NAME

groffer - display groff files and man pages on X and tty

SYNOPSIS

groffer [**--**] [**filespec** ...]

groffer [**mode-option** ...] [**groff-options** ...] [**man-options** ...] [**X-options** ...] [**--**] [-**filespec** ...]

groffer -h | --help

groffer -v | --version

DESCRIPTION

The **groffer** program is the easiest way to use groff(1). It can display arbitrary documents written in the *groff* language, see groff(7), or other *roff* languages, see roff(7), that are compatible to the original *troff* language. It finds and runs all necessary *groff* preprocessors, such as **chem**.

The **groffer** program also includes many of the features for finding and displaying the `Unix` manual pages (*man pages*), such that it can be used as a replacement for a man(1) program. Moreover, compressed files that can be handled by gzip(1) or bzip2(1) are decompressed on-the-fly.

The normal usage is quite simple by supplying a file name or name of a *man page* without further options. But the option handling has many possibilities for creating special behaviors. This can be done either in configuration files, with the shell environment variable **$GROFFER_OPT**, or on the command line.

The output can be generated and viewed in several different ways available for *groff*. This includes the *groff* native `X Window` viewer gxditview(1), each *Postcript*, *pdf*, or *dvi* display program, a web browser by generating *html* in *www mode*, or several *text modes* in text terminals.

Most of the options that must be named when running **groff** directly are determined automatically for **groffer**, due to the internal usage of the grog(1) program. But all parts can also be controlled manually by arguments.

Several file names can be specified on the command line arguments. They are transformed into a single document in the normal way of **groff**.

Option handling is done in `GNU` style. Options and file names can be mixed freely. The option '**--**' closes the option handling, all following arguments are treated as file names. Long options can be abbreviated in several ways.

OPTION OVERVIEW

breaking options

[**-h | --help**] [**-v | --version**]

groffer mode options

[**--auto**] [**--default**] [**--default-modes** *mode1,mode2,...*] [**--dvi**] [**--groff**] [**--html**] [**--latin1**] [**--mode** *display_mode*] [**--pdf**] [**--pdf2**] [**--ps**] [**--source**] [**--text**] [**--to-stdout**] [**--tty**] [**--utf8**] [**--viewer** *prog*] [**--www**] [**--x | --X**]

options related to groff

[**-T | --device** *device*] [**-Z | --intermediate-output | --ditroff**]

All further **groff** short options are accepted.

options for man pages

[**--apropos**] [**--apropos-data**] [**--apropos-devel**] [**--apropos-progs**] [**--man**] [**--no-man**] [**--no-special**] [**--whatis**]

long options taken over from GNU man

[**--all**] [**--ascii**] [**--ditroff**] [**--extension** *suffix*] [**--locale** *language*] [**--local-file**] [**--location | --where**] [**--manpath** *dir1:dir2:...*] [**--no-location**] [**--pager** *program*]

[**--sections** *sec1:sec2:...*] [**--systems** *sys1,sys2,...*] [**--troff-device** *device*]

Further long options of `GNU` **man** are accepted as well.

X Window Toolkit options

[**--bd | --bordercolor** *pixels*] [**--bg | --background** *color*]
[**--bw | --borderwidth** *pixels*] [**--display** *X-display*] [**--fg | --foreground** *color*]
[**--fn | --ft | --font** *font_name*] [**--geometry** *size_pos*] [**--resolution** *value*] [**--rv**]
[**--title** *string*] [**--xrm** *X-resource*]

options for development

[**--debug**] [**--debug-filenames**] [**--debug-grog**] [**--debug-keep**] [**--debug-params**]
[**--debug-tmpdir**] [**--do-nothing**] [**--print** *text*] [**-V**]

filespec arguments

The *filespec* parameters are all arguments that are neither an option nor an option argument. They usually mean a file name or a*man p age* searching scheme.

In the following, the term *section_extension* is used. It means a word that consists of a *man section* that is optionally followed by an *extension*. The name of a*man se ction* is a single character from **[1–9on]**, the *extension* is some word. The*extension* is mostly lac king.

No *filespec* parameters means standard input.

- stands for standard input (can occur several times).

filename the path name of an existing file.

man:*name*(*section_extension*)
man:*name***.***section_extension*
name(*section_extension*)
*name***.***section_extension*
section_extension name
search the man page *name* in the section with optional extension *section_extension*.

man:*name* man page in the lowest *man section* that has *name*.

name if *name* is not an existing file search for the man page *name* in the lowest man section.

OPTION DETAILS

The **groffer** program can usually be run with very few options. But for special purposes, it supports many options. These can be classified in 5 option classes.

All short options of **groffer** are compatible with the short options of groff(1). All long options of **groffer** are compatible with the long options of man(1).

Arguments for long option names can be abbreviated in several ways. First, the argument is checked whether it can be prolonged as is. Furthermore, each minus sign **-** is considered as a starting point for a new abbreviation. This leads to a set of multiple abbreviations for a single argument. For example, **--de-n-f** can be used as an abbreviation for **--debug-not-func**, but **--de-n** works as well. If the abbreviation of the argument leads to several resulting options an error is raised.

These abbreviations are only allowed in the environment variable **$GROFFER_OPT**, but not in the configuration files. In configuration, all long options must be exact.

groffer breaking Options

As soon as one of these options is found on the command line it is executed, printed to standard output, and the running **groffer** is terminated thereafter. All other arguments are ignored.

-h | --help
Print help information with a short explanation of options to standard output.

-v | --version
Print version information to standard output.

groffer Mode Options

The display mode and the viewer programs are determined by these options. If none of these mode and viewer options is specified **groffer** tries to find a suitable display mode automatically. The default modes are *mode pdf*, *mode ps*, *mode html*, *mode x*, and *mode dvi* in `X Window` with different viewers and *mode tty* with device *utf8* under **less** on a terminal; other modes are tested if the programs for the main default mode do not exist.

