Warning! This package deliberately breaks certain functionality. Use only for debugging.

Package unhook v. 1.0 Implementation

Conrad Kosowsky

September 2026

kosowsky.latex@gmail.com

Overview

The unhook package disables various LaTeX3 features to improve tracing. It is intended for LaTeX 2_ε package developers to make debugging easier, and you should not load it for any other purpose because it breaks a lot of things. The package file is unhook.tex, so you should input it directly instead of calling \usepackage.

This file documents the code for the unhook package. It is not a user guide! For discussion of how unhook works and how to load it, see unhook-user-guide.pdf, which is included with the unhook installation and is available on CTAN. I believe the LaTeX team incorporated LaTeX3 into the kernel in August 2020, so developers probably won't gain anything when using unhook with an earlier LaTeX format. (But you should not be using a TeX distribution that is that old anyway.) We're skipping package declaration because we're not allowing users to load unhook with \usepackage, so it is a good idea to start with a check of the format. The first 59 lines of unhook.tex are comments. Before anything else, we save and set the catcode of \% to 14 in case we're loading the file in a situation where the catcode is something else, e.g. while using docstrip.

```
\edef\temp{\number\catcode`\%}
\catcode`\%=14\relax
\ifx\LaTeX\undefined
  \begingroup
  \newlinechar=`\^^J
  \errhelp{The file unhook.tex is for use with the LaTeX^^J%
           format. I'm guessing you're using plain TeX,^^J%
           in which case there's no point loading it, so^^J%
           I'm going to stop reading it in. To resolve^^J%
           this error, typeset your document with LaTeX.^^J}
```

We need an error message if unhook can't detect LaTeX. This next code is based on lterror.dtx and incorporates what the documentation describes as a "horrible hack," which I think is actually ingenious, to make an error message that looks like LaTeX errors, i.e. it ends in a line containing `␣...` followed by a blank line. When TeX encounters \errmessage, it prints the argument of the control sequence on the terminal. If the \errmessage command appears inside a macro, TeX then prints the definition of the macro up to \errmessage on one line followed by any remaining portion of the macro definition on the next line. TeX may shorten this text by replacing some of it with an ellipsis. It follows that

1. If the macro name is stored internally as a space, the line with the macro definition will begin with a space followed by ...
2. If the final characters of \errmessage and the closing brace are stored internally as spaces, then the line with the macro definition will contain nothing after the ...
3. If \errmessage appears last in the macro, TeX will include a blank line after the error

Warning! This package deliberately breaks certain functionality. Use only for debugging.

We can achieve these three conditions by (a) defining the error using `~`; (b) using `\lccode` to turn `~` and `}` into a space; and (c) ending the `\errmessage` with a macro whose name ends in a large number of space characters and that expands to nothing. Inside `\errmessage`, the actual error message doesn't show the control sequence because it expands when TEX prints the message, and TEX doesn't show anything after the `...` because all final characters of the definition of `~` are stored in memory as spaces.

```
  \catcode`\~=13\relax
  \lccode`\~=`\ %
  \lccode`\}=`\ %
```

We set other `\lccode`'s so that `\lowercase` doesn't mess up the text of our error message.

```
  \lccode`\P=`\P% Package
  \lccode`\N=`\N% Needs
  \lccode`\L=`\L% LaTeX
  \lccode`\T=`\T%
  \lccode`\X=`\X%
  \lccode`\S=`\S% See
% \lccode`\T=`\T% Type (commented because we already did T)
  \lccode`\H=`\H% H
  \catcode`\ =11%
\def% |<---- These spaces are part of the control sequence name ---->|
\empty                                                                %
{}%
\lowercase{%
\def~{\errmessage{^^J^^JPackage unhook error: Needs LaTeX format^^J^^J%
See the unhook package documentation for explanation.^^J%
Type H<return> for immediate help%
\empty                                                                %
}}~%
```

Brace that matches `\lowercase{` appears on its own line so that the most recent line that TEX has scanned contains minimal text. This means TEX will print minimal extra text after the error message.

```
}
\endgroup
  \expandafter\endinput % balance the conditional before \endinput
\fi
```

Now check that `@` has catcode 11. If no, stop loading the package and print a warning message. We could change the catcode ourselves, but if we don't, it serves as a check to prevent casual users from loading the file. If someone needs to use this package, they will understand what to do without further explanation. Yes, I am deliberately making it inaccessible to load the package—if you are persistent enough to get this far and don't understand what's happening, feel free to email me, and I will explain in detail.

```
\ifnum\catcode`\@=11\relax
\else
```

