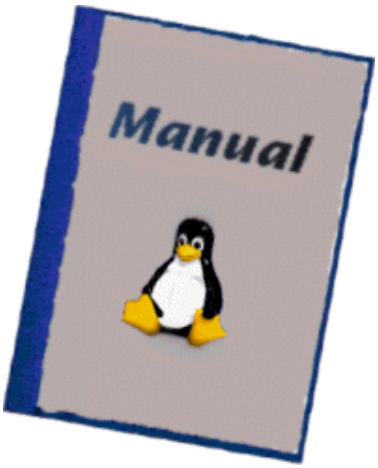
LinuxFocus article number 309
http://linuxfocus.org

Writing man-pages

by Guido Socher (homepage)

About the author:

Guido likes Linux because it is a very flexible and offers much more possibilities than any other operating system.



Abstract:

Every good program that can be used from the UNIX shell should be documented in its own man-page. This tutorial will give you a quick introduction into writing manual pages.

Introduction

Documentation is often more important than the software itself especially if the software should not only be used by the author. Even if I write a program which I do not intend to publish I write documentation for it because a couple of month later I might have forgotten how to use the program and good structured documentation will tell me within seconds how to use the program.

Traditional Linux command line utilities have always been documented in man-pages. A simple `man` *`commandname`* will tell you how to use the command.

The advantage of man-pages over other forms of documentation is that

1. They can be viewed within seconds in any Linux terminal
2. They can easily be converted to other formats: HTML, PDF, Postscript, Text,...
3. Man pages can not only be viewed in terminal windows but also other programs like konqueror (simply type: man:commandname)

The sections

Man-pages are structured in sections. Just like a book which is structured in in chapters. There are e.g two man-pages on printf. One for the C-library function (section 3) and the other for the shell command printf (section 1):

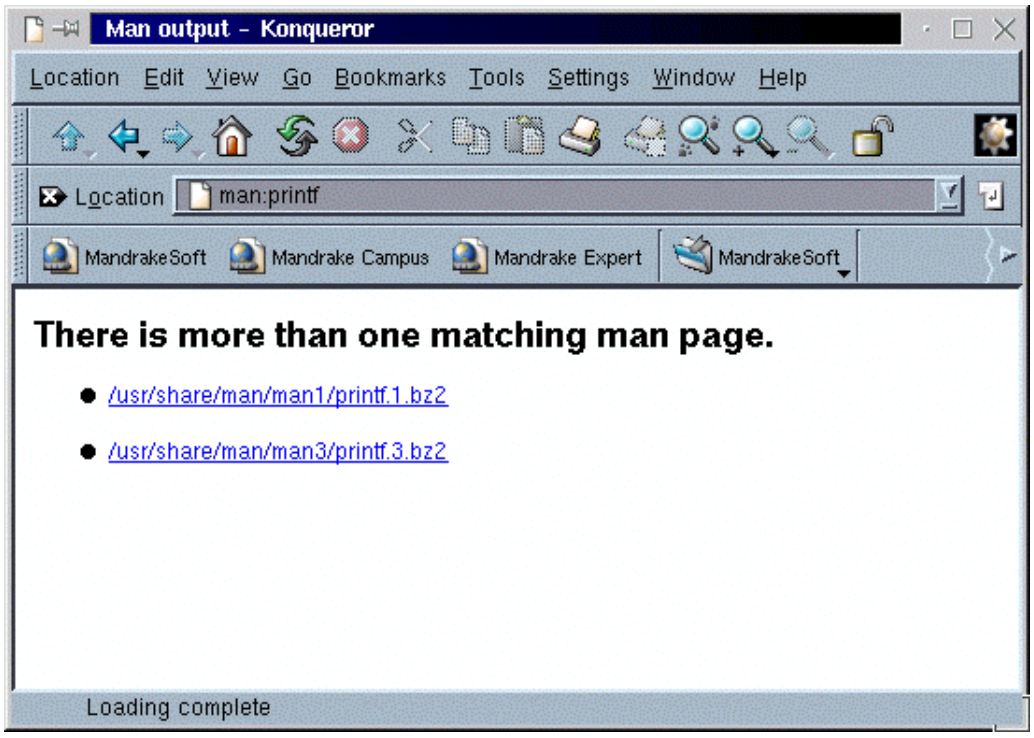
```
> whichman -0 printf
/usr/share/man/man1/printf.1.bz2
/usr/share/man/man3/printf.3.bz2
```