In `X Window`, many programs create their own window when called. **groffer** can run these viewers as an independent program in the background. As this does not work in text mode on a terminal (tty) there must be a way to know which viewers are `X Window` graphical programs. The **groffer** script has a small set of information on some viewer names. If a viewer argument of the command-line chooses an element that is kept as `X Window` program in this list it is treated as a viewer that can run in the background. All other, unknown viewer calls are not run in the background.

For each mode, you are free to choose whatever viewer you want. That need not be some graphical viewer suitable for this mode. There is a chance to view the output source; for example, the combination of the options **--mode=ps** and **--viewer=less** shows the content of the *Postscript* output, the source code, with the pager **less**.

--auto Equivalent to **--mode=auto**.

--default
Reset all configuration from previously processed command line options to the default values. This is useful to wipe out all former options of the configuration, in **$GROFFER_OPT**, and restart option processing using only the rest of the command line.

--default-modes *mode1,mode2,...*
Set the sequence of modes for *auto mode* to the comma separated list given in the argument. See**--mode** for details on modes. Displa y in the default manner; actually, this means to try the modes *x*, *ps*, and *tty* in this sequence.

--dvi Equivalent to **--mode=dvi**.

--viewer *prog*
Choose a viewer program for *dvi mode*. This can be a file name or a program to be searched in **$PATH**. Known `X Window` *dvi* viewers include **xdvi(1)** and **dvilx(1).** In each case, arguments can be provided additionally.

--groff Equivalent to **--mode=groff**.

--html Equivalent to **--mode=html**.

--viewer
Choose a web browser program for viewing in *html mode*. It can be the path name of an executable file or a program in **$PATH**. In each case, arguments can be provided additionally.

--mode *value*
Set the display mode. The following mode values are recognized:

auto Select the automatic determination of the display mode. The sequence of modes that are tried can be set with the **--default-modes** option. Useful for restoring the *default mode* when a different mode was specified before.

dvi Display formatted input in a *dvi* viewer program. By default, the formatted input is displayed with the **xdvi(1)** program.

groff After the file determination, switch **groffer** to process the input like groff(1) would do. This disables the *groffer* viewing features.

html Translate the input into html format and display the result in a web browser program. By default, the existence of a sequence of standard web browsers is tested, starting with **konqueror(1)** and **mozilla(1).** The text html viewer is **lynx(1).**

pdf Transform *roff input files*in to a *PDF file* by using the groff(1) device **-Tpdf**. This is the default **PDF** generator. The generated *PDF file* is displayed with suitable viewer programs, such as **okular(1).**

pdf2 This is the traditional *pdf mode*. Sometimes this mode produces more correct output than the default **PDF mode**. By default, the input is formatted by **groff** using the Postscript device, then it is transformed into the PDF file format using **gs(1),** or **ps2pdf(1).** If that's not possible, the *Postscript mode (ps)* is used instead. Finally it is displayed using different viewer programs.

ps Display formatted input in a Postscript viewer program. By default, the formatted input is displayed in one of many viewer programs.

text Format in a *groff text mode* and write the result to standard output without a pager or viewer program. The text device, *latin1* by default, can be chosen with option **-T**.

tty Format in a *groff text mode* and write the result to standard output using a text pager program, even when in `X Window`.

www Equivalent to **--mode=html**.

x Display the formatted input in a native *roff* viewer. By default, the formatted input is displayed with the gxditview(1) program being distributed together with **groff**. But the standard`X Window` tool **xditview(1)** can also be c hosen with the option **--viewer**. The default resolution is**75dpi**, but **100dpi** are also possible. The default *groff* device for the resolution of **75dpi** is **X75-12**, for **100dpi** it is **X100**. The corresponding*gr off intermediate output* for the actual device is generated and the result is displayed. For a resolution of **100dpi**, the default width of the geometry of the display program is chosen to **850dpi**.

X Equivalent to **--mode=x**.

The following modes do not use the *groffer* viewing features. They are only interesting for advanced applications.

groff Generate device output with plain *groff* without using the special viewing features of *groffer*. If no device was specified by option **-T** the *groff* default **ps** is assumed.

source
Output the roff source code of the input files without further processing.

--pdf Equivalent to **--mode=pdf**.

--pdf2 Equivalent to **--mode=pdf2**.

--viewer *prog*
Choose a viewer program for *pdf mode*. This can be a file name or a program to be searched in **$PATH**; arguments can be provided additionally.

--ps Equivalent to **--mode=ps**.

--viewer *prog*
Choose a viewer program for *ps mode*. This can be a file name or a program to be searched in **$PATH**. Common Postscript viewers include **okular(1), evince(1), gv(1), ghostview(1),** and **gs(1),** In each case, arguments can be provided additionally.

--source
Equivalent to **--mode=source**.

--text Equivalent to **--mode=text**.

--to-stdout
The file for the chosen mode is generated and its content is printed to standard output. It will not be displayed in graphical mode.

--tty Equivalent to **--mode=tty**.

--viewer *prog*
Choose a text pager for mode *tty*. The standard pager is less(1). This option is equivalent to *man* option **--pager=***prog*. The option argument can be a file name or a program to be searched in **$PATH**; arguments can be provided additionally.

--www
Equivalent to **--mode=html**.

--viewer
prog.

--X | --x
Equivalent to **--mode=x**.

--viewer *prog*
Choose a viewer program for *x mode*. Suitable viewer programs are gxditview(1) which is the default and **xditview(1).** The argument can be any executable file or a program in **$PATH**; arguments can be provided additionally.

-- Signals the end of option processing; all remaining arguments are interpreted as *filespec* parameters.

Besides these, **groffer** accepts all short options that are valid for the groff(1) program. All non-**groffer** options are sent unmodified via **grog** to **groff**. So postprocessors, macro packages, compatibility with *classical troff*, and much more can be manually specified.

