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 \varepsilon\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 \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

\begin{verbatim}
\tracingassigns=1\tracinggroups=1\tracingall
\end{verbatim}

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 \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 \texttt{calc} package. A simple statement like \texttt{\setlength linewidth \texttt{(}1\texttt{cm)}} inside your macro will result in

\begin{verbatim}
\setlength ->\protect \setlength
\relax
\end{verbatim}

\begin{verbatim}
\setlength ->\calc@assign@skip
\end{verbatim}
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 \begi
group \aftergroup \calc@initB \calc@pre@scan
{\begingroup}
\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}
\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{\textbackslash traceon} and \texttt{\textbackslash traceoff} to uncondi-
tionally turn tracing on or off, respectively. \texttt{\textbackslash traceon} is like \texttt{\textbackslash tracingall} but
additionally adds \texttt{\textbackslash tracingassignments} and \texttt{\textbackslash tracinggroups} if the \texttt{\epsilon-\TeX} program
(in extended mode) is used. And \texttt{\textbackslash traceoff} will turn tracing off again, a com-
mand which is already badly missing in plain \TeX, 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{\textbackslash tracingall} is used). This is useful if the \TeX 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{\textbackslash traceon} will trace all parts of the
code, i.e., essentially work like \texttt{\textbackslash tracingall}. 

3
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}
$foo=\bar a$
\small \texttt{$} \stop

3 Implementation

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

\begin{verbatim}
\NeedsTeXFormat{LaTeX2e}[1998/12/01]

The package has a option that suppresses tracing on the terminal, i.e., if used will not set \texttt{\tracingonline} to one. This has been added in version 1.1a since some \TeX implementations get rather slow when outputting to a terminal.

\DeclareOption{logonly}
\{
\let\tracingonline@p\z@\}

The default is showing the tracing information on the terminal.

\let\tracingonline@p\@ne

If the option \texttt{full} is selected then all code should be traced, i.e., the commands \texttt{\conditionally@traceoff} and \texttt{\conditionally@traceon} should do nothing. We set them to \texttt{\empty} not \texttt{\relax} since the latter might produce a math ord in certain circumstances. We also have to make sure that \texttt{\traceon} (as defined further down) is not redefining \texttt{\conditionally@traceoff} again. To make this all work these redefinitions have to wait until the end of the package.

\DeclareOption{full}
\{
\AtEndOfPackage\{
\let\conditionally@traceoff\empty
\let\conditionally@traceon\empty
\let\traceon\tr@ce@n
\}
\}
\ProcessOptions\relax
\if@tracing
We need a switch to determine if we want any tracing at all. Otherwise, if we use \texttt{\traceoff...\traceon} internally, we would unconditionally turn on tracing even when no tracing was asked for in the first place.
\newif\if@tracing

\end{verbatim}

\texttt{\if\tracing}

\texttt{\end{document}}
\traceon This macro ensures that \conditionally@traceoff is actually turning off
switches (since \tracingall 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 only 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
is then followed by setting the various tracing primitives of \tex.
17 \% \@tracingtrue
18 \% \tracingstats\tw@ 19 \% \tracingpages\@ne 20 \% \tracinglostchars\@ne 21 \% \tracingparagraphs\@ne 22 \% \errorcontextlines\maxdimen 23 \% \tracingoutput\@ne 24 \% \showboxbreadth\maxdimen 25 \% \showboxdepth\maxdimen 26 \% \errorstopmode 27 \% \tracingmacros\tw@ 28 \% \tracingrestores\@ne 29 \% \tracingcommands\tw@
The setting of \texttt{tracingonline} depends on the option logonly:

\begin{verbatim}
30 \% \texttt{\textbackslash tracingonline}\texttt{tracingonline@p}
31 \% }

Now what should \texttt{conditionally@traceoff} do in this case? Should it re-
vert all settings changed by \texttt{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 \texttt{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 \texttt{tracingrestores} or \texttt{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
\texttt{conditionally@traceoff} itself (see example on page 3).

In principle we would need to test the \texttt{tracing} switch to see if there is any-
thing 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).

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

As remarked above there are more tracing switches set by \texttt{trace@n}, however
there is no point in resetting \texttt{tracinglostchars} so we leave it alone.

\begin{verbatim}
41 \% \texttt{\textbackslash tracingstats\textbackslash One}
42 \% \texttt{\textbackslash tracinglostchars\textbackslash z@}
\end{verbatim}

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

\begin{verbatim}
43 \% }
\end{verbatim}

That covers the case for the standard \TeX{} program. If \texttt{tracingsgroups} was
defined we assume that we are running with \vTeX{} in extended mode.

\begin{verbatim}
44 \texttt{\textbackslash else}

In that case \texttt{trace@n} does more than \texttt{tracingall}: it also turns on tracing
of assignments and tracing of grouping.\footnote{These are my personal preference settings; \vTeX{} does in fact offer some more tracing switches
and perhaps one or more of them should be added here as well.} To keep tracing at a minimum
\texttt{tracingassigns} should be turned on last (in fact like before we disassemble
\texttt{tracingall} and reorder it partially).
\begin{verbatim}
45 \texttt{\edef\texttt{\textbackslash ce@n}{\%}
46 \texttt{\noexpand\textbackslash tracingtrue}
47 \texttt{\textbackslash tracingstats\textbackslash tw@}
\end{verbatim}
We only change \tracingoutput if it hasn’t already been enabled by \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 \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.

If \tracingoutput is 2 it was set above, if it is 1 it was set by \showoutput and we leave it alone and if it is 0 there is nothing to do as well.

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.
85  \let\unconditionally@traceoff\conditionally@traceoff

\tracingall  We redefine \tracingall to trace the same stuff than \tracingon (i.e., more when $c$-TEX is being used) and ensure that everything gets traced by disabling \conditionally@traceoff. And, of course, \tracingall should always report on the terminal.
86  \def\tracingall{\let\conditionally@traceoff@empty
87    \let\tracingonline@p\one
88    \tracingon
89  }

\traceoff  Above we have defined \conditionally@traceoff and \tracingon so now we have to define their counterparts.
To stop tracing unconditionally we call \unconditionally@traceoff and then reset the @tracing switch to false.
90  \def\traceoff{\unconditionally@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).
91  \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.

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.
92  \def\calc@open{\begingroup
93    \conditionally@traceoff
94    \aftergroup\calc@initB
95    \begingroup\aftergroup\calc@initB
96    \calc@pre@scan}

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.
97  \def\define@newfont{%
98    \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}

4 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
\maybe@ictrue
\expandafter\@tfor\expandafter\reserved@a\expandafter=%
\nocorrlist
\do \t@st@ic\ifmaybe@ic \sw@slant \fi
\conditionally@traceon
\fi
}
\frozen@everymath