Scan past the end of `\PackageError` so that minimal extra text (just a `}`) gets printed with the error message.

```
    \csname @firstofone\endcsname{%
      \PackageError{unhook}{Wrong catcode for @}
      {The catcode of @ needs to be 11 when you load unhook.tex.^^J%
      I'm going to stop reading in the package file now.^^J}}
}
  \expandafter\endinput % balance the conditional before \endinput
\fi
```

Reset the primitives `\@@par`, `\everypar`, and `\shipout`. We reset `\@@par` because this is traditionally how the LaTeX kernel stores `\par`. We are assuming that if the LaTeX3 names of these primitives don't exist, then we don't need to restore the original control sequences.

```
\ifcsname tex_par:D\endcsname
  \expandafter\let\expandafter\@@par\csname tex_par:D\endcsname
\fi
\ifcsname tex_everypar:D\endcsname
  \expandafter\let\expandafter\everypar\csname tex_everypar:D\endcsname
  \everypar{}
\fi
\ifcsname tex_shipout:D\endcsname
  \expandafter\let\expandafter\shipout\csname tex_shipout:D\endcsname
\fi
```

More `\par` resetting. The kernel expects `\par` to be the same as `\@@par` by default, and `\endgraf` is a synonym for `\par`.

```
\let\par\@@par
\let\endgraf\@@par
```

Supporting macro that checks if the next token is `[` and gobbles an optional argument if yes.

```
\def\unhook@n@@pt[#1]{}
\def\unhook@noopt{\@ifnextchar[\unhook@n@@pt\relax}
```

We disable some of the code for using hooks.

```
\protected\def\UseHook#1{}
```

The `WithArguments` hooks are harder because they have a variable number of arguments. We need to gobble the first argument, then process the second argument, which should be a number, and then remove that many arguments from the input stream.

```
\def\unhook@gobblex#1{%
  \count@=\numexpr#1\relax
```

If the number of arguments is non-positive, we assume that means zero arguments, in which case we have nothing to do. Otherwise, we have to remove arguments from the input stream. If the user requested more than 9 arguments, raise an error and cap the request at 9.

```
  \ifnum\count@>0\relax
    \ifnum\count@>9\relax
      \@latex@error{Hook limited to 9 arguments}\@ehc
      \count@=9\relax
    \fi
```

The next few lines get a bit wild. We are using parameter arguments to change the parameter specification of `\@tempa` from `#1#2#3#4#5#6#7#8#9` to only `\count@` arguments. Here is the

value of `\@tempa` immediately after each of the next three lines, where ⟨*number*⟩ is the value of `#1`:

1. `->####1`⟨*number*⟩`####2`
2. `#1`⟨*number*⟩`#2\@nil->#1`⟨*number*⟩
3. `#1#2…#`⟨*number*⟩`->`

In the third line below, the application of `\@tempa` yields

`#1<-\def\@tempa##1 … ##`⟨*number* − 1⟩`##`
`#2<-##`⟨*number* + 1⟩` … ##9\@nil`

so we end up with `\def\@tempa` followed by the correct number of parameters.

```
    \edef\@tempa{####1\the\count@####2}%
    \expandafter\edef\expandafter\@tempa\@tempa\@nil{##1\the\count@}%
    \@tempa\def\@tempa##1##2##3##4##5##6##7##8##9\@nil{}%
```

Now we are ready to gobble from the input stream.

```
    \expandafter\@tempa
 \fi}
```

We can use this macro to disable the commands for using hook with arguments.

```
\protected\def\UseHookWithArguments#1{\unhook@gobblex}
\protected\def\UseOneTimeHookWithArguments#1{\unhook@gobblex}
```

Also disable code for modifying hooks.

```
\protected\def\AddToHook#1{\expandafter\@gobble\unhook@noopt}
\protected\def\AddToHookWithArguments#1{\expandafter\@gobble\unhook@noopt}
\protected\def\RemoveFromHook#1{\unhook@noopt}
\protected\def\AddToHookNext#1#2{}
\protected\def\AddToHookNextWithArguments#1#2{}
\protected\def\ClearHookNext#1{}
\protected\def\PushDefaultHookLabel#1\PopDefaultHookLabel{}
\protected\def\SetDefaultHookLabel#1{}
```