Options related to groff

All short options of **groffer** are compatible with the short options of groff(1). The following of **groff** options have either an additional special meaning within **groffer** or make sense for normal usage.

Because of the special outputting behavior of the **groff** option **-Z groffer** was designed to be switched into *groff mode*; the *groffer* viewing features are disabled there. The other **groff** options do not switch the mode, but allow to customize the formatting process.

--a This generates an ascii approximation of output in the *text modes*. That could be important when the text pager has problems with control sequences in *tty mode*.

--m *file*
Add *file* as a *groff* macro file. This is useful in case it cannot be recognized automatically.

--P *opt_or_arg*
Send the argument *opt_or_arg* as an option or option argument to the actual **groff** postprocessor.

--T *devname* **| --device** *devname*
This option determines **groff**'s output device. The most important devices are the text output devices for referring to the different character sets, such as **ascii**, **utf8**, **latin1**, **utf8**, and others. Each of these arguments switches **groffer** into a *text mode* using this device, to *mode tty* if the actual mode is not a *text mode*. The following *devname* arguments are mapped to the corresponding **groffer --mode=***devname* option: **dvi**, **html**, and **ps**. All**X*** argumen ts are mapped to *mode x*. Each other *devname* argument switches to *mode groff* using this device.

--X is equivalent to **groff -X**. It displays the *groff intermediate output* with **gxditview**. As the quality is relatively bad this option is deprecated; use **--X** instead because the *x mode* uses an *X** device for a better display.

-Z | --intermediate-output | --ditroff
Switch into *groff mode* and format the input with the *groff intermediate output* without postprocessing; see groff_out(5). This is equivalent to option **--ditroff** of *man*, which can be used as well.

All other **groff** options are supported by **groffer**, but they are just transparently transferred to **groff** without any intervention. The options that are not explicitly handled by **groffer** are transparently passed to **groff**. Therefore these transparent options are not documented here, but in groff(1). Due to the automatism in**groffer**, none of these **groff** options should be needed, except for advanced usage.

Options for man pages

--apropos
Start the apropos(1) command or facility of man(1) for searching the *filespec* arguments within all *man page* descriptions. Each *filespec* argument is taken for search as it is; *section* specific parts are not handled, such that **7 groff** searches for the two arguments **7** and **groff**, with a large result; for the *filespec* **groff.7** nothing will be found. The *language* locale is handled only when the called programs do support this; the GNU **apropos** and **man -k** do not. The display differs from the **apropos** program by the following concepts:

- Construct a *groff* frame similar to a *man page* to the output of **apropos**,
- each *filespec* argument is searched on its own.
- The restriction by **--sections** is handled as well,
- wildcard characters are allowed and handled without a further option.

--apropos-data
Show only the **apropos** descriptions for data documents, these are the man(7) *sections 4*, *5*, and *7*. Direct*se ction* declarations are ignored, wildcards are accepted.

--apropos-devel
Show only the **apropos** descriptions for development documents, these are the man(7) *sections 2*, *3*, and *9*. Direct*se ction* declarations are ignored, wildcards are accepted.

--apropos-progs
Show only the **apropos** descriptions for documents on programs, these are the man(7) *sections 1*, *6*, and *8*. Direct*se ction* declarations are ignored, wildcards are accepted.

--whatis
For each *filespec* argument search all *man pages* and display their description — or say that it is not a *man page*. This is written from anew, so it differs from*man*'s **whatis** output by the following concepts

- each retrieved file name is added,
- local files are handled as well,
- the *language* and *system* locale is supported,
- the display is framed by a *groff* output format similar to a *man page*,
- wildcard characters are allowed without a further option.

The following options were added to **groffer** for choosing whether the file name arguments are interpreted as names for local files or as a search pattern for *man pages*. The default is looking up for local files.

--man Check the non-option command line arguments (*filespecs*) first on being *man pages*, then whether they represent an existing file. By default, a *filespec* is first tested whether it is an existing file.

--no-man | --local-file
Do not check for *man pages*. **--local-file** is the corresponding **man** option.

--no-special
Disable former calls of **--all**, **--apropos***, and **--whatis**.

Long options taken over from GNU man

The long options of **groffer** were synchronized with the long options of `GNU` **man**. All long options of `GNU` **man** are recognized, but not all of these options are important to **groffer**, so most of them are just ignored. These ignored **man** options are **--catman**, **--troff**, and **--update**.

In the following, the **man** options that have a special meaning for **groffer** are documented.

If your system has `GNU` **man** installed the full set of long and short options of the `GNU` **man** program can be passed via the environment variable **$MANOPT**; see man(1).

--all In searching *man pages*, retrieve all suitable documents instead of only one.

-7 | --ascii
In *text modes*, display ASCII translation of special characters for critical environment. This is equivalent to **groff -mtty_char**; see groff_tmac(5).

--ditroff
Produce *groff intermediate output*. This is equivalent to **groffer -Z**.

--extension *suffix*
Restrict *man page* search to file names that have *suffix* appended to their section element. For example, in the file name */usr/share/man/man3/terminfo.3ncurses.gz* the *man page* extension is *ncurses*.

--locale *language*
Set the language for *man pages*. This has the same effect, but overwrites **$LANG**.

--location
Print the location of the retrieved files to standard error.

--no-location
Do not display the location of retrieved files; this resets a former call to **--location**. This was added by **groffer**.

--manpath *'dir1:dir2:...'*
Use the specified search path for retrieving *man pages* instead of the program defaults. If the argument is set to the empty string the search for *man page* is disabled.

--pager
Set the pager program in *tty mode*; default is **less**. This can be set with**--view er**.

--sections *sec1:sec2:...*
Restrict searching for *man pages* to the given *sections*, a colon-separated list.

--systems *sys1,sys2,...*
Search for *man pages* for the given operating systems; the argument *systems* is a comma-separated list.

--where
Equivalent to **--location**.

