LaTeX

The trace package[*]

Frank Mittelbach

Introduction

When writing new macros one often finds that they do not work as expected (at least I do :-). If this happens and one can't immediately figure out why there is a problem one has to start doing some serious debugging. TeX offers a lot of bells and whistles to control what is being traced but often enough I find myself applying the crude command `\tracingall` which essentially means "give me whatever tracing information is available".

In fact I normally use ε-TeX in such a case, since that TeX extension offers me a number of additional tracing possibilities which I find extremely helpful. The most important ones are `\tracingassigns`, which will show you changes to register values and changes to control sequences when they happen, and `\tracinggroups`, which will tell you what groups are entered or left (very useful if your grouping got out of sync).

So what I really write is

```
\tracingassigns=1\tracinggroups=1\tracingall
```

That in itself is already a nuisance (since it is a mouthful) but there is a worse catch: when using `\tracingall` you do get a awful lot of information and some of it is really useless.

For example, if LaTeX has to load a new font it enters some internal routines of NFSS which scan font definition tables etc. And 99.9% of the time you are not at all interested in that part of the processing but in the two lines before and the five lines after. However, you have to scan through a few hundred lines of output to find the lines you need.

Another example is the `calc` package. A simple statement like `\setlength \linewidth {1cm}` inside your macro will result in

```
\setlength ->\protect \setlength
{\relax}

\setlength  ->\calc@assign@skip

\calc@assign@skip ->\calc@assign@generic \calc@Askip \calc@Bskip

\calc@assign@generic #1#2#3#4->\let \calc@A #1\let \calc@B #2\expandafter \calc
@open \expandafter (#4!\global \calc@A \calc@B \endgroup #3\calc@B
#1<-\calc@Askip
#2<-\calc@Bskip
#3<-\linewidth
#4<-1cm
{\let}
{\let}
{\expandafter}
{\expandafter}

\calc@open (->\begingroup \aftergroup \calc@initB \begingroup \aftergroup \calc
@initB \calc@pre@scan
{\begingroup}
{\aftergroup}
{\begingroup}
{\aftergroup}
```

[*]This file has version number 1.0a trace LaTeX code, last revised 2000/02/16.

```
\calc@pre@scan #1->\ifx (#1\expandafter \calc@open \else \ifx \widthof #1\expan
dafter \expandafter \expandafter \calc@textsize \else \calc@numeric \fi \fi #1
#1<-1
{\ifx}
{false}
{\ifx}
{false}

\calc@numeric ->\afterassignment \calc@post@scan \global \calc@A
{\afterassignment}
{\global}
{\fi}
{\fi}

\calc@post@scan #1->\ifx #1!\let \calc@next \endgroup \else \ifx #1+\let \calc@
next \calc@add \else \ifx #1-\let \calc@next \calc@subtract \else \ifx #1*\let
\calc@next \calc@multiplyx \else \ifx #1/\let \calc@next \calc@dividex \else \i
fx #1)\let \calc@next \calc@close \else \calc@error #1\fi \fi \fi \fi \fi \fi \
calc@next
#1<-!
{\ifx}
{true}
{\let}
{\else}
{\endgroup}
{restoring \calc@next=undefined}

\calc@initB ->\calc@B \calc@A
{\skip44}
{\global}
{\endgroup}
{restoring \skip44=0.0pt}

\calc@initB ->\calc@B \calc@A
{\skip44}
{\dimen27}
```

Do you still remember what I was talking about?

No? We're trying to find a problem in macro code without having to scan too many uninteresting lines. To make this possible we have to redefine a number of key commands to turn tracing off temporarily in the hope that this will reduce the amount of noise during the trace. For example, if we change one of the `calc` internals slightly, the above tracing output can be reduced to:

```
\setlength ->\protect \setlength
{\relax}

\setlength  ->\calc@assign@skip

\calc@assign@skip ->\calc@assign@generic \calc@Askip \calc@Bskip

\calc@assign@generic #1#2#3#4->\let \calc@A #1\let \calc@B #2\expandafter \calc
@open \expandafter (#4!\global \calc@A \calc@B \endgroup #3\calc@B
#1<-\calc@Askip
#2<-\calc@Bskip
#3<-\linewidth
#4<-1cm
{\let}
{\let}
{\expandafter}
{\expandafter}

\calc@open (->\begingroup \conditionally@traceoff \aftergroup \calc@initB \begi
ngroup \aftergroup \calc@initB \calc@pre@scan

\conditionally@traceoff ->\tracingrestores \z@ \tracingcommands \z@ \tracingpag
es \z@ \tracingmacros \z@ \tracingparagraphs \z@
```

```
{\tracingrestores}
{\tracingcommands}
{restoring \tracingrestores=1}

\calc@initB ->\calc@B \calc@A
{\skip44}
{\dimen27}
```

Still a lot of noise but definitely preferable to the original case.

I redefined those internals that I found most annoyingly noisy. There are probably many others that could be treated in a similar fashion, so if you think you found one worth adding please drop me a short note.