When I've seen `\IfHookEmptyTF`, it has come after `\romannumeral` to force expansion. If `\romannumeral` appears right before `\IfHookEmptyTF`, we want it to expand to 0 (or something negative) to satisfy `\romannumeral` and make it disappear. I believe this is the purpose of `\exp_end:`, which is defined as `\char"0` and appears in the original definitions of these macros.

```
\def\IfHookEmptyTF#1#2#3{0\relax}
\def\IfHookEmptyT#1#2{0\relax}
\def\IfHookEmptyF#1#2{0\relax}
```

Disable some of the code for using sockets.

```
\protected\def\UseSocket#1{}
\protected\def\UseTaggingSocket#1{}
```

I'm assuming it is the same situation for the checks of sockets.

```
\def\IfSocketExistsTF#1#2#3{0\relax}
\def\IfSocketPlugExistsTF#1#2#3#4{0\relax}
\def\IfSocketPlugAssignedTF#1#2#3#4{0\relax}
```

Warning! This package deliberately breaks certain functionality. Use only for debugging.

As far as I can tell, all the material for the beginning of the document gets stored in `\__hook_toplevel␣begindocument`. We redefine `\UseOneTimeHook` to call this macro in place of processing the `begindocument` hook name and similarly for `enddocument`. For any other hook name, the macro gobbles its argument. We implement this approach by making `\@tempa` be `\relax` unless we're at the beginning or end of the document, in which case it holds the material from the hook.

```
\protected\def\UseOneTimeHook#1{%
  \let\@tempa\relax
  \def\@tempb{#1}%
  \def\@tempc{begindocument}%
  \ifx\@tempb\@tempc
    \expandafter\let\expandafter\@tempa
      \csname __hook_toplevel begindocument\endcsname
  \else
    \def\@tempc{enddocument}%
    \ifx\@tempb\@tempc
      \expandafter\let\expandafter\@tempa
        \csname __hook_toplevel enddocument\endcsname
    \fi
  \fi
  \@tempa}
```

We redefine `\AtBeginDocument` to append code to `\__hook_toplevel␣begindocument`. This macro is global in the kernel, so we use `\xdef` when we redefine it. If the macro does not exist, we do not redefine it because that could mess up another implementation of `\AtBeginDocument`.

```
\ifcsname __hook_toplevel begindocument\endcsname
  \protected\def\AtBeginDocument#1{%
    \bgroup
      \toks@\expandafter\expandafter\expandafter{%
        \csname __hook_toplevel begindocument\endcsname}%
      \@temptokena{#1}%
      \expandafter\xdef
        \csname __hook_toplevel begindocument\endcsname{%
          \the\toks@\the\@temptokena}%
    \egroup}
\fi
```

And do `\AtEndDocument`.

```
\ifcsname __hook_toplevel enddocument\endcsname
  \protected\def\AtEndDocument#1{%
    \bgroup
      \toks@\expandafter\expandafter\expandafter{%
        \csname __hook_toplevel enddocument\endcsname}%
      \@temptokena{#1}%
      \expandafter\xdef
        \csname __hook_toplevel enddocument\endcsname{%
          \the\toks@\the\@temptokena}%
```

```
    \egroup}
\fi
```

If the next few commands are undefined, letting them `\relax` or `\@gobble` their argument shouldn't be a problem.

```
\let\@expl@sys@load@backend@@                \relax
\let\@kernel@before@begindocument            \relax
\let\@kernel@after@begindocument             \relax
\let\@kernel@after@begindocument@before      \relax
\let\@kernel@before@enddocument              \relax
\let\@kernel@after@enddocument               \relax
\let\@kernel@before@enddocument@afterlastpage\relax
\let\@kernel@after@enddocument@afterlastpage \relax
\let\@kernel@before@para@before              \relax
\let\@kernel@before@para@begin               \relax
\let\@kernel@after@para@after                \relax
\let\@kernel@after@para@end                  \relax
\let\@execute@begin@hook                     \@gobble
```

We also need to restore a small piece of the output routine.

```
\def\@opcol{%
  \if@twocolumn
    \@outputdblcol
  \else
    \@outputpage
  %\global\@colht\textheight
  \fi
  \global\@mparbottom\z@
  \global\@textfloatsheight\z@
  \@floatplacement}
```