X Window Toolkit Options

The following long options were adapted from the corresponding `X Window Toolkit` options. **groffer** will pass them to the actual viewer program if it is an `X Window` program. Otherwise these options are ignored.

Unfortunately these options use the old style of a single minus for long options. For **groffer** that was changed to the standard with using a double minus for long options, for example, **groffer** uses the option **--font** for the `X Window` option **-font**.

See **X(7)** and the documentation on the `X Window Toolkit` options for more details on these options and their arguments.

--background *color*
Set the background color of the viewer window.

--bd *pixels*
This is equivalent to **--bordercolor**.

--bg *color*
This is equivalent to **--background**.

--bw *pixels*
This is equivalent to **--borderwidth**.

--bordercolor *pixels*
Specifies the color of the border surrounding the viewer window.

--borderwidth *pixels*
Specifies the width in pixels of the border surrounding the viewer window.

--display *X-display*
Set the `X Window` display on which the viewer program shall be started, see the `X Window` documentation for the syntax of the argument.

--foreground *color*
Set the foreground color of the viewer window.

--fg *color*
This is equivalent to **--foreground**.

--fn *font_name*
This is equivalent to **--font**.

--font *font_name*
Set the font used by the viewer window. The argument is an `X Window` font name.

--ft *font_name*
This is equivalent to **--font**.

--geometry *size_pos*
Set the geometry of the display window, that means its size and its starting position. See **X(7)** for the syntax of the argument.

--resolution *value*
Set `X Window` resolution in dpi (dots per inch) in some viewer programs. The only supported dpi values are **75** and **100**. Actually, the default resolution for **groffer** is set to **75dpi**. The resolution also sets the default device in*mo de x*.

--rv Reverse foreground and background color of the viewer window.

--title *'some text'*
Set the title for the viewer window.

--xrm *'resource'*
Set `X Window` resource.

Options for Development

--debug
Enable all debugging options **--debug-***type*. The temporary files are kept and not deleted, the **grog** output is printed, the name of the temporary directory is printed, the displayed file names are printed, and the parameters are printed.

--debug-filenames
Print the names of the files and *man pages* that are displayed by **groffer**.

--debug-grog
Print the output of all **grog** commands.

--debug-keep
Enable two debugging informations. Print the name of the temporary directory and keep the temporary files, do not delete them during the run of **groffer**.

--debug-params
Print the parameters, as obtained from the configuration files, from **GROFFER_OPT**, and the command line arguments.

--debug-tmpdir
Print the name of the temporary directory.

--do-nothing
This is like **--version**, but without the output; no viewer is started. This makes only sense in development.

--print=*text*
Just print the argument to standard error. This is good for parameter check.

-V This is an advanced option for debugging only. Instead of displaying the formatted input, a lot of *groffer* specific information is printed to standard output:

- the output file name in the temporary directory,
- the display mode of the actual **groffer** run,
- the display program for viewing the output with its arguments,
- the active parameters from the config files, the arguments in **$GROFFER_OPT**, and the arguments of the command line,
- the pipeline that would be run by the **groff** program, but without executing it.

Other useful debugging options are the **groff** option **-Z** and **--mode=groff**.

Filespec Arguments
A *filespec* parameter is an argument that is not an option or option argument. In**groffer**, *filesp ec* parameters are a file name or a template for searching *man pages*. These input sources are collected and composed into a single output file such as **groff** does.

The strange `POSIX` behavior to regard all arguments behind the first non-option argument as *filespec* arguments is ignored. The `GNU` behavior to recognize options even when mixed with *filespec* arguments is used throughout. But, as usual, the double minus argument **--** ends the option handling and interprets all following arguments as *filespec* arguments; so the `POSIX` behavior can be easily adopted.

The options **--apropos*** have a special handling of *filespec* arguments. Each argument is taken as a search scheme of its own. Also a regexp (regular expression) can be used in the filespec. For example, **groffer --apropos '^gro.f$'** searches **groff** in the *man page* name, while **groffer --apropos groff** searches **groff** somewhere in the name or description of the *man pages*.

All other parts of *groffer*, such as the normal display or the output with **--whatis** have a different scheme for *filespecs*. No regular expressions are used for the arguments. The*filesp ec* arguments are handled by the following scheme.

It is necessary to know that on each system the *man pages* are sorted according to their content into several sections. The *classical man sections* have a single-character name, either a digit from **1** to **9** or one of the characters **n** or **o**.

This can optionally be followed by a string, the so-called *extension*. The*extension* allo ws to store several *man pages* with the same name in the same *section*. But the*extension* is only rarely used,

usually it is omitted. Then the *extensions* are searched automatically by alphabet.

In the following, we use the name *section_extension* for a word that consists of a single character *section* name or a *section* character that is followed by an *extension*. Each *filespec* parameter can have one of the following forms in decreasing sequence.

- No *filespec* parameters means that **groffer** waits for standard input. The minus option **-** always stands for standard input; it can occur several times. If you want to look up a *man page* called **-** use the argument **man:-**.
- Next a *filespec* is tested whether it is the path name of an existing file. Otherwise it is assumed to be a searching pattern for a *man page*.
- **man:***name*(*section_extension*), **man:***name***.***section_extension*, *name*(*section_extension*), or *name***.***section_extension* search the man page *name* in man section and possibly extension of *section_extension*.
- Now **man:***name* searches for a *man page* in the lowest *man section* that has a document called *name*.
- *section_extension name* is a pattern of 2 arguments that originates from a strange argument parsing of the **man** program. Again, this searches the man page *name* with *section_extension*, a combination of a *section* character optionally followed by an *extension*.
- We are left with the argument *name* which is not an existing file. So this searches for the *man page* called *name* in the lowest *man section* that has a document for this name.

Several file name arguments can be supplied. They are mixed by **groff** into a single document. Note that the set of option arguments must fit to all of these file arguments. So they should have at least the same style of the *groff* language.

