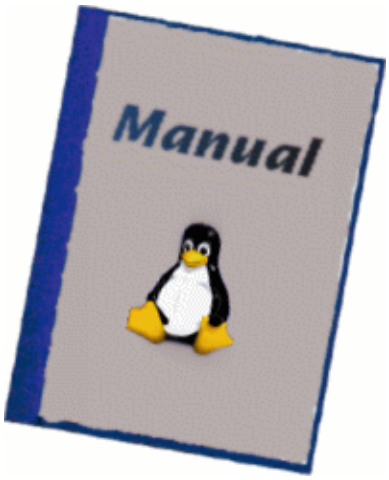
Man−pages schreiben

by Guido Socher (homepage)

About the author:

Guido mag Linux, weil es sehr flexibel ist und viel mehr bietet, als jedes andere Betriebssystem.



Abstract:

Jedes gute Programm, das man von der Kommandozeile starten kann, sollte in einer eigenen man−Seite dokumentiert sein. Dieses Tutorial erklärt, wie man so eine man−Seite schreibt.

Einleitung

Dokumentation ist oft viel wichtiger als das Programm selbst. Das ist besonders dann der Fall, wenn die Software nicht nur vom Autor selbst benutzt werden soll. Selbst, wenn ich ein Programm nur für mich schreibe, dokumentiere ich es. Die Erfahrung hat gezeigt, dass man oft schon einige Monate später vergessen hat, wie das Programm zu benutzen ist. Mit Hilfe von gut strukturierter Dokumentation weiß man dann innerhalb von Sekunden wieder, wie das Programm zu benutzen ist.

Traditonalle Linux Kommandozeilenprogramme waren schon immer in man−Seiten dokumentiert. Ein einfaches
`man name_des_Befehls` erklärt, wie der Befehl zu benutzen ist.

Die Vorteile von man−Seiten gegenüber anderer Dokumentation sind:

1. Man kann sie innerhalb von Sekunden auf jedem Linux Terminal ansehen.
2. Sie sind leicht in andere Formate zu konvertieren: HTML, PDF, Postscript, Text,...
3. Man−Seiten kann man nicht nur in einem Terminalfenster lesen, sondern auch in anderen Programmen wie z.B. konqueror (einfach man:name_des_Befehls tippen)

Die "sections"

Man−Seiten sind in sogenannten "sections" organisiert. Es gibt z.B zwei man−Seiten zu dem Befehl printf. Die eine Seite ist für den printf der C−Library−Funktion (section 3) und die andere für den Kommandozeilenbefehl printf (section 1):

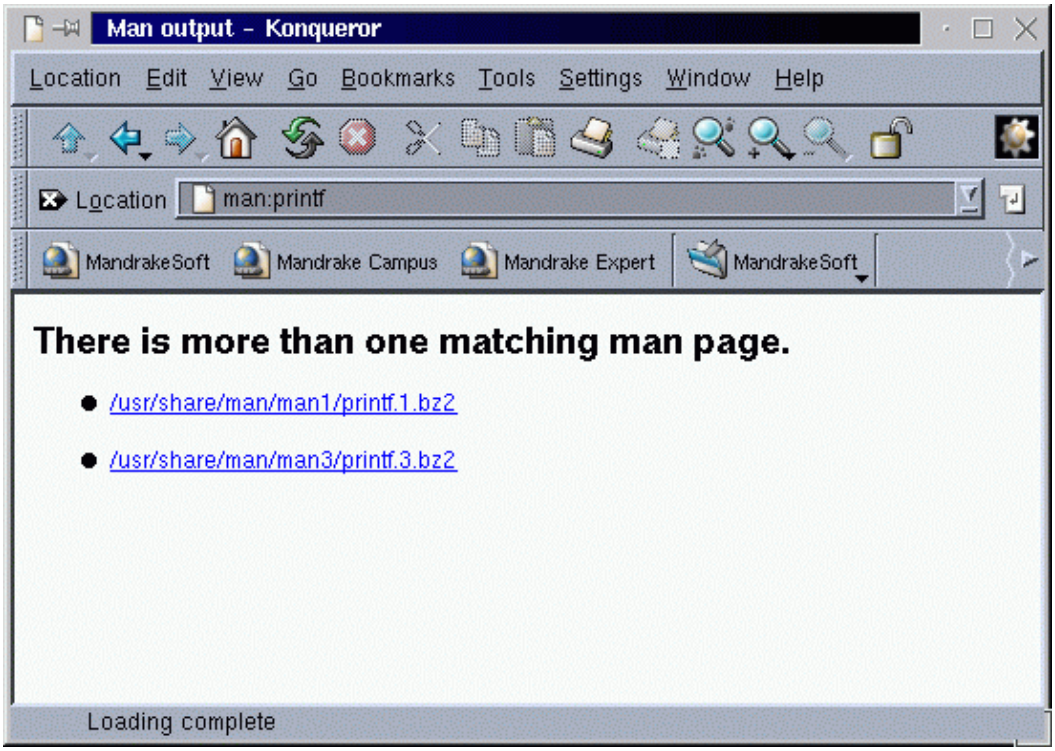
```
> whichman -0 printf
/usr/share/man/man1/printf.1.bz2
/usr/share/man/man3/printf.3.bz2
```

