtBeginOfClasses [1]
  {\AddToHook{class/before}{#1}}
\newcommand\AtEndOfClasses [1]
  {\AddToHook{class/after}{#1}}
```

For normal files we drop the `.tex` extension for now:

```
\newcommand\AtBeginOfFile [2]
  {\AddToHook{file/before/#1}{#2}}
\newcommand\AtEndOfFile [2]
  {\AddToHook{file/after/#1}{#2}}

\DeclareDocumentCommand \AtBeginOfPackageFile {smm}
   {\IfBooleanTF{#1}%
     {\@ifpackageloaded{#2}%
         {#3}%
         {\AddToHook{package/before/#2}{#3}}}%
     {\AddToHook{package/before/#2}{#3}}%
   }
\DeclareDocumentCommand \AtEndOfPackageFile {smm}
   {\IfBooleanTF{#1}%
     {\@ifpackageloaded{#2}%
         {#3}%
         {\AddToHook{package/after/#2}{#3}}}%
```

```
      {\AddToHook{package/after/#2}{#3}}%
   }
```

Are the * forms here of any use? I know they are use 3–4 times on CTAN but I wonder if those are real or mistaken usages.

```
\DeclareDocumentCommand \AtBeginOfClassFile {smm}
   {\IfBooleanTF{#1}%
     {\@ifclassloaded{#2}%
         {#3}%
         {\AddToHook{class/before/#2}{#3}}}%
     {\AddToHook{class/before/#2}{#3}}%
   }
\DeclareDocumentCommand \AtEndOfClassFile {smm}
   {\IfBooleanTF{#1}%
     {\@ifclassloaded{#2}%
         {#3}%
         {\AddToHook{class/after/#2}{#3}}}%
     {\AddToHook{class/after/#2}{#3}}%
   }
\newcommand\AtBeginOfIncludeFile [2]
  {\AddToHook{include/before/#1}{#2}}
\newcommand\AtEndOfIncludeFile [2]
  {\AddToHook{include/end/#1}{#2}}
\newcommand\AfterIncludeFile [2]
  {\AddToHook{include/after/#1}{#2}}
```

This is missing some interfaces so disabling the package isn't really correct, but then this code above is not supposed to stay like this anyway.

```
\expandafter\let\csname ver@filehook.sty\endcsname\fmtversion
\@namedef {ver@filehook.sty}{2020/10/01}
```

⟨/filehook-draft⟩

3.2 Package scrlfile emulation

This is a partial implementation of the scrlfile interfaces. It is only meant for guidance in case that package gets updated to use the hook management.

⟨*scrlfile-draft⟩

I think this is roughly correct (using the `file/...` hooks rather than the class or package hooks at least for the `\After...` commands but it needs some further verification.

The star and plus variants haven't been implemented so far, this is only a rough draft.

```
\newcommand\BeforeClass[2]
  {\AddToHook{file/before/#1.cls}{#2}}
\newcommand\AfterClass [2]
  {\AddToHook{file/after/#1.cls}{#2}}
\newcommand\AfterAtEndOfClass [2]
  {\AddToHook{class/after/#1}{#2}}
```

```
\newcommand\BeforePackage [2]
  {\AddToHook{package/before/#1.sty}{#2}}
\newcommand\AfterPackage [2]
  {\AddToHook{file/after/#1.sty}{#2}}
\newcommand\AfterEndOfPackage [2]
  {\AddToHook{package/after/#1}{#2}}
\newcommand\BeforeFile [2]
  {%
  \typeout{BeforeFile: #1!!!}%
   \AddToHook{file/before/#1}{#2}}
\newcommand\AfterFile [2]
  {%
    \typeout{AfterFile: #1!!!}%
    \AddToHook{file/after/#1}{#2}}
```

This is missing some interfaces so disabling the package isn't really correct, but then this code above is not supposed to stay like this anyway.

```
\expandafter\let\csname ver@scrlfile.sty\endcsname\fmtversion
\@namedef {ver@scrlfile.sty}{2020/10/01}
```

⟨/scrlfile-draft⟩

4 A sample package for structuring the log output

⟨*structuredlog⟩
⟨@@=filehook⟩

```
\ProvidesExplPackage
    {structuredlog}{\ltfilehookdate}{\ltfilehookversion}
    {Structuring the TeX transcript file}
\int_new:N \g__filehook_nesting_level_int
\tl_new:N   \g__filehook_nesting_prefix_tl
\tl_gset:Nn \g__filehook_nesting_prefix_tl { }
\AddToHook{file/before}{
  \int_gincr:N \g__filehook_nesting_level_int
  \tl_gput_right:Nn\g__filehook_nesting_prefix_tl {=}
  \iow_term:x {
    \g__filehook_nesting_prefix_tl \space
    ( LEVEL~ \int_use:N \g__filehook_nesting_level_int \space START )~
    \CurrentFile  ^^J
  }
}
```

We don't want to install the `file/after` hook immediately, because that would mean it is the first time executed when the package finishes. We therefore put the declaration inside `\AddToHookNext` so that it gets only installed when we have left the package.