OUTPUT MODES

By default, the **groffer** program collects all input into a single file, formats it with the **groff** program for a certain device, and then chooses a suitable viewer program. The device and viewer process in **groffer** is called a *mode*. The mode and viewer of a running **groffer** program is selected automatically, but the user can also choose it with options. The modes are selected by option the arguments of **--mode=***anymode*. Additionally, each of this argument can be specified as an option of its own, such as **anymode**. Most of these modes have a viewer program, which can be chosen by the option **--viewer**.

Several different modes are offered, graphical modes for `X Window`, *text modes*, and some direct *groff modes* for debugging and development.

By default, **groffer** first tries whether *x mode* is possible, then *ps mode*, and finally *tty mode*. This mode testing sequence for *auto mode* can be changed by specifying a comma separated list of modes with the option **--default-modes.**

The searching for *man pages* and the decompression of the input are active in every mode.

Graphical Display Modes

The graphical display modes work mostly in the `X Window` environment (or similar implementations within other windowing environments). The environment variable **$DISPLAY** and the option **--display** are used for specifying the `X Window` display to be used. If this environment variable is empty **groffer** assumes that no `X Window` is running and changes to a *text mode*. You can change this automatic behavior by the option **--default-modes**.

Known viewers for the graphical display modes and their standard `X Window` viewer programs are

- in a PDF viewer (*pdf mode*)
- in a web browser (*html* or *www mode*)
- in a Postscript viewer (*ps mode*)

- `X Window` *roff* viewers such as gxditview(1) or **xditview(1)** (in *x mode*)
- in a dvi viewer program (*dvi mode*)

The *pdf mode* has a major advantage — it is the only graphical display mode that allows to search for text within the viewer; this can be a really important feature. Unfortunately, it takes some time to transform the input into the PDF format, so it was not chosen as the major mode.

These graphical viewers can be customized by options of the `X Window Toolkit`. But the**groffer** options use a leading double minus instead of the single minus used by the `X Window Toolkit`.

Text modes

There are two modes for text output, *mode text* for plain output without a pager and *mode tty* for a text output on a text terminal using some pager program.

If the variable **$DISPLAY** is not set or empty, **groffer** assumes that it should use *tty mode*.

In the actual implementation, the *groff* output device *latin1* is chosen for *text modes*. This can be changed by specifying option **-T** or **--device**.

The pager to be used can be specified by one of the options **--pager** and **--viewer**, or by the environment variable **$PAGER**. If all of this is not used the less(1) program with the option **-r** for correctly displaying control sequences is used as the default pager.

Special Modes for Debugging and Development

These modes use the *groffer* file determination and decompression. This is combined into a single input file that is fed directly into **groff** with different strategy without the *groffer* viewing facilities. These modes are regarded as advanced, they are useful for debugging and development purposes.

The *source mode* with option **--source** just displays the decompressed input.

Option **--to-stdout** does not display in a graphical mode. It just generates the file for the chosen mode and then prints its content to standard output.

The *groff mode* passes the input to **groff** using only some suitable options provided to **groffer**. This enables the user to save the generated output into a file or pipe it into another program.

In *groff mode*, the option **-Z** disables post-processing, thus producing the *groff intermediate output*. In this mode, the input is formatted, but not postprocessed; see groff_out(5) for details.

All **groff** short options are supported by **groffer**.

MAN PAGE SEARCHING

The default behavior of **groffer** is to first test whether a file parameter represents a local file; if it is not an existing file name, it is assumed to represent the name of a *man page*. The following options can be used to determine whether the arguments should be handled as file name or *man page* arguments.

--man forces to interpret all file parameters as *filespecs* for searching *man pages*.

--no-man
--local-file
: disable the *man* searching; so only local files are displayed.

If neither a local file nor a *man page* was retrieved for some file parameter a warning is issued on standard error, but processing is continued.

Search Algorithm

Let us now assume that a *man page* should be searched. The**groffer** program pro vides a search facility for *man pages*. All long options, all environment variables, and most of the functionality of the `GNU` man(1) program were implemented. The search algorithm shall determine which file is displayed for a given *man page*. The process can be modified by options and environment variables.

The only *man* action that is omitted in **groffer** are the preformatted *man pages*, also called

cat pages. With the excellent performance of the actual computers, the preformatted *man pages* aren't necessary any longer. Additionally, **groffer** is a *roff* program; it wants to read *roff* source files and format them itself.

The algorithm for retrieving the file for a *man page* needs first a set of directories. This set starts with the so-called *man path* that is modified later on by adding names of *operating system* and *language*. This arising set is used for adding the section directories which contain the *man page* files.

The *man path* is a list of directories that are separated by colon. It is generated by the following methods.

- The environment variable **$MANPATH** can be set.
- It can be read from the arguments of the environment variable **$MANOPT**.
- The *man path* can be manually specified by using the option **--manpath**. An empty argument disables the *man page* searching.
- When no *man path* was set the manpath(1) program is tried to determine one.
- If this does not work a reasonable default path from **$PATH** is determined.

We now have a starting set of directories. The first way to change this set is by adding names of *operating systems*. This assumes that*man p ages* for several *operating systems* are installed. This is not always true. The names of such *operating systems* can be provided by 3 methods.

- The environment variable **$SYSTEM** has the lowest precedence.
- This can be overridden by an option in **$MANOPT**.
- This again is overridden by the command line option **--systems**.

Several names of *operating systems* can be given by appending their names, separated by a comma.

The *man path* is changed by appending each *system* name as subdirectory at the end of each directory of the set. No directory of the *man path* set is kept. But if no*system* name is specified the *man path* is left unchanged.

After this, the actual set of directories can be changed by *language* information. This assumes that there exist *man pages* in different languages. The wanted *language* can be chosen by several methods.

- Environment variable **$LANG**.
- This is overridden by **$LC_MESSAGES**.
- This is overridden by **$LC_ALL**.
- This can be overridden by providing an option in **$MANOPT**.
- All these environment variables are overridden by the command line option **--locale**.