The rest of the code here is from an older version of `ltfiles.dtx`. We start with `\@iinput`, which is from the LaTeX version of `\input` when followed by a square bracket.

```
\def\@iinput#1{%
  \InputIfFileExists{#1}{}%
  {\filename@parse{#1}%
   \edef\reserved@a{\noexpand\@missingfileerror
     {\filename@area\filename@base}%
     {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
   \reserved@a}}
```

The next two macros are the "safe" file-loading commands.

```
\long\def\InputIfFileExists#1#2{%
  \IfFileExists{#1}%
  {#2\@addtofilelist{#1}\@@input\@filef@und}}
\long\def\IfFileExists#1#2#3{%
  \openin\@inputcheck#1 %
  \ifeof\@inputcheck
    \ifx\input@path\@undefined
      \def\reserved@a{#3}%
```

```
    \else
      \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
    \fi
  \else
    \closein\@inputcheck
    \edef\@filef@und{#1 }%
    \def\reserved@a{#2}%
  \fi
  \reserved@a}
```

Macro for handling file paths.

```
\long\def\@iffileonpath#1{%
  \let\reserved@a\@secondoftwo
  \expandafter\@tfor\expandafter\reserved@b\expandafter
      :\expandafter=\input@path\do{%
    \openin\@inputcheck\reserved@b#1 %
    \ifeof\@inputcheck\else
      \edef\@filef@und{\reserved@b#1 }%
      \let\reserved@a\@firstoftwo%
      \closein\@inputcheck
      \@break@tfor
    \fi}%
  \reserved@a}
```

The next three macros are for loading package and class files and handling their options.

```
\def\@fileswith@pti@ns#1[#2]#3[#4]{%
  \ifx#1\@clsextension
    \ifx\@classoptionslist\relax
      \xdef\@classoptionslist{\zap@space#2 \@empty}%
      \def\reserved@a{%
        \@onefilewithoptions#3[{#2}][{#4}]#1%
        \@documentclasshook}%
    \else
      \def\reserved@a{%
        \@onefilewithoptions#3[{#2}][{#4}]#1}%
    \fi
  \else
    \def\reserved@b##1,{%
      \ifx\@nnil##1\relax
      \else
        \ifx\@nnil##1\@nnil
        \else
          \noexpand\@onefilewithoptions##1[{#2}][{#4}]%
          \noexpand\@pkgextension
        \fi
        \expandafter\reserved@b
      \fi}%
    \edef\reserved@a{\zap@space#3 \@empty}%
    \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
```

```
  \fi
  \reserved@a}
\let\@@fileswith@pti@ns\@fileswith@pti@ns
\def\@onefilewithoptions#1[#2][#3]#4{%
  \@pushfilename
  \xdef\@currname{#1}%
  \global\let\@currext#4%
  \expandafter\let\csname\@currname.\@currext-h@@k\endcsname\@empty
  \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
      \InputIfFileExists
        {\@currname.\@currext}%
        {}%
        {\@missingfileerror\@currname\@currext}%
      \let\@unprocessedoptions\@@unprocessedoptions
      \csname\@currname.\@currext-h@@k\endcsname
      \expandafter\let\csname\@currname.\@currext-h@@k\endcsname\@undefined
      \@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}
```

Macro for passing options to a class or package file.

```
\def\@pass@ptions#1#2#3{%
  \expandafter\xdef\csname opt@#3.#1\endcsname{%
  \@ifundefined{opt@#3.#1}\@empty
  {\csname opt@#3.#1\endcsname,}%
  \zap@space#2 \@empty}}
```

The last three macros are for handling the filename stack.

```
\def\@pushfilename{%
  \xdef\@currnamestack{%
    {\@currname}%
    {\@currext}%
    {\the\catcode`\@}%
    \@currnamestack}}
\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil}
\def\@p@pfilename#1#2#3#4\@nil{%
  \gdef\@currname{#1}%
  \gdef\@currext{#2}%
  \catcode`\@#3\relax
  \gdef\@currnamestack{#4}}
```

And reset %

```
\catcode`\%=\temp\relax
```

Finished!

Warning! This package deliberately breaks certain functionality. Use only for debugging.

Version History

New features and updates with each version. Listed in no particular order.

1.0 . . . . . September 2026
—initial release