1 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 \texttt{\textbackslash tracingall} which essentially means “give me whatever tracing information is available”.

In fact I normally use \texttt{\textasciitilde-\textsc{TeX}} in such a case, since that \textsc{TeX} extension offers me a number of additional tracing possibilities which I find extremely helpful. The most important ones are \texttt{\textbackslash tracingassigns}, which will show you changes to register values and changes to control sequences when they happen, and \texttt{\textbackslash 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

\texttt{\textbackslash tracingassigns=1\textbackslash tracinggroups=1\textbackslash tracingall}

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

For example, if \texttt{\textsc{TeX}} has to load a new font it enters some internal routines of \textsc{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 \texttt{calc} package. A simple statement like \texttt{\setlength \linewidth \{1cm\}} inside your macro will result in

\texttt{\setlength ->\protect \setlength \{\relax\}}
\texttt{\setlength ->\calc@assign@skip}

*This file has version number v1.1e, last revised 2018/10/13.
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 \texttt{calc}
internals slightly, the above tracing output can be reduced to:

\begin{verbatim}
\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 \begingroup \aftergroup \calc@initB \calc@pre@scan
{\begingroup}
\conditionally@traceoff ->\tracingrestores \z@ \tracingcommands \z@ \tracingpages \z@ \tracingmacros \z@ \tracingparagraphs \z@
{\tracingrestores}
{\tracingcommands}
{restoring \tracingrestores=1}
\calc@initB ->\calc@B \calc@A
{\skip44}
{\dimen27}
\endgroup
endgroup}
endgroup
endgroup
endgroup}
endgroup
\end{verbatim}

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.

∗ ∗ ∗

The package defines the two macros \texttt{\traceon} and \texttt{\traceoff} to uncondi-
tionally turn tracing on or off, respectively. \texttt{\traceon} is like \texttt{\tracingall}
but additionally adds \texttt{\tracingassigns} and \texttt{\tracinggroups} if the \texttt{\LaTeX}\XeXeX
program (in extended mode) is used. And \texttt{\traceoff} will turn tracing off again, a com-
mand which is already badly missing in plain \texttt{\LaTeX}, since it is often not desirable
to restrict the tracing using extra groups in the document.

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 \texttt{\usepackage{trace}}.

The package offers the option \texttt{logonly} that suppresses terminal output during
tracing (unless \texttt{\tracingall} is used). This is useful if the \texttt{\LaTeX}\XeXeX
implementation used gets rather slow when writing a lot of information to the terminal.

It also offers the option \texttt{full} in which case \texttt{\traceon} will trace all parts of the
code, i.e., essentially work like \texttt{\tracingall}.
2 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 6594 lines in the log file; adding the package will reduce this to 1618 lines.

\documentclass{article}
\usepackage{calc}
%\usepackage{trace} % uncomment to see difference
\begin{document}
\ifx\traceon\undefined \tracingall \else \traceon \fi
\setlength\linewidth{1cm}$\texttt{\$\}\texttt{\$} \stop

3 Implementation

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

\begin{itemize}
\item \verb|\NeedsTeXFormat{LaTeX2e}[1998/12/01]|
\item The package has a option that suppresses tracing on the terminal, i.e., if used will not set \verb|\tracingonline| to one. This has been added in version 1.1a since some \TeX{} implementations get rather slow when outputting to a terminal.
\item \verb|\DeclareOption{logonly}|\let\tracingonline@p\z@
\item The default is showing the tracing information on the terminal.
\item \let\tracingonline@p\@ne\let\conditionally@traceoff\@empty\let\conditionally@traceon\@empty\let\traceon\tr@ce@n\let\tracingonline\z@
\item If the option \verb|full| is selected then all code should be traced, i.e., the commands \verb|\conditionally@traceoff| and \verb|\conditionally@traceon| should do nothing. We set them to \verb|\empty| not \verb|\relax| since the latter might produce a math ord in certain circumstances. We also have to make sure that \verb|\traceon| (as defined further down) is not redefining \verb|\conditionally@traceoff| again. To make this all work these redefinitions have to wait until the end of the package.
\item \verb|\DeclareOption{full}|{\AtEndOfPackage{\let\conditionally@traceoff\@empty\let\conditionally@traceon\@empty\let\traceon\tr@ce@n}}
\item \verb|\ProcessOptions|\relax\if@tracing\verb|\if\@tracing| We need a switch to determine if we want any tracing at all. Otherwise, if we use \verb|\traceoff|. . . \traceon internally, we would unconditionally turn on tracing even when no tracing was asked for in the first place.
\item \verb|\newif\if@tracing|\verb|\if@tracing|
\traceon This macro ensures that \conditionally@traceoff is actually turning off
switches (since \tracinall might have disabled it) and then calls \tr@ce@n to
setup tracing.