The *default language* can be specified by specifying one of the pseudo-language parameters `C` or `POSIX`. This is like deleting a formerly given *language* information. The *man pages* in the *default language* are usually in English.

Of course, the *language* name is determined by **man**. In`GNU` **man**, it is specified in the `POSIX 1003.1` based format:

<*language*>[_<*territory*>[**.**<*character-set*>[**,**<*version*>]]],

but the two-letter code in <*language*> is sufficient for most purposes. If for a complicated *language* formulation no *man pages* are found **groffer** searches the country part consisting of these first two characters as well.

The actual directory set is copied thrice. The *language* name is appended as subdirectory to each directory in the first copy of the actual directory set (this is only done when a language informa-

tion is given). Then the 2-letter abbreviation of the*language* name is appended as subdirectories to the second copy of the directory set (this is only done when the given language name has more than 2 letters). The third copy of the directory set is kept unchanged (if no *language* information is given this is the kept directory set). These maximally 3 copies are appended to get the new directory set.

We now have a complete set of directories to work with. In each of these directories, the *man* files are separated in *sections*. The name of a*se ction* is represented by a single character, a digit between *1* and *9*, or the character *o* or *n*, in this order.

For each available *section*, a subdirectory **man**<*section*> exists containing all *man* files for this *section*, where <*section*> is a single character as described before. Each *man* file in a *section* directory has the form **man**<*section*>**/**<*name*>**.**<*section*>*[*<*extension*>*][***.**<*compression*>*]*, where <*extension*> and <*compression*> are optional. <*name*> is the name of the *man page* that is also specified as filespec argument on the command line.

The *extension* is an addition to the section. This postfix acts like a subsection. An *extension* occurs only in the file name, not in name of the *section* subdirectory. It can be specified on the command line.

On the other hand, the *compression* is just an information on how the file is compressed. This is not important for the user, such that it cannot be specified on the command line.

There are 4 methods to specify a *section* on the command line:

- Environment variable **$MANSECT**
- Command line option **--sections**
- Appendix to the *name* argument in the form <*name*>.<*section*>
- Preargument before the *name* argument in the form <*section*> <*name*>

It is also possible to specify several *sections* by appending the single characters separated by colons. One can imagine that this means to restrict the*man p age* search to only some *sections*. The multiple *sections* are only possible for **$MANSECT** and**--sections**.

If no *section* is specified all *sections* are searched one after the other in the given order, starting with *section 1*, until a suitable file is found.

There are 4 methods to specify an *extension* on the command line. But it is not necessary to provide the whole extension name, some abbreviation is good enough in most cases.

- Environment variable **$EXTENSION**
- Command line option **--extension**
- Appendix to the <*name*>.<*section*> argument in the form <*name*>.<*section*><*extension*>
- Preargument before the *name* argument in the form <*section*><*extension*> <*name*>

For further details on *man page* searching, see man(1).

Examples of man files

/usr/share/man/man1/groff.1

This is an uncompressed file for the *man page* `groff` in *section 1*. It can be called by

sh# `groffer groff`

No *section* is specified here, so all *sections* should be searched, but as *section 1* is searched first this file will be found first. The file name is composed of the following components. **/usr/share/man/** must be part of the *man path*; the subdirectory **man1/** and the part **.1** stand for the *section*; **groff** is the name of the *man page*.

/usr/local/share/man/man7/groff.7.gz

The file name is composed of the following components. **/usr/local/share/man** must be part of the *man path*; the subdirectory **man7/** and the part **.7** stand for the *section*; **groff** is the name of the *man page*; the final part **.gz** stands for a compression with

gzip(1). As these *ction* is not the first one it must be specified as well. This can be done by one of the following commands.

```
sh# groffer groff.7
sh# groffer 7 groff
sh# groffer --sections=7 groff
```

/usr/local/man/man1/ctags.1emacs21.bz2

Here **/usr/local/man** must be in *man path*; the subdirectory **man1/** and the file name part **.1** stand for *section 1*; the name of the *man page* is **ctags**; the section has an extension **emacs21**; and the file is compressed as **.bz2** with bzip2(1). The file can be viewed with one of the following commands

```
sh# groffer ctags.1e
sh# groffer 1e ctags
sh# groffer --extension=e --sections=1 ctags
```

where `e` works as an abbreviation for the extension `emacs21`.

/usr/man/linux/de/man7/man.7.Z

The directory **/usr/man** is now part of the *man path*; then there is a subdirectory for an *operating system* name **linux/**; next comes a subdirectory **de/** for the German *language*; the *section* names **man7** and **.7** are known so far; **man** is the name of the *man page*; and **.Z** signifies the compression that can be handled by gzip(1). We want now show how to provide several values for some options. That is possible for *sections* and *operating system* names. So we use as *sections 5* and *7* and as *system* names *linux* and *aix*. The command is then

```
sh# groffer --locale=de --sections=5:7 --systems=linux,aix man
sh# LANG=de MANSECT=5:7 SYSTEM=linux,aix groffer man
```

DECOMPRESSION

The program has a decompression facility. If standard input or a file that was retrieved from the command line parameters is compressed with a format that is supported by either gzip(1) or bzip2(1) it is decompressed on-the-fly. This includes the`GNU` **.gz**, **.bz2**, and the traditional **.Z** compression. The program displays the concatenation of all decompressed input in the sequence that was specified on the command line.

ENVIRONMENT

The **groffer** program supports many system variables, most of them by courtesy of other programs. All environment variables of groff(1) and `GNU` man(1) and some standard system variables are honored.

Native groffer Variables

$GROFFER_OPT

Store options for a run of **groffer**. The options specified in this variable are overridden by the options given on the command line. The content of this variable is run through the shell builtin 'eval'; so arguments containing white-space or special shell characters should be quoted. Do not forget to export this variable, otherwise it does not exist during the run of **groffer**.