Die verschiedenen Sections sind:

```
Section
   1    User commands (Befehle für den Benutzer)
   2    System calls, that is, functions provided
        by the kernel. (Systembefehle, Funktionen die
        durch den Kernel zur Verfügung gestellt werden)
   3    Subroutines, that is, library functions.
        (Subroutinen, Libraryfunktionen)
   4    Devices, that is, special files in the /dev directory.
        (Devices in speziellen Dateien unter /dev)
   5    File format descriptions, e.g. /etc/passwd.
        (Beschreibung von Dateiformaten, z.B /etc/passwd)
   6    Games, self-explanatory.
        (Spiele)
   7    Miscellaneous, e.g. Makro packages, conventions.
        (Verschiedenes)
   8    System administration tools that only root can execute.
        (Systemadministration, Befehle, die nur root benutzen kann)
   9    Other  (anderes)
   n    New documentation, that may be moved to a more
        appropriate section.
        (neue Dokumentation, sie wird später ggf. in andere Sections
        verschoben)
   l    Local documentation referring to this particular system.
        (lokale, für dieses System spezifische Dokumentation)
```

Wenn man also "man 1 printf" tippt, bekommt man die Dokumentation für den Shellbefehl printf und mit "man 3 printf" bekommt man die Beschreibung der C−Library−Funktion. Tippt man nur "man printf", so erhält man die Seite, die zuerst gefunden wird (normalerweise printf aus section 1).

Um zu prüfen, ob es mehrere Seite gibt, kann man entweder den whichman Befehl bnutzen (herunterladbar von meiner Homepage), oder man tippt einfach "man:printf" in konqueror und konqueror wird eine Auswahl anzeigen:



MANPATH

Der man Befehl sucht die man−Seiten basierend auf dem Wert der Variablen MANPATH. Leider setzen viele Linux Distributionen diese Variable falsch. Oft ist /usr/lib/perl5/man, wo sich alle Perlfunktionen finden, nicht im MANPATH. Man kann dieses Verzeichnis wie folgt zu seinem MANPATH (in .bashrc oder .tcshrc oder ...) hinzufügen:

```
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/perl5/man"
```

Nachdem der MANPATH gesetzt ist, kann man "man Pod::Man" probieren, um zu sehen, ob man die man−Seiten von Perl bekommt.

Formatierung des Textes

Eine man−Seite zu schreiben, ist sehr einfach. In man−Seiten wird eine einfache "markup" Sprache benutzt. Es werden also Schlüsselworte in den Text eingebaut, die festlegen, wie der Text formatiert werden soll. Diese Schlüsselworte fangen mit einem Punkt am Anfang der Zeile an. Diese Schlüsselworte werden im Fall von man−Seiten auch als Makros bezeichnet. Die wichtigesten Makros sind:

```
.TH -> Druckt Titel und Kopfzeile der man-Seite
.SH -> Überschrift des Abschnittes
.PP -> Neuer Paragraph
."  -> Eine Kommentarzeile
.TP -> Einrücken des Textes, der zwei Zeilen nach diesem Makro erscheint
```

Die Syntax von .TH ist:
.TH [name of program] [section number] [center footer] [left footer] [center header]

Die Syntax von .SH ist:
.SH Überschrift

Die Syntax von .PP ist offensichtlich. Die Zeile wird einfach umgebrochen.

Ich finde es manchmal nützlich, vorformatierten Text für Programmcode zu benutzen. Das kann man mit folgenden Makros machen:

```
.nf
_vorformatierter
_Text_hier
.fi
```

Beachte, dass dies groff/nroff Makros sind, die als solches nicht zu den speziellen man−Seiten Makros gehören. Sie scheinen aber problemlos auf allen Unixsystemen zu funktionieren.

Alle Schlüsselwörter für man−Seiten sind in der groff_man(7) man−Seite dokumentiert (Hier klicken, um die HTML−version der groff_man(7) man−Seite zu lesen). Ich werde diese Makros hier nicht weiter erklären, da die groff_man(7) Seite schon sehr gut ist und alles enthält, was man wissen muß.

Die Kapitel

Bevor man anfängt, eine eigene man−Seite zu schreiben, sollte man wissen, dass die man−Seiten normalerweise in Standardkapitel unterteilt sind. Die möglichen Kapitel sind:

```
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.
```

Eine einface man–Seite

Hier ist eine einfache man–Seite. Beachte, dass "\–" nötig ist, um den Strich von einem Trennungszeichen zu unterscheiden. Den folgenden Text tippt man einfach in einen Texteditor und speichert ihn als 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)
```

Hier klicken, (cdspeed.html) um obige Seite als html formatierte man–Seite zu sehen.

Betrachten und Formatieren von man–Seiten

Während man seine man–Seite schreibt, sollte man von Zeit zu Zeit die formatierte Seite betrachten, um zu sehen, ob sie richtig aussieht. Tippe dazu:

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

oder

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

Um die man−Seite in reinen Text (keine Farben oder Fettdruck) zu konvertieren, benutzt man:

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

Um die man−Seite in Postscript (zum Drucken oder Weiterverarbeiten in PDF) format zu erhalten:

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

Um die man−Seite in HTML umzuwandeln:

```
man2html your_manpagefile.1
```

Mit perl POD man−Seiten generieren

Ich weiß, dass es viele Leute merkwürdig finden, eine man−Seite in einem Texteditor zu scheiben. Sie wollen die Seite generieren. Das perl POD Dokumentationsformat ist dazu gut geeignt. Man scheibt die man−Seite in POD Syntax und benutzt dann den folgenden Befehl, um die man−Seite zu erzeugen:

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

Die Syntax der POD Dokumentationssprache ist in der man−Seite namens perlpod dokumentiert. Das obige Beispiel würde im POD−format wie folgt aussehen. Beachte, dass Leerzeichen und Leerzeilen eine Rolle in der POD Dokumentationssprache spielen. Es ist wichtig, um die Zeilen mit "=head" Leerzeilen zu lassen.

```
=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)
```

Referenzen

- Man−page HOWTO
- groff_man(7), man−Seitenmakros

2005−01−11, generated by lfparser_pdf version 2.51