Name

groff_tmac – macro files in the GNU *roff* typesetting system

Description

Definitions of macros, strings, and registers for use in a *roff*(7) document can be collected into *macro files*, *roff* input files designed to produce no output themselves but instead ease the preparation of other *roff* documents. There is no syntactical difference between a macro file and any other *roff* document; only its purpose distinguishes it. When a macro file is installed at a standard location, named according to a certain convention, and suitable for use by a general audience, it is termed a *macro package*. Macro packages can be loaded by supplying the **–m** option to *troff*(1) or a *groff* front end.

Each macro package stores its macro, string, and register definitions in one or more *tmac* files. This name originated in early Unix culture as an abbreviation of "*troff* macros".

A macro file must have a name in the form name.*tmac* (or *tmac.*name) and be placed in a "*tmac* directory" to be loadable with the **–m***name* option. Section "Environment" of *troff*(1) lists these directories. Alternatively, a *groff* document requiring a macro file can load it with the **mso** ("macro source") request.

Like any other *roff* document, a macro file can use the "**so**" request ("source") to load further files relative to its own location.

Macro files are named for their most noteworthy application, but a macro file need not define any macros. It can restrict itself to defining registers and strings or invoking other *groff* requests. It can even be empty.

Macro packages

Macro packages come in two varieties; those which assume responsibility for page layout and other critical functions ("major" or "full-service") and those which do not ("supplemental" or "auxiliary"). GNU *roff* provides most major macro packages found in AT&T and BSD Unix systems, an additional full-service package, and many supplemental packages. Multiple full-service macro packages cannot be used by the same document. Auxiliary packages can generally be freely combined, though attention to their use of the *groff* language name spaces for identifiers (particularly registers, macros, strings, and diversions) should be paid. Name space management was a significant challenge in AT&T *troff*; *groff*'s support for arbitrarily long identifiers affords few excuses for name collisions, apart from attempts at compatibility with the demands of historical documents.

Man pages

an
man
: *an* is used to compose man pages in the format originating in Version 7 Unix (1979). It has a small macro interface and is widely used; see *groff_man*(7).

doc
mdoc
: *doc* is used to compose man pages in the format originating in 4.3BSD-Reno (1990). It provides many more features than *an*, but is also larger, more complex, and not as widely adopted; see *groff_mdoc*(7).

Because readers of man pages often do not know in advance which macros are used to format a given document, a wrapper is available.

andoc
mandoc
: This macro file, specific to *groff*, recognizes whether a document uses *man* or *mdoc* format and loads the corresponding macro package. Multiple man pages, in either format, can be handled; *andoc* reloads each macro package as necessary.

Full-service packages

The packages in this section provide a complete set of macros for writing documents of any kind, up to whole books. They are similar in functionality; it is a matter of taste which one to use.

me The classical *me* macro package; see *groff_me*(7).

mm The semi-classical *mm* macro package; see *groff_mm*(7).

mom The *mom* macro package, only available in groff. As this was not based on other packages, it was freely designed as quite a nice, modern macro package. See *groff_mom*(7).

ms The classical *ms* macro package; see *groff_ms*(7).

Localization packages

For Western languages, the localization file sets the hyphenation mode and loads hyphenation patterns and exceptions. Localization files can also adjust the date format and provide translations of strings used by some of the full-service macro packages; alter the input encoding (see the next section); and change the amount of additional inter-sentence space. For Eastern languages, the localization file defines character classes and sets flags on them. By default, *troffrc* loads the localization file for English.

trans loads localized strings used by various macro packages after their localized forms have been prepared by a localization macro file.

groff provides the following localization files.

cs Czech; localizes *man*, *me*, *mm*, *mom*, and *ms*. Sets the input encoding to Latin-2 by loading *latin2.tmac*.

de
den German; localizes *man*, *me*, *mm*, *mom*, and *ms*. Sets the input encoding to Latin-1 by loading *latin1.tmac*.

de.tmac selects hyphenation patterns for traditional orthography, and *den.tmac* does the same for the new orthography ("Rechtschreibreform").

en English.

fr French; localizes *man*, *me*, *mm*, *mom*, and *ms*. Sets the input encoding to Latin-9 by loading *latin9.tmac*.

it Italian; localizes *man*, *me*, *mm*, *mom*, and *ms*.

ja Japanese.

sv Swedish; localizes *man*, *me*, *mm*, *mom*, and *ms*. Sets the input encoding to Latin-1 by loading *latin1.tmac*. Some of the localization of the *mm* package is handled separately; see *groff_mmse*(7).

zh Chinese.