System Variables

The following variables have a special meaning for **groffer**.

$DISPLAY

If this variable is set this indicates that the `X Window` system is running. Testing this variable decides on whether graphical or text output is generated. This variable should not be changed by the user carelessly, but it can be used to start the graphical **groffer** on a remote `X Window` terminal. For example, depending on your system, **groffer** can be started on the second monitor by the command

```
sh# DISPLAY=:0.1 groffer what.ever &
```

$LC_ALL
$LC_MESSAGES
$LANG
If one of these variables is set (in the above sequence), its content is interpreted as the locale, the language to be used, especially when retrieving *man pages*. A locale name is typically of the form *language*[_*territory*[.*codeset*[@*modifier*]]], where *language* is an ISO 639 language code, *territory* is an ISO 3166 country code, and *codeset* is a character set or encoding identifier like ISO-8859-1 or UTF-8; see setlocale(3). The locale values `C` and `POSIX` stand for the default, i.e. the *man page* directories without a language prefix. This is the same behavior as when all 3 variables are unset.

$PAGER
This variable can be used to set the pager for the tty output. For example, to disable the use of a pager completely set this variable to the cat(1) program

```
sh# PAGER=cat groffer anything
```

$PATH
All programs within the **groffer** script are called without a fixed path. Thus this environment variable determines the set of programs used within the run of **groffer**.

Groff Variables

The **groffer** program internally calls **groff**, so all environment variables documented in groff(1) are internally used within **groffer** as well. The following variable has a direct meaning for the **groffer** program.

$GROFF_TMPDIR
If the value of this variable is an existing, writable directory, **groffer** uses it for storing its temporary files, just as **groff** does. See the groff(1) man page for more details on the location of temporary files.

Man Variables

Parts of the functionality of the **man** program were implemented in **groffer**; support for all environment variables documented in man(1) was added to **groffer**, but the meaning was slightly modified due to the different approach in **groffer**; but the user interface is the same. The **man** environment variables can be overwritten by options provided with **$MANOPT**, which in turn is overwritten by the command line.

$EXTENSION
Restrict the search for *man pages* to files having this extension. This is overridden by option **--extension**; see there for details.

$MANOPT
This variable contains options as a preset for man(1). As not all of these are relevant for **groffer** only the essential parts of its value are extracted. The options specified in this variable overwrite the values of the other environment variables that are specific to *man*. All options specified in this variable are overridden by the options given on the command line.

$MANPATH
If set, this variable contains the directories in which the *man page* trees are stored. This is overridden by option **--manpath**.

$MANSECT
If this is a colon separated list of section names, the search for *man pages* is restricted to those manual sections in that order. This is overridden by option **--sections**.

$SYSTEM
If this is set to a comma separated list of names these are interpreted as *man page* trees for different operating systems. This variable can be overwritten by option **--systems**;

see there for details.

The environment variable **$MANROFFSEQ** is ignored by **groffer** because the necessary preprocessors are determined automatically.

CONFIGURATION FILES

The **groffer** program can be preconfigured by two configuration files.

/etc/groff/groffer.conf
: System-wide configuration file for **groffer**.

$HOME/.groff/groffer.conf
: User-specific configuration file for **groffer**, where **$HOME** denotes the user's home directory. This file is called after the system-wide configuration file to enable overriding by the user.

Both files are handled for the configuration, but the configuration file in **/etc** comes first; it is overwritten by the configuration file in the home directory; both configuration files are overwritten by the environment variable **$GROFFER_OPT**; everything is overwritten by the command line arguments.

The configuration files contain options that should be called as default for every **groffer** run. These options are written in lines such that each contains either a long option, a short option, or a short option cluster; each with or without an argument. So each line with configuration information starts with a minus character '**-**'; a line with a long option starts with two minus characters '**--**', a line with a short option or short option cluster starts with a single minus '**-**'.

The option names in the configuration files may not be abbreviated, they must be exact.

The argument for a long option can be separated from the option name either by an equal sign '=' or by whitespace, i.e. one or several space or tab characters. An argument for a short option or short option cluster can be directly appended to the option name or separated by whitespace. The end of an argument is the end of the line. It is not allowed to use a shell environment variable in an option name or argument.

It is not necessary to use quotes in an option or argument, except for empty arguments. An empty argument can be provided by appending a pair of quotes to the separating equal sign or whitespace; with a short option, the separator can be omitted as well. For a long option with a separating equal sign '=', the pair of quotes can be omitted, thus ending the line with the separating equal sign. All other quote characters are cancelled internally.

In the configuration files, arbitrary whitespace is allowed at the beginning of each line, it is just ignored. Each whitespace within a line is replaced by a single space character ' ' internally.

All lines of the configuration lines that do not start with a minus character are ignored, such that comments starting with '#' are possible. So there are no shell commands in the configuration files.

As an example, consider the following configuration file that can be used either in **/etc/groff/groffer.conf** or **~/.groff/groffer.conf .**

```
# groffer configuration file
#
# groffer options that are used in each call of groffer
--foreground=DarkBlue
--resolution=100
--viewer=gxditview -geometry 900x1200
--viewer xpdf -Z 150
```

The lines starting with # are just ignored, so they act as command lines. This configuration sets four **groffer** options (the lines starting with '**-**'). This has the following effects:

- Use a text color of **DarkBlue** in all viewers that support this, such as **gxditview**.
- Use a resolution of **100dpi** in all viewers that support this, such as **gxditview**. By this, the default device in *x mode* is set to **X100**.
- Force gxditview(1) as the *x-mode* viewer using the geometry option for setting the width to **900px** and the height to **1200px**. This geometry is suitable for a resolution of**100dpi**.
- Use **xpdf(1)** as the *pdf-mode* viewer with the argument **-Z 150**.