```
\AddToHookNext{file/after}{
  \AddToHook{file/after}{
    \iow_term:x {
      \g__filehook_nesting_prefix_tl \space
      ( LEVEL~ \int_use:N \g__filehook_nesting_level_int \space STOP )~
      \CurrentFile  ^^J
    }
```

```
    \int_gdecr:N \g__filehook_nesting_level_int
    \tl_gset:Nx \g__filehook_nesting_prefix_tl
       {\exp_after:wN \use_none:n \g__filehook_nesting_prefix_tl}
  }
}
```

We have to manually increment the level because now that we have installed the code in `file/after` it gets decremented when we leave the package without ever being incremented upon entry.

```
%\int_incr:N\g__filehook_nesting_level_int
```

⟨/structuredlog⟩

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

A

\AddToHook *2*, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 351, 352, 358, 359, 365, 366, 372, 373, 376, 378, 380, 386, 388, 390, 392, 394, 396, 400, 404, 416, 426
\AddToHookNext *16*, 425
\After... . . . . *15*
\AfterAtEndOfClass 389
\AfterClass 387
\AfterEndOfPackage 395
\AfterFile 401
\AfterIncludeFile 379
\AfterIncludes 333
\AfterPackage 393
\AtBeginOfClasses 339
\AtBeginOfClassFile 361
\AtBeginOfEveryFile 325
\AtBeginOfFile 343
\AtBeginOfIncludeFile 375
\AtBeginOfIncludes 329
\AtBeginOfPackageFile 347
\AtBeginOfPackages 335
\AtEndOfClass *3*
\AtEndOfClasses 341
\AtEndOfClassFile 368
\AtEndOfEveryFile 327
\AtEndOfFile 345
\AtEndOfIncludeFile 377
\AtEndOfIncludes 331
\AtEndOfPackage *3*
\AtEndOfPackageFile 354
\AtEndOfPackages 337

B

\BeforeClass 385
\BeforeFile 397
\BeforePackage 391
\begingroup 64

C

class/after *3*
class/after/... . . . . *3*
class/before *3*
class/before/... . . . . *3*
\clearpage *3*, *4*, *12*, *12*, 235, 262, 266, 281, 307, 311
\closeout 270, 315
cs commands:
  \cs_new_protected:Npn 7, 9, 21, 26, 32
\csname 70, 100, 101, 120, 121, 142, 176, 177, 195, 197, 198, 203, 206, 227, 381, 405
\CurrentFile *2*, *2*, *2*, 3, *5*, *6*, *8*, *9*, *11*, 17, 18, 24, 35, 45, 54, 57, 422, 430
\CurrentFilePath *2*, 3, *5*, *6*, *7*, 17, 24, 34, 46
\CurrentOption 78, 154

D

\deadcycles 273, 318
\DeclareDocumentCommand 347, 354, 361, 368
\DeclareRobustCommand 40
\def 3, 4, 45, 46, 63, 74, 81, 150, 157, 234, 280
\do 243, 289
\documentclass 95, 171

E

\edef 44, 242, 288

\else 109, 127, 185, 202, 212, 272, 317
\endcsname 70, 100, 101, 120, 121, 142, 176, 177, 195, 197, 198, 203, 206, 227, 381, 405
\endgroup 71
ERROR commands:
\ERROR_should_not_happen 30
\escapechar 65
exp commands:
\exp_after:wN 29, 434
\exp_args:NV 8
\exp_args:Nx 13
\expandafter ... 48, 67, 68, 69, 99, 101, 121, 175, 177, 195, 198, 205, 381, 405
\ExplSyntaxOff 37
\ExplSyntaxOn 5

F

\fi 113, 114, 131, 132, 144, 189, 190, 204, 216, 217, 229, 238, 244, 245, 251, 271, 275, 284, 290, 291, 297, 316, 320
file commands:
\file_full_name:n 7, 14
\file_if_exist:nTF 11
\file_parse_full_name:nNNN .. 13, 16
\l_file_search_path_seq 2, 5, 7
file/after 1
file/after/... 1
file/before 1
file/before/... 1
filehook internal commands:
__filehook_file_pop_assign:nn .. 20
\g__filehook_input_file_seq .. 6, 20
\l__filehook_internal_tl ... 5, 28, 29
\g__filehook_nesting_level_int 413, 417, 421, 429, 432, 437
\g__filehook_nesting_prefix_tl 414, 415, 418, 420, 428, 433, 434
__filehook_normalise_file_-name:n 5
\fmtversion 148, 278, 381, 405

G

\global 77, 99, 153, 175

H

hook commands:
\hook_gput:nnn 2

I

\IfBooleanTF 348, 355, 362, 369
\IfFileExists 6, 42
\ifx 106, 110, 124, 128, 144, 182, 186, 200, 209, 213, 229, 244, 290
\immediate 237, 249, 250, 270, 283, 295, 296, 315
\include 1, 3, 4, 11, 12