The different sections are:

```
Section
   1    User commands
   2    System calls, that is, functions provided
        by the kernel.
   3    Subroutines, that is, library functions.
   4    Devices, that is, special files in the /dev directory.
   5    File format descriptions, e.g. /etc/passwd.
   6    Games, self-explanatory.
   7    Miscellaneous, e.g. macro packages, conventions.
   8    System administration tools that only root can execute.
   9    Another
   n    New documentation, that may be moved to a more
        appropriate section.
   l    Local documentation referring to this particular system.
```

Therefore typing "man 1 printf" will give you the documentation about the shell command printf and "man 3 printf" will display the description of the C-library function. Running just "man printf" will print the page that is first found (usually printf from section 1).

To check if there are multiple versions of man pages you can either use the whichman command as shown above (download from my homepage or you just enter "man:printf" in konqueror and it will tell you:



MANPATH

The man command searches the man pages based on the value of the environment variable MANPATH. Unfortunately many Linux distributions set this path incorrectly. Often /usr/lib/perl5/man which contains a rich set of documentation on all perl functions is not included. You can add it to your MANPATH (in .bashrc or .tcshrc or ...) like this:

```
Bash:
  MANPATH="/usr/local/man:/usr/man:/usr/share/man:/usr/X11R6/man:/usr/lib/perl5/man"
  export MANPATH

Tcsh:
  setenv MANPATH "/usr/local/man:/usr/man:/usr/share/man:/usr/X11R6/man:/usr/lib/per
```

After setting the man path you can try "man Pod::Man" to see if you get one of the pages from perl.

The formatting keywords

To write a man page is very simple. It is a simple make-up language where keywords for the markup language start with a dot at the beginning of the line. These keywords are also called macros. The most important macros are:

```
.TH -> This starts the title/header of the man page
.SH -> Section heading
.PP -> New paragraph
."  -> A comment line
```

```
.TP -> Indent the text that comes 2 lines after this macro
```

The syntax of .TH is:
.TH [name of program] [section number] [center footer] [left footer] [center header]

The syntax of .SH is:
.SH text for a heading

The syntax of .PP is very straight forward. It just causes a line break.

I find it sometimes useful to include pre-formatted text for program code examples. This can be done with:

```
.nf
_your_pre_fromatted_
_text_goes_here_____
.fi
```

Note that these are groff/nroff macros and do as such not belong to the special man-page macros. They seem however to work fine on all Unix systems.

All man-page formatter macros are documented in the man-page called groff_man(7) (Click here to view a html version of the groff_man(7) page). I will not explain the macros here instead suggest to read the groff_man page. The groff_man page is very detailed and contains all you need to know.

The chapters

Before we start to write our own page you should know that man pages are normally structured in chapters. By convention the possible chapter headings are as follows:

```
NAME            Name section, the name of the function or command.
SYNOPSIS        Usage.
DESCRIPTION     General description
OPTIONS         Should include options and parameters.
RETURN VALUES   Sections two and three function calls.
ENVIRONMENT     Describe environment variables.
FILES           Files associated with the subject.
EXAMPLES        Examples and suggestions.
DIAGNOSTICS     Normally used for section 4 device interface
                diagnostics.
ERRORS          Sections two and three error and signal handling.
SEE ALSO        Cross references and citations.
STANDARDS       Conformance to standards if applicable.
BUGS            Gotchas and caveats.
SECURITY CONSIDERATIONS
                Security issues to be aware of.
other           Customized headers may be added at the authors
                discretion.
```