* * *

`\traceon` `\traceoff` The package defines the two macros `\traceon` and `\traceoff` to unconditionally turn tracing on or off, respectively. `\traceon` is like `\tracingall` but additionally adds `\tracingassigns` and `\tracinggroups` if the ε-TeX program (in extended mode) is used. And `\traceoff` will turn tracing off again, a command which is already badly missing in plain TeX, since it is often not desirable to restrict the tracing using extra groups in the document.

`\conditionally@traceon` `\conditionally@traceoff` There are also two internal macros that turn tracing on and off, but only if the user requested tracing in the first place. These are the ones that are used internally within the code below.

Since the package overwrites some internals of other packages you should load it as the last package in your preamble using `\usepackage{trace}`.

A sample file

The following small test file shows the benefits of the `trace` package. If one uncomments the line loading the package, the amount of tracing data will be drastically reduced. Without the `trace` package we get 6573 lines in the log file; adding the package will reduce this to 1593 lines.

```
\documentclass{article}
\usepackage{calc}
%\usepackage{trace} % uncomment to see difference

\begin{document}
\ifx\traceon\undefined \tracingall \else \traceon \fi

\setlength\linewidth{1cm}

$foo=\bar a$

\small \texttt{\$}  \stop
```

Implementation

This package is for use with LaTeX (though something similar could be produced for other formats).

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}[1998/12/01]
```

`\if@tracing` We need a switch to determine if we want any tracing at all. Otherwise, if we use `\traceoff`...`\traceon` internally, we would unconditionally turn on tracing even when no tracing was asked for in the first place.

```
\newif\if@tracing
```

`\traceon`
`\conditionally@traceoff`

As stated in the introduction, the amount of tracing being done should depend on the formatter we use. So we first test if we are running with ε-TEX in extended mode. In the latter csse the command `\tracinggroups` is defined.[1]

```
\ifx\tracinggroups\undefined
```

If we are using standard TEX then `\traceon` is more or less another name for `\tracingall`. The only differences are that we set the above `@tracing` switch to true and reorder the assignments within it somewhat so that it will output no tracing information about itself. In contrast, `\tracingall` itself produces

```
{vertical mode: \tracingstats}
{\tracingpages}
{\tracinglostchars}
{\tracingmacros}
{\tracingparagraphs}
{\tracingrestores}
{\errorcontextlines}

\showoutput ->\tracingoutput \@ne \showboxbreadth \maxdimen \showboxdepth \maxd
imen \errorstopmode \showoverfull
{\tracingoutput}
{\showboxbreadth}
{\showboxdepth}
{\errorstopmode}


\showoverfull ->\tracingonline \@ne
{\tracingonline}
```

Which is quite a lot given that none of it is of any help to the task at hand. In contrast `\traceon` will produce nothing whatsoever since the noise generating switches are set at the very end.

```
  \def\traceon{%
```

We start by setting the `@tracing` switch to signal that tracing is asked for. This is then followed by setting the various tracing primitives of TEX.

```
    \@tracingtrue
    \tracingstats\tw@
    \tracingpages\@ne
    \tracinglostchars\@ne
    \tracingparagraphs\@ne
    \errorcontextlines\maxdimen\showoutput
    \tracingmacros\tw@
    \tracingrestores\@ne
    \tracingcommands\tw@
  }
```

Now what should `\conditionally@traceoff` do in this case? Should it revert all settings changed by `\traceon`? It should not, since our goal is to shorten the trace output, thus setting all of the uninteresting values back makes the output unnecessarily longer. Therefore we restrict ourself to those `\tracing...` internals that really contribute to listings like the above.

And one additional point is worth mentioning. The order in which we turn the tracing internals off has effects on the output we see. So what needs to be turned off first? Either `\tracingrestores` or `\tracingcommands`; it makes no difference which, as long as they both come first. This is because those two are the only tracing switches that produce output while tracing the command `\conditionally@traceoff` itself (see example on page 95).

[1] If some package writer has defined that command name for some reason—too bad—then we make the wrong deduction from this fact and as a result the package will fail.

In principle we would need to test the `@tracing` switch to see if there is anything to turn off; after all, this is the conditional trace off. However this would lead to extra output if we are currently tracing so we skip the test and instead accept that in case we are not doing any tracing we unnecessarily set the tracing primitives back to zero (i.e., the value they already have).

```
    \def\conditionally@traceoff{%
      \tracingrestores\z@
      \tracingcommands\z@
      \tracingpages\z@
      \tracingmacros\z@
      \tracingparagraphs\z@
```

As remarked above there are more tracing switches set by `\traceon`, however there is no point in resetting `\tracingstats` or `\tracinglostchars` so we leave them alone.

```
  % \tracingstats\z@
  % \tracinglostchars\z@
```

Since this is the command that only conditionally turns off tracing we do not touch the `@tracing` switch. This way a `\conditionally@traceon` will be able to turn the tracing on again.

```
    }
```

That covers the case for the standard TEX program. If `\tracingsgroups` was defined we assume that we are running with ε-TEX in extended mode.