include/after 4
include/after/... 4
include/before 4
include/before/... 4
include/end 4
include/end/... 4
\includeonly 3, 4
\input 1, 2, 2, 5, 6
\InputIfFileExists 6, 7, 8, 9, 39, 102, 178
int commands:
\int_gdecr:N 432
\int_gincr:N 417
\int_incr:N 437
\int_new:N 413
\int_use:N 421, 429
iow commands:
\iow_term:n 419, 427

L

\let . 39, 77, 78, 100, 101, 119, 121, 144, 153, 154, 176, 177, 195, 198, 201, 205, 229, 247, 276, 293, 321, 381, 405
\LoadClass 144, 229
\LoadPackageWithOptions 2
\ltfilehookdate 411
\ltfilehookversion 411

M

\makeatletter 80, 156
\MessageBreak 87, 88, 90, 91, 92, 94, 96, 139, 140, 141, 142, 163, 164, 166, 167, 168, 170, 172, 224, 225, 226, 227

N

\newcommand 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 375, 377, 379, 385, 387, 389, 391, 393, 395, 397, 401
\noexpand 45, 46, 95, 171

O

\openin 1
\openout 249, 295

P

package/after 3
package/after/... 3
package/before 3
package/before/... 3
\ProvidesExplPackage 410

R

\relax 200, 250, 296
\RequirePackage *2*
reserved@a commands:
\reserved@a: 243, 289

S

seq commands:
\seq_gpop:NNTF 28
\seq_gpush:Nn 23
\seq_new:N 20
\space . 85, 88, 91, 93, 97, 140, 161, 164, 167, 169, 173, 225, 420, 421, 428, 429
\string 69, 237, 283

T

TeX and LaTeX 2ε commands:
\@@input 55
\@@unprocessedoptions 119, 196
\@addtofilelist 52
\@auxout 247, 276, 293, 321
\@cls@pkg 85, 140, 161, 225
\@clsextension *8*, 110, 128, 144, 186, 213, 229
\@curr@file *7*, *7*, *7*, 66, 72
\@currext 77, 82, 83, 88, 94, 98, 100, 101, 103, 106, 110, 118, 120, 121, 124, 128, 136, 142, 144, 153, 158, 159, 164, 170, 174, 176, 177, 179, 182, 186, 194, 195, 197, 198, 203, 206, 209, 213, 221, 227, 229
\@currname 76, 100, 101, 103, 108, 112, 118, 120, 121, 125, 129, 152, 176, 177, 179, 184, 188, 194, 195, 197, 198, 203, 206, 210, 214
\@empty . 70, 78, 100, 101, 154, 176, 177
\@filef@und 44
\@filehook@file@pop *9*, *9*, 20, 59, 61, 133, 218
\@filehook@file@push *6*, *9*, 20, 41, 115, 191
\@filehook@set@curr@file . 5, *5*, *7*, 72
\@firstofone 70
\@for 243, 289
\@if@ptions 83, 159
\@ifclassloaded 363, 370
\@ifl@aded 82, 158
\@ifl@t@r 148, 278
\@ifl@ter 136, 221
\@ifpackageloaded 349, 356
\@include *11*, 234
\@input *1*, 237, 283
\@input@ 257, 302
\@latex@error 84, 160
\@latex@warning@no@line 137, 222
\@mainaux 237, 276, 283, 321
\@missingfileerror 118, 194
\@namedef 382, 406
\@nameuse 274, 319
\@partaux 247, 249, 250, 270, 293, 295, 296, 315
\@partlist 243, 289
\@pass@ptions 98, 174
\@pkgextension . . *8*, 106, 124, 182, 209
\@popfilename 145, 230
\@ptionlist 88, 94, 164, 170
\@pushfilename 75, 151
\@reset@ptions 79, 146, 155, 231
\@swaptwoargs 48
\@tempswafalse 241, 287
\@tempswatrue 239, 244, 285, 290
\@twoloadclasserror 144, 229
\@undefined 39, 122, 199, 201, 207
\@unprocessedoptions 119, 135, 200, 201
\@writeckpt 268, 313
\if@filesw . 236, 248, 269, 282, 294, 314
\if@partsw 240, 286
\if@tempswa 246, 292
\input@path *2*, *5*, *7*
\load@onefilewithoptions 74
\m@ne 65
\on@line 138, 223
\reserved@a 44, 49, 81, 147, 157, 232, 244, 290
\reserved@b 242, 244, 288, 290
\set@curr@file *6*, *7*, *11*, 63, 253
\unquote@name 67
\z@ 273, 318
tl commands:
\tl_gput_right:Nn 418
\tl_gset:Nn 415, 433
\tl_new:N 6, 414
\tl_set:Nn 18, 34, 35
\typeout 399, 403

U

use commands:
\use_none:n 434
\UseHook *2*, *3*, *6*, 53, 54, 57, 58, 107, 108, 111, 112, 125, 126, 129, 130, 183, 184, 187, 188, 210, 211, 214, 215, 254, 255, 259, 260, 264, 265, 299, 300, 304, 305, 309, 310
\usepackage *1*, *2*, *3*

W

\write 237, 250, 283, 296

X

\xdef 66, 76, 152