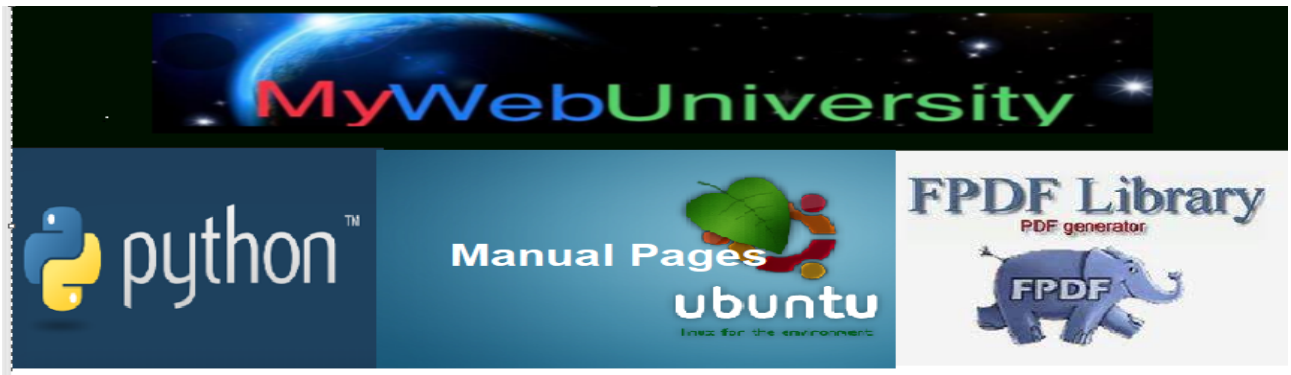

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'grog.1'

$ man grog.1

grog(1) General Commands Manual grog(1)

Name

grog - ?groff guess??infer the groff command a document requires

Synopsis

grog [--run] [--ligatures] [groff-option ...] [--] [file ...]

grog -h

grog --help

grog -v

grog --version

Description

grog reads its input and guesses which groff(1) options are needed to render it. If no operands are given, or if file is ?-?, grog reads the standard input stream. The corre? sponding groff command is normally written to the standard output stream. With the option --run, the inferred command is written to the standard error stream and then executed.

Options

-h and --help display a usage message, whereas -v and --version display version information; all exit afterward.

--ligatures

includes the arguments -P-y -PU in the inferred groff command. These are supported only by the pdf output device.

--run writes the inferred command to the standard error stream and then executes it.

All other specified short options (that is, arguments beginning with a minus sign ?-? fol? lowed by a letter) are interpreted as groff options or option clusters with or without an

option argument. Such options are included in the constructed groff command line.

Details

grog reads each file operand, pattern-matching strings that are statistically likely to be characteristic of roff(7) documents. It tries to guess which of the following groff options are required to correctly render the input: -e, -g, -G, -j, -p, -R, -t (preprocessors); and -man, -mdoc, -mdoc-old, -me, -mm, -mom, and -ms (macro packages). The inferred groff com? mand including these options and any file parameters is written to the standard output stream.

It is possible to specify arbitrary groff options on the command line. These are included in the inferred command without change. Choices of groff options include -C to enable AT&T troff compatibility mode and -T to select a non-default output device. If the input is not encoded in US-ASCII, ISO 8859-1, or IBM code page 1047, specification of a groff option to run the preconv(1) preprocessor is advised; see the -D, -k, and -K options of groff(1). For UTF-8 input, -k is a good choice.

groff may issue diagnostic messages when an inappropriate -m option, or multiple conflicting ones, are specified. Consequently, it is best to specify no -m options to grog unless it cannot correctly infer all of the -m arguments a document requires. A roff document can also be written without recourse to any macro package. In such cases, grog will infer a groff command without an -m option.

Limitations

grog presumes that the input does not change the escape, control, or no-break control char? acters. grog does not parse roff input line continuation or control structures (brace es? cape sequences and the ?if?, ?ie?, and ?el? requests) nor groff's ?while?. Thus the input

```
.if \
t .NH 1
.if n .SH
Introduction
```

will conceal the use of the ms macros NH and SH from grog. Such constructions are regarded by grog's implementors as insufficiently common to cause many inference problems. Pre? processors can be even stricter when matching macro calls that bracket the regions of an in? put file they replace. pic, for example, requires PS, PE, and PF calls to immediately fol? low the default control character at the beginning of a line.

Detection of the -s option (the soelim(1) preprocessor) is tricky; to correctly infer its

necessity would require grog to recursively open all files given as arguments to the .so re? quest under the same conditions that soelim itself does so; see its man page. Recall that soelim is necessary only if sourced files need to be preprocessed. Therefore, as a workaround, you may want to run the input through soelim manually, piping it to grog, and compare the output to running grog on the input directly. If the ?soelim?ed input causes grog to infer additional preprocessor options, then -s is likely necessary.

```
$ printf ".TS\nl.\nI'm a table.\n.TE\n" > 3.roff
$ printf ".so 3.roff\n" > 2.roff
$ printf ".XP\n.so 2.roff\n" > 1.roff
$ grog 1.roff
groff -ms 1.roff
$ soelim 1.roff | grog
groff -t -ms -
```

In the foregoing example, we see that this procedure enabled grog to detect tbl(1) macros, so we would add -s as well as the detected -t option to a revised grog or groff command.

```
$ grog -st 1.roff
groff -st -ms 1.roff
```

Exit status

grog exits with error status 1 if a macro package appears to be in use by the input docu? ment, but grog was unable to infer which one, or 2 if there were problems handling an option or operand. It otherwise exits with status 0. (If the --run option is specified, groff's exit status is discarded.) Inferring no preprocessors or macro packages is not an error condition; a valid roff document need not use either. Even plain text is valid input, if one is mindful of the syntax of the control and escape characters.

Examples

Running

```
grog /usr/share/doc/groff-base/meintro.me
```

at the command line results in

```
groff -me /usr/share/doc/groff-base/meintro.me
```

because grog recognizes that the file meintro.me is written using macros from the me pack? age. The command

```
grog /usr/share/doc/groff-base/pic.ms
```

outputs

groff -e -p -t -ms /usr/share/doc/groff-base/pic.ms

on the other hand. Besides discerning the ms macro package, grog recognizes that the file pic.ms additionally needs the combination of -t for tbl, -e for eqn, and -p for pic.

Consider a file doc/grnexampl.me, which uses the grn preprocessor to include a gremlin(1) picture file in an me document. Let's say we want to suppress color output, produce a DVI file, and get backtraces for any errors that troff encounters. The command

grog -bc -Idoc -Tdvi doc/grnexmpl.me

is processed by grog into

groff -bc -Idoc -Tdvi -e -g -me doc/grnexmpl.me

where we can see that grog has inferred the me macro package along with the eqn and grn pre? processors. (The input file is located in /usr/share/doc/groff-base if you'd like to try this example yourself.)

Authors

grog was originally written in Bourne shell by James Clark. The current implementation in Perl was written by Bernd Warken and heavily revised by G. Branden Robinson.

See also

groff(1)