The trace package*

Frank Mittelbach
LaTeX3 project
frank.mittelbach@latex-project.org

2003/04/30

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 `\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
```

*This file has version number v1.1c, last revised 2003/04/30.

```
@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}

\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 \
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
{\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}
```

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}`.

The package offers the option `logonly` that suppresses terminal output during tracing (unless `\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 `full` in which case `\traceon` will trace all parts of the code, i.e., essentially work like `\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}

$foo=\bar a$

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