Input encodings

latin1
latin2
latin5
latin9 are various ISO 8859 input encodings supported by *groff*. On systems using ISO character encodings, *groff* loads *latin1.tmac* automatically at startup. A document that uses Latin-2, Latin-5, or Latin-9 can specify one of these alternative encodings.

cp1047 provides support for EBCDIC-based systems. On those platforms, *groff* loads *cp1047.tmac* automatically at startup.

Because different input character codes constitute valid GNU *troff* input on ISO and EBCDIC systems, the *latin* macro files cannot be used on EBCDIC systems, and *cp1047* cannot be used on ISO systems.

Auxiliary packages

The macro packages in this section are not intended for stand-alone use, but can add functionality to any other macro package or to plain ("raw") *groff* documents.

62bit provides macros for addition, multiplication, and division of 62-bit integers (allowing safe multiplication of signed 31-bit integers, for example).

hdtbl allows the generation of tables using a syntax similar to the HTML table model. This Heidelberger table macro package is not a preprocessor, which can be useful if the contents of table entries are determined by macro calls or string interpolations. Compare to *tbl*(1). It works only with the **ps** and **pdf** output devices. See *groff_hdtbl*(7).

papersize
enables the paper format to be set on the command line by giving a "**−d paper=***format*" option to *troff*. Possible values for *format* are the ISO and DIN formats "**A0**–**A6**", "**B0**–**B6**", "**C0**–**C6**", and "**D0**–**D6**"; the U.S. formats "**letter**", "**legal**", "**tabloid**", "**ledger**", "**statement**", and "**executive**"; and the envelope formats "**com10**", "**monarch**", and "**DL**". All formats, even those for envelopes, are in portrait orientation: the length measurement is vertical. Appending "l" (ell) to any of these denotes landscape orientation instead. This macro file assumes one-inch horizontal margins, and sets registers recognized by the *groff man*, *mdoc*, *mm*, *mom*, and *ms* packages to configure them accordingly. If you want different margins, you will need to use those packages' facilities, or *troff* **ll** and/or **po** requests to adjust them. An output device typically requires command-line options **−p** and **−l** to override the paper dimensions and orientation, respectively, defined in its *DESC* file; see subsection "Paper format" of *groff*(1). This macro file is normally loaded at startup by the *troffrc* file when formatting for a typesetting device (but not a terminal).

pdfpic provides a single macro, **PDFPIC**, to include a PDF graphic in a document using features of the **pdf** output driver. For other output devices, **PDFPIC** calls **PSPIC**, with which it shares an interface (see below). This macro file is normally loaded at startup by the *troffrc* file.

pic supplies definitions of the macros **PS**, **PE**, and **PF**, usable with the *pic*(1) preprocessor. They center each picture. Use it if your document does not use a full-service macro package, or that package does not supply working *pic* macro definitions. Except for *man* and *mdoc*, those provided with *groff* already do so (exception: *mm* employs the name **PF** for a different purpose).

pspic provides a macro, **PSPIC**, that includes a PostScript graphic in a document. The **ps**, **dvi**, **html**, and **xhtml** output devices support such inclusions; for all other drivers, the image is replaced with a rectangular border of the same size. *pspic.tmac* is loaded at startup by the *troffrc* file.

Its syntax is as follows.

.PSPIC [**−L**|**−R**|**−C**|**−I** *n*] *file* [*width* [*height*]]

file is the name of the PostScript file; *width* and *height* give the desired width and height of the image. If neither a *width* nor a *height* argument is specified, the image's natural width (as given in the file's bounding box) or the current line length is used as the width, whatever is smaller. The *width* and *height* arguments may have scaling units attached; the default scaling unit is **i**. **PSPIC** scales the graphic uniformly in the horizontal and vertical directions so that it is no more than *width* wide and *height* high. Option **−C** centers the graphic horizontally; this is the default. **−L** and **−R** left- and right-align the graphic, respectively. **−I** indents the graphic by *n* (with a default scaling unit of **m**).