EXAMPLES

The usage of **groffer** is very easy. Usually, it is just called with a file name or *man page*. The following examples, however, show that **groffer** has much more fancy capabilities.

```
sh# groffer /usr/local/share/doc/groff/meintro.ms.gz
```

Decompress, format and display the compressed file **meintro.ms.gz** in the directory **/usr/local/share/doc/groff**, using the standard viewer **gxditview** as graphical viewer when in `X Window`, or the less(1) pager program when not in `X Window`.

```
sh# groffer groff
```

If the file **./groff** exists use it as input. Otherwise interpret the argument as a search for the *man page* named **groff** in the smallest possible *man section*, being section 1 in this case.

```
sh# groffer man:groff
```

search for the *man page* of **groff** even when the file **./groff** exists.

```
sh# groffer groff.7
sh# groffer 7 groff
```

search the *man page* of **groff** in *man section* **7**. This section search works only for a digit or a single character from a small set.

```
sh# groffer fb.modes
```

If the file **./fb.modes** does not exist interpret this as a search for the *man page* of **fb.modes**. As the extension *modes* is not a single character in classical section style the argument is not split to a search for **fb**.

```
sh# groffer groff 'troff(1)' man:roff
```

The arguments that are not existing files are looked-up as the following *man pages*: **groff** (automatic search, should be found in *man* section 1), **troff** (in section 1), and **roff** (in the section with the lowest number, being 7 in this case). The quotes around *'troff(1)'* are necessary because the parentheses are special shell characters; escaping them with a backslash character *(* and *)* would be possible, too. The formatted files are concatenated and displayed in one piece.

```
sh# LANG=de groffer --man --viewer=galeon ls
```

Retrieve the German *man page* (language *de*) for the **ls** program, decompress it, format it to *html* format (*www mode*) and view the result in the web browser **galeon**. The option**--man** guarantees that the *man page* is retrieved, even when a local file **ls** exists in the actual directory.

```
sh# groffer --source 'man:roff(7)'
```

Get the *man page* called *roff* in *man* section 7, decompress it, and print its unformatted content, its source code.

```
sh# groffer --de-p --in --ap
```

This is a set of abbreviated arguments, it is determined as

```
sh# groffer --debug-params --intermediate-output --apropos
```

```
sh# cat file.gz | groffer -Z -mfoo
```

The file **file.gz** is sent to standard input, this is decompressed, and then this is transported to the *groff intermediate output mode* without post-processing (**groff** option **-Z**), using macro package *foo* (**groff** option **-m**).

```
sh# echo '\f[CB]WOW!' |
> groffer --x --bg red --fg yellow --geometry 200x100 -
```

Display the word **WOW!** in a small window in constant-width bold font, using color yellow on red background.

COMPATIBILITY

The **groffer** program is written in Perl, the Perl version during writing was v5.8.8.

groffer provides its own parser for command line arguments that is compatible to both `POSIX` **getopts(1)** and `GNU` getopt(1). It can handle option arguments and file names containing white space and a large set of special characters. The following standard types of options are supported.

- The option consisting of a single minus **-** refers to standard input.
- A single minus followed by characters refers to a single character option or a combination thereof; for example, the **groffer** short option combination **-Qmfoo** is equivalent to **-Q -m foo**.
- Long options are options with names longer than one character; they are always preceded by a double minus. An option argument can either go to the next command line argument or be appended with an equal sign to the argument; for example, **--long=arg** is equivalent to **--long arg**.
- An argument of **--** ends option parsing; all further command line arguments are interpreted as *filespec* parameters, i.e. file names or constructs for searching *man pages*).
- All command line arguments that are neither options nor option arguments are interpreted as *filespec* parameters and stored until option parsing has finished. For example, the command line

  ```
  sh# groffer file1 -a -o arg file2
  ```

  is equivalent to

  ```
  sh# groffer -a -o arg -- file1 file2
  ```

The free mixing of options and *filespec* parameters follows the GNU principle. That does not fulfill the strange option behavior of `POSIX` that ends option processing as soon as the first non-option argument has been reached. The end of option processing can be forced by the option '**--**' anyway.

BUGS

Report bugs to the bug-groff mailing list. Include a complete, self-contained example that will allow the bug to be reproduced, and say which version of **groffer** you are using.

You can also use the groff mailing list, but you must first subscribe to this list. You can do that by visiting the groff mailing list web page.

See groff(1) for information on availability.

SEE ALSO

groff(1), troff(1)

Details on the options and environment variables available in **groff**; all of them can be used with **groffer**.

grog(1) This program tries to guess the necessary **groff** command line options from the input and the **groffer** options.

groff(7) Documentation of the *groff* language.

groff_char(7) Documentation on the *groff* characters, special characters, and glyphs..

groff_tmac(5) Documentation on the *groff* macro files.

groff_out(5) Documentation on the *groff intermediate output* before the run of a *postprocessor*. (*ditroff* output). This can be run by the **groff** or **groffer** option **-Z**.

man(1) The standard program to display *man pages*. The information there is only useful if it is the *man page* for GNU **man**. Then it documents the options and environment variables that are supported by **groffer**.

gxditview(1)
xditview(1x)
Viewers for **groffer**'s *x mode*.

kpdf(1)
kghostview(1)
evince(1)
ggv(1)
gv(1)
ghostview(1)
gs(1) Viewers for **groffer**'s *ps mode*.

kpdf(1)
acroread(1)
evince(1)
xpdf(1)
gpdf(1)
kghostview(1)
ggv(1)
Viewers for **groffer**'s *pdf mode*.

kdvi(1), xdvi(1), dvilx(1)
Viewers for **groffer**'s *dvi mode*.

konqueror(1)
epiphany(1)
firefox(1)
mozilla(1)
netscape(1)
lynx(1)
Web-browsers for **groffer**'s *html* or *www mode*.

less(1)
more(1) Standard pager program for the *tty mode*.

gzip(1)
bzip2(1)
xz(1) The decompression programs supported by **groffer**.

COPYING

AUTHORS

This file was written by Bernd Warken.