13 \def\traceon{\let\conditionally@traceoff\unconditionally@traceoff
14 \tr@ce@n}

\tr@ce@n As stated in the introduction, the amount of tracing being done should depend on
the formatter we use. Initially first test if we are running with \TeX in extended
mode. In the latter case the command \tracinggroups is defined. But for a
number of years now \TeX only works with \TeX so we drop that part of the
code. For now I leave it in the file together with its documentation, but commented
out.

15 %\ifx\tracinggroups\undefined
If we are using standard \TeX then \tr@ce@n 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 \one \showboxbreadth \maxdimen \showboxdepth \maxd
imen \errorstopmode \showoverfull
\{\tracingoutput\}
\{\showboxbreadth\}
\{\showboxdepth\}
\{\errorstopmode\}
\showoverfull ->\tracingonline \one
\{\tracingonline\}

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

16 % \def\tr@ce@n{% We start by setting the @tracing switch to signal that tracing is asked for. This
17 % is then followed by setting the various tracing primitives of \TeX.
18 % \tracingtrue
19 % \tracingstats\two
20 % \tracingpages\one
21 % \tracinglostchars\one
22 % \tracingparagraphs\one
23 % \errorcontextlines\maxdimen
24 % \tracingoutput\one
25 % \showboxbreadth\maxdimen
26 % \showboxdepth\maxdimen
27 % \errorstopmode
28 % \tracingmacros\two
29 % \tracingrestores\one
30 % \tracingcommands\two
The setting of `\tracingonline` depends on the option `logonly`:

```
30 \% \tracingonline\tracingonline@p
31 \% }
```

Now what should `conditionally@traceoff` do in this case? Should it revert all settings changed by `\trace@n`? 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 ourselves 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 3).

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).

```
32 \% \def\conditionally@traceoff{%
33 \% \tracingrestores\z@
34 \% \tracingcommands\z@
35 \% \tracingpages\z@
36 \% \tracingmacros\z@
37 \% \tracingparagraphs\z@
38 \% \tracingoutput\z@
39 \% \showboxbreadth\m@ne
40 \% \showboxdepth\m@ne
```

As remarked above there are more tracing switches set by `\tr@ce@n`, however there is no point in resetting `\tracinglostchars` so we leave it alone.

```
41 \% \tracingstats\one
42 \% \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.

```
43 \% }
```

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

```
44 \%else
```

In that case `\tr@ce@n` does more than `\tracingall`: it also turns on tracing of assignments and tracing of grouping.\textsuperscript{1} To keep tracing at a minimum `\tracingassigns` should be turned on last (in fact like before we disassemble `\tracingall` and reorder it partially).

```
45 \def\tr@ce@n{%
46 \tracingtrue
47 \tracingstats\tw@
```

\textsuperscript{1}These are my personal preference settings; ε-TεX does in fact offer some more tracing switches and perhaps one or more of them should be added here as well.
We only change \texttt{\tracingoutput} if it hasn’t already been enabled by \texttt{\showoutput}. If that’s not the case, we set it to 2 so that we can distinguish the two cases.

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 \texttt{\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 \texttt{\tracingrestores} and \texttt{\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 @texttt{\tracing} switch and always set the primitives back to zero.

This concludes the part that depends on the formatter being used.
\unconditionally@traceoff  A saved version of whatever \conditionally@traceoff  was defined to be. We need this since the latter might get disabled by \tracingall  or by the full option.

\tracingall  We redefine \tracingall  to trace the same stuff than \tracingon  (i.e., more when \LaTeX{} is being used) and ensure that everything gets traced by disabling \conditionally@traceoff. And, of course, \tracingall  should always report on the terminal.

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

To stop tracing unconditionally we call \unconditionally@traceoff  and then reset the @tracing switch to false.

\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.

3.1 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.

3.2 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.
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\}

4 Checking for italic corrections

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.

\maybe@ic@ {\
def \maybe@ic@ {\
ifdim \fontdimen\one\font>\z@ 
else 
  \conditionally@traceoff \maybe@ictrue 
  \expandafter\@tfor\expandafter\reserved@a\expandafter:=% 
  \nocorrlist 
  \do \t@stop@ic 
  \ifmaybe@ic \sw@slant \fi 
  \conditionally@traceon 
  \fi 
  \fi 
  \} 
(/package)