A sample man-page

Here is a small sample man-page. Note that the \- is required to make the dash distinct from hyphens. Type all that into your text editor, and save it as cdspeed.1.

```
.TH cdspeed 1  "September 10, 2003" "version 0.3" "USER COMMANDS"
.SH NAME
cdspeed \- decrease the speed of you cdrom to get faster access time
.SH SYNOPSIS
.B cdspeed
[\-h] [\-d device] \-s speed
.SH DESCRIPTION
Modern cdrom drives are too fast. It can take several seconds
on a 60x speed cdrom drive to spin it up and read data from
the drive.  The result is that these drives are just a lot slower
than a 8x or 24x drive.  This is especially true if you are only
occasionally (e.g every 5 seconds) reading a small file. This
utility limits the speed and makes the drive more responsive
when accessing small files.
.PP
cdspeed makes the drive also less noisy and is very useful if
you want to listen to music on your computer.
.SH OPTIONS
.TP
\-h
display a short help text
.TP
\-d
use the given device instead of /dev/cdrom
.TP
\-s
set the speed. The argument is a integer. Zero means restore maximum
speed.
.SH EXAMPLES
.TP
Set the maximum speed to 8 speed cdrom:
.B cdspeed
\-s 8
.PP
.TP
Restore maximum speed:
.B cdspeed
\-s 0
.PP
.SH EXIT STATUS
cdspeed returns a zero exist status if it succeeds to change to set the
maximum speed of the cdrom drive. Non zero is returned in case of failure.
.SH AUTHOR
Guido Socher (guido (at) linuxfocus.org)
.SH SEE ALSO
eject(1)
```

Click here (cdspeed.html) to view the above page.

Viewing and formatting your man-page

While writing you man-page you should from time to time view it to see that it looks right. Type:

```
nroff -man your_manpagefile.1 | less
```

or

```
groff -man -Tascii your_manpagefile.1 | less
```

To convert a man page to plain pre-formatted text (e.g for spell checking) use:

```
nroff -man your_manpagefile.1 | col -b > xxxx.txt
```

To convert it to Postscript (for printing or further conversion to pdf) use:

```
groff -man -Tps your_manpagefile.1 > your_manpagefile.ps
```

To convert the man page to html use:

```
man2html your_manpagefile.1
```

Using perl POD to generate man-pages

I know that many people feel that it is strange to just edit a man-page in a text editor. They want to generate the man page. The perl POD documentation format is a good choice. You can write the page in POD syntax and then run the command

```
pod2man your_manpagefile.pod > your_manpagefile.1
```

The syntax of the perl pod documentation language is described in a man page called perlpod. The above man page example would look in pod format as shown below. Note that POD is sensitive to space and the empty lines around the "=head" are also needed.

```
=head1 NAME

cdspeed -  decrease  the speed of you cdrom to get faster access time

=head1 SYNOPSIS

cdspeed [-h] [-d device] -s speed

=head1 DESCRIPTION

Modern cdrom drives are too fast. It can take several seconds  on a  60x
speed cdrom drive to spin it up and read data from the drive.  The result
is that these drives  are just  a lot slower than a 8x or 24x drive.
This is especially true if you are only occasionally (e.g every 5 seconds)
reading a small file. This utility limits the speed and makes the  drive
more  responsive  when  accessing  small files.

cdspeed makes the drive also less noisy and is very useful
if you want to listen to music on your computer.
```

```
=head1 OPTIONS

B<-h> display a short help text

B<-d> use the given device instead of /dev/cdrom

B<-s> set the speed. The  argument  is a  integer.  Zero
means restore maximum speed.

=head1 EXAMPLES

Set the maximum speed to 8 speed cdrom:

          cdspeed -s 8

Restore maximum speed:

          cdspeed -s 0


=head1 EXIT STATUS

cdspeed returns  a  zero  exist  status  if it succeeds to
change to set the maximum speed of the  cdrom  drive.  Non
zero is returned in case of failure.

=head1 AUTHOR

Guido Socher

=head1 SEE ALSO

eject(1)
```

References

- Man-page HOWTO
- groff_man(7), man-page macros