Name

groff_trace – macros for debugging GNU *roff* documents

Synopsis

groff –m trace [*option* . . .] [*file* . . .]

Description

trace is a macro package for the *groff*(7) document formatting system, designed as an aid for debugging documents written in its language. It issues a message to the standard error stream upon entry to and exit from each macro call. This can ease the process of isolating errors in macro definitions.

Activate the package by specifying the command-line option "**–m trace**" to the formatter program (often *groff*(1)). You can achieve finer control by including the macro file within the document; invoke the **mso** request, as in "**.mso trace.tmac**". Only macros that are defined after this invocation are traced. If the **trace–full** register is set to a true value, as with the command-line option "**–r trace–full=1**", register and string assignments, along with some other requests, are traced also. If another macro package should be traced as well, specify it after "**–m trace**" on the command line.

The macro file *trace.tmac* is unusual because it does not contain any macros to be called by a user. Instead, *groff*'s macro definition and alteration facilities are wrapped such that they display diagnostic messages.

Limitations

Because *trace.tmac* wraps the **de** request (and its cousins), macro arguments are expanded one level more. This causes problems if an argument uses four or more backslashes to delay interpretation of an escape sequence. For example, the macro call

```
.foo \\\\n[bar]
```

normally passes "\\n[bar]" to macro "foo", but with **de** redefined, it passes "\n[bar]" instead.

The solution to this problem is to use *groff*'s **\E** escape sequence, an escape character that is not interpreted in copy mode.

```
.foo \En[bar]
```

Examples

We will illustrate *trace.tmac* using the shell's "here document" feature to supply *groff* with a document on the standard input stream. Since we are interested only in diagnostic messages appearing on the standard error stream, we discard the formatted output by redirecting the standard output stream to */dev/null*.

Observing nested macro calls

Macro calls can be nested, even with themselves. Tracing recurses along with them; this feature can help to detangle complex call stacks.

```
$ cat <<EOF | groff -m trace > /dev/null
.de countdown
. nop \\$1
. nr count (\\$1 - 1)
. if \\n[count] .countdown \\n[count]
..
.countdown 3
blastoff
EOF
 *** .de countdown
 *** de trace enter: .countdown "3"
  *** de trace enter: .countdown "2"
   *** de trace enter: .countdown "1"
   *** trace exit: .countdown "1"
  *** trace exit: .countdown "2"
 *** trace exit: .countdown "3"
```

Tracing with the mso request

Now let us activate tracing within the document, not with a command-line option. We might do this when using a macro package like *ms* or *mom*, where we may not want to be distracted by traces of macros we didn't write.

```
$ cat <<EOF | groff -ms > /dev/null
.LP
This is my introductory paragraph.
.mso trace.tmac
.de Mymac
..
.Mymac
.PP
Let us review the existing literature.
EOF
 *** .de Mymac
 *** de trace enter: .Mymac
 *** trace exit: .Mymac
```

As tracing was not yet active when the macros "LP" and "PP" were defined (by *s.tmac*), their calls were not traced; contrast with the macro "Mymac".

Files

/usr/local/share/groff/1.23.0/tmac/trace.tmac
: implements the package.

Authors

trace.tmac was written by James Clark. This document was written by Bernd Warken ⟨groff–bernd .warken–72@web.de⟩ and G. Branden Robinson ⟨g.branden.robinson@gmail.com⟩.

See also

Groff: The GNU Implementation of troff, by Trent A. Fisher and Werner Lemberg, is the primary *groff* manual. You can browse it interactively with "info groff".

groff (1)
: gives an overview of the *groff* document formatting system.

troff (1) supplies details of the **–m** command-line option.

groff_tmac(5)
: offers a survey of *groff* macro packages.

groff (7)
: is a reference manual for the *groff* language.