NAME

man-pages − conventions for writing Linux man pages

SYNOPSIS

man [*section*] *title*

DESCRIPTION

This page describes the conventions that should be employed when writing man pages for the Linux *man-pages* project, which comprises Sections 2, 3, 4, 5, and 7 of the Linux manual pages. The conventions described on this page may also be useful for authors writing man pages for other projects.

Sections of the Manual Pages

The manual Sections are traditionally defined as follows:

1 Commands (Programs)
Those commands that can be executed by the user from within a shell.

2 System calls
Those functions which must be performed by the kernel.

3 Library calls
Most of the *libc* functions.

4 Special files (devices)
Files found in */dev*.

5 File formats and conventions
The format for */etc/passwd* and other human-readable files.

6 Games

7 Conventions and miscellaneous
Overviews of various topics, conventions and protocols, character set standards, and miscellaneous other things.

8 System management commands
Commands like **mount**(8), many of which only root can execute.

Macro package

New manual pages should be marked up using the **groff an.tmac** package described in **man**(7). This choice is mainly for consistency: the vast majority of existing Linux manual pages are marked up using these macros.

Conventions for source file layout

Please limit source code line length to no more than about 75 characters wherever possible. This helps avoid line-wrapping in some mail clients when patches are submitted inline.

New sentences should be started on new lines. This makes it easier to see the effect of patches, which often operate at the level of individual sentences.

Title line

The first command in a man page should be a **TH** command:

.TH *title section date source manual*

where:

title The title of the man page, written in all caps (e.g., *MAN-PAGES*).

section The section number in which the man page should be placed (e.g., *7*).

date The date of the last revision — remember to change this every time a change is made to the man page, since this is the most general way of doing version control. Dates should be written in the form YYYY-MM-DD.

source The source of the command, function, or system call.

For those few *man-pages* pages in Sections 1 and 8, probably you just want to write *GNU*.

For system calls, just write *Linux*. (An earlier practice was to write the version number of the kernel from which the manual page was being written/checked. However, this was never done consistently, and so was probably worse than including no version number. Henceforth, avoid including a version number.)

For library calls that are part of glibc or one of the other common GNU libraries, just use *GNU C Library*, *GNU*, or an empty string.

For Section 4 pages, use *Linux*.

In cases of doubt, just write *Linux*, or *GNU*.

manual The title of the manual (e.g., for Section 2 and 3 pages in the *man-pages* package, use *Linux Programmer's Manual*).

Sections within a manual page

The list below shows conventional or suggested sections. Most manual