```
  \else
```

In that case `\traceon` does more than `\tracingall`: it also turns on tracing of assignments and tracing of grouping.[2] To keep tracing at a minimum `\tracingassigns` should be turned on last (in fact like before we disassemble `\tracingall` and reorder it partially).

```
    \def\traceon{%
      \@tracingtrue
      \tracingstats\tw@
      \tracingpages\@ne
      \tracinglostchars\@ne
      \tracingparagraphs\@ne
      \errorcontextlines\maxdimen\showoutput
      \tracingmacros\tw@
      \tracinggroups\@ne
      \tracingrestores\@ne
      \tracingcommands\tw@
      \tracingassigns\@ne
    }
```

When turning tracing off again we now also have to turn off those additional tracing switches. But what to turn off in what order? Since `\tracingassigns` is quite noisy (two lines of output per assignment) and the whole command expansion consists of assignments, we had best start with this switch and follow it again by `\tracingrestores` and `\tracingcommands`. The rest can be in any order, it doesn't make a difference.

With the same reasoning as before we omit testing for the `@tracing` switch and always set the primitives back to zero.

```
    \def\conditionally@traceoff{%
      \tracingassigns\z@
      \tracingrestores\z@
```

[2] These are my personal preference settings; ε-TEX does in fact offer some more tracing switches and perhaps one or or more of them should be added here as well.

```
      \tracingcommands\z@
      \tracingpages\z@
      \tracingmacros\z@
      \tracingparagraphs\z@
      \tracinggroups\z@
    }
```

This concludes the part that depends on the formatter being used.

```
  \fi
```

`\traceoff` `\conditionally@traceon` Above we have defined `\conditionally@traceoff` and `\traceon` so now we have to define their counterparts.

To stop tracing unconditionally we call `\conditionally@traceoff` (which in reality is far from conditional except for not setting the `@tracing` switch :-) and then reset the `@tracing` switch to false.

```
  \def\traceoff{\conditionally@traceoff \@tracingfalse}
```

Now the `\conditionally@traceon` command will look at the `@tracing` switch and if it is true it will call `\traceon` to restart tracing (note that the latter command unnecessarily sets the switch to true as well). The reason for the `\expandafter` is to get rid of the `\fi` primitive which would otherwise show up in the tracing output (and perhaps puzzle somebody).

```
  \def\conditionally@traceon{\if@tracing \expandafter \traceon \fi}
```

The rest of the package now consists of redefinitions of certain commands to make use of `\conditionally@traceoff`.

Taming calc

`\calc@open` Near the start of parsing a calc expression the macro `\calc@open` is called. Since it already involves a group it is perfectly suitable for our task—we don't even have to restart the tracing as this is done automatically for us.

```
  \def\calc@open({\begingroup
     \conditionally@traceoff
     \aftergroup\calc@initB
     \begingroup\aftergroup\calc@initB
     \calc@pre@scan}
```

Making NFSS less noisy

`\define@newfont` Whenever NFSS determines that the font currently asked for is not already loaded, it will start looking through font definition files and then load the font. This results in a very large number of tracing lines which are not normally of interest (unless there is a bug in that area—something we hope should have been found by now). Again the code already contains its own group so we only have to turn the tracing off.

```
  \def\define@newfont{%
    \begingroup
      \conditionally@traceoff
      \let\typeout\@font@info
      \escapechar\m@ne
      \expandafter\expandafter\expandafter
         \split@name\expandafter\string\font@name\@nil
        \try@load@fontshape % try always
      \expandafter\ifx
         \csname\curr@fontshape\endcsname \relax
        \wrong@fontshape\else
        \extract@font\fi
    \endgroup}
```

`\frozen@everymath`
`\frozen@everydisplay`

At the beginning of every math formula NFSS will check whether or not the math fonts are properly set up and if not will load whatever is needed. So we surround that part of the code with `\conditionally@traceoff` and `\conditionally@traceon` thereby avoiding all this uninteresting output.

```
\frozen@everymath =
   {\conditionally@traceoff \check@mathfonts \conditionally@traceon
    \the\everymath}
\frozen@everydisplay =
   {\conditionally@traceoff \check@mathfonts \conditionally@traceon
    \the\everydisplay}
```

Checking for italic corrections

`\maybe@ic@`

When executing `\textit` or its friends, LaTeX looks ahead to determine whether or not to add an italic correction at the end. This involves looping through the `\nocorrlist` which outputs a lot of tracing lines we are normally not interested in. So we disable tracing for this part of the processing.

```
\def \maybe@ic@ {%
  \ifdim \fontdimen\@ne\font>\z@
  \else
    \conditionally@traceoff
    \@tempswatrue
    \expandafter\@tfor\expandafter\reserved@a\expandafter:\expandafter=%
        \nocorrlist
    \do \t@st@ic
    \if@tempswa \sw@slant \fi
    \conditionally@traceon
  \fi
}
```

◇ Frank Mittelbach