To use **PSPIC** within a diversion, we recommend extending it with the following code, assuring that the diversion's width completely covers the image's width.

```
.am PSPIC
.  vpt 0
\h'(\\n[ps-offset]u + \\n[ps-deswid]u)'
.  sp -1
.  vpt 1
..
```

Failure to load **PSPIC**'s image argument is not an error. (The **psbb** request does issue an error diagnostic.) To make such a failure fatal, append to the **pspic*error−hook** macro.

```
.am pspic*error-hook
.  ab
..
```

ptx provides a macro, **xx**, to format permuted index entries as produced by the GNU *ptx*(1) program. If your formatting needs differ, copy the macro into your document and adapt it to your needs.

rfc1345
defines special character escape sequences named for the glyph mnemonics specified in RFC 1345 and the digraph table of the Vim text editor. See *groff_rfc1345*(7).

sboxes offers an interface to the "**pdf: background**" device control command supported by *gropdf*(1). Using this package, *groff ms* documents can draw colored rectangles beneath any output.

.BOXSTART SHADED *color* **OUTLINED** *color* **INDENT** *size* **WEIGHT** *size*
begins a box, where the argument after **SHADED** gives the fill color and that after **OUTLINED** the border color. Omit the former to get a borderless filled box and the latter for a border with no fill. The specified **WEIGHT** is used if the box is **OUTLINED**.

INDENT precedes a value which leaves a gap between the border and the contents inside the box.

Each *color* must be a defined *groff* color name, and each *size* a valid *groff* numeric expression. The keyword/value pairs can be specified in any order.

Boxes can be stacked, so you can start a box within another box; usually the later boxes would be smaller than the containing box, but this is not enforced. When using **BOXSTART**, the left position is the current indent minus the **INDENT** in the command, and the right position is the left position (calculated above) plus the current line length and twice the indent.

.BOXSTOP
takes no parameters. It closes the most recently started box at the current vertical position after adding its **INDENT** spacing.

Your *groff* documents can conditionally exercise the *sboxes* macros. The register **GSBOX** is defined if the package is loaded, and interpolates a true value if the **pdf** output device is in use.

sboxes furthermore hooks into the *groff_ms*(7) package to receive notifications when footnotes are growing, so that it can close boxes on a page before footnotes are printed. When that condition obtains, *sboxes* will close open boxes two points above the footnote separator and re-open them on the next page. (This amount probably will not match the box's **INDENT**.)

See "Using PDF boxes with *groff* and the *ms* macros" ⟨file:///usr/local/share/doc/groff−1.23.0/msboxes.pdf⟩ for a demonstration.

trace aids the debugging of *groff* documents by tracing macro calls. See *groff_trace*(7).

www defines macros corresponding to HTML elements. See *groff_www*(7).

Naming

AT&T *nroff* and *troff* were implemented before the conventions of the modern C *getopt*(3) call evolved, and used a naming scheme for macro packages that looks odd to modern eyes. Macro packages were typically loaded using the **−m** option to the formatter; when directly followed by its argument without an intervening space, this looked like a long option preceded by a single minus—a sensation in the computer stone age. Macro packages therefore came to be known by names that started with the letter "m", which was omitted from the name of the macro file as stored on disk. For example, the manuscript macro package was stored as *tmac.s* and loaded with the option **−ms**.

groff commands permit space between an option and its argument. The syntax "**groff −m s**" makes the macro file name more clear but may surprise users familiar with the original convention, unaware that the package's "real" name was "s" all along. For such packages of long pedigree, *groff* accommodates different users' expectations by supplying wrapper macro files that load the desired file with **mso** requests. Thus, all of "**groff −m s**", "**groff −m ms**", "**groff −ms**", and "**groff −mms**" serve to load the manuscript macros.

Wrappers are not provided for packages of more recent vintage, like *www.tmac*.

As noted in passing above, AT&T *troff* named macro files in the form *tmac.*name. It has since become conventional in operating systems to use a suffixed file name extension to suggest a file type or format.

Inclusion

The traditional method of employing a macro package is to specify the **−m** *package* option to the formatter, which then reads *package*'s macro file prior to any input files. Historically, *package* was sought in a file named *tmac.*package (that is, with a "**tmac.**" prefix). GNU *troff* searches for package*.tmac* in the macro path; if not found, it looks for *tmac.*package instead, and vice versa.

Alternatively, one could include a macro file by using the request "**.so** *file-name*" in the document; *file-name* is resolved relative to the location of the input document. GNU *troff* offers an improved feature in the similar request "**mso** *package-file-name*", which searches the macro path for *package-file-name*. Because its argument is a file name, its "**.tmac**" component must be included for the file to be found; however, as a convenience, if opening it fails, **mso** strips any such suffix and tries again with a "**tmac.**" prefix, and vice versa.

If a sourced file requires preprocessing, for example if it includes *tbl* tables or *eqn* equations, the preprocessor *soelim*(1) must be used. This can be achieved with a pipeline or, in *groff*, by specifying the **−s** option to the formatter (or front end). *man*(1) librarian programs generally call *soelim* automatically. (Macro packages themselves generally do not require preprocessing.)

Writing macros

A *roff*(7) document is a text file that is enriched by predefined formatting constructs, such as requests, escape sequences, strings, numeric registers, and macros from a macro package. These elements are described in *roff*(7).

To give a document a personal style, it is most useful to extend the existing elements by defining some macros for repeating tasks; the best place for this is near the beginning of the document or in a separate file.

Macros without arguments are just like strings. But the full power of macros occurs when arguments are passed with a macro call. Within the macro definition, the arguments are available as the escape sequences **\$1**, . . ., **\$9**, **\$[**. . .**]**, **\$***, and **\$@**, the name under which the macro was called is in **\$0**, and the number of arguments is in register **\n[.$]**; see *groff*(7).

Draft mode

Writing groff macros is easy when the escaping mechanism is temporarily disabled. In groff, this is done by enclosing the macro definition(s) within a pair of **.eo** and **.ec** requests. Then the body in the macro definition is just like a normal part of the document — text enhanced by calls of requests, macros, strings, registers, etc. For example, the code above can be written in a simpler way by

```
.eo
.ds midpart was called with the following
.de print_args
\f[I]\$0\f[] \*[midpart] \n[.$] arguments:
\$*
..
.ec
```

Unfortunately, draft mode cannot be used universally. Although it is good enough for defining normal macros, draft mode fails with advanced applications, such as indirectly defined strings, registers, etc. An optimal way is to define and test all macros in draft mode and then do the backslash doubling as a final step; do not forget to remove the *.eo* request.

Tips for macro definitions

• Start every line with a dot, for example, by using the groff request **.nop** for text lines, or write your own macro that handles also text lines with a leading dot.

```
.de Text
.  if (\\n[.$] == 0) \
.    return
.  nop \)\\$*\)
..
```

- Write a comment macro that works both for copy and draft modes; since the escape character is off in draft mode, trouble might occur when comment escape sequences are used. For example, the following macro just ignores its arguments, so it acts like a comment line:

  ```
  .de c
  ..
  .c This is like a comment line.
  ```

- In long macro definitions, make ample use of comment lines or almost-empty lines (this is, lines which have a leading dot and nothing else) for a better structuring.
- To increase readability, use groff's indentation facility for requests and macro calls (arbitrary whitespace after the leading dot).

Diversions

Diversions can be used to implement quite advanced programming constructs. They are comparable to pointers to large data structures in the C programming language, but their usage is quite different.

In their simplest form, diversions are multi-line strings, but diversions get their power when used dynamically within macros. The (formatted) information stored in a diversion can be retrieved by calling the diversion just like a macro.

Most of the problems arising with diversions can be avoided if you remember that diversions always store complete lines. Using diversions when the line buffer has not been flushed produces strange results; not knowing this, many people get desperate about diversions. To ensure that a diversion works, add line breaks at the right places. To be safe, enclose everything that has to do with diversions within a pair of line breaks; for example, by explicitly using **.br** requests. This rule should be applied to diversion definition, both inside and outside, and to all calls of diversions. This is a bit of overkill, but it works nicely.

(If you really need diversions which should ignore the current partial line, use environments to save the current partial line and/or use the **.box** request.)

The most powerful feature using diversions is to start a diversion within a macro definition and end it within another macro. Then everything between each call of this macro pair is stored within the diversion and can be manipulated from within the macros.

Authors

This document was written by Bernd Warken ⟨groff−bernd.warken−72@web.de⟩, Werner Lemberg ⟨wl@gnu.org⟩, 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".

The Filesystem Hierarchy Standard ⟨https://wiki.linuxfoundation.org/lsb/fhs⟩ is maintained by the Linux Foundation.

groff(1)
  is an overview of the *groff* system.

groff_man(7),
groff_mdoc(7),
groff_me(7),
groff_mm(7),
groff_mom(7),
groff_ms(7),
groff_rfc1345(7),
groff_trace(7),
  and
groff_www(7)
  are *groff* macro packages.

groff(7)
: summarizes the language recognized by GNU *troff*.

troff(1)
: documents the default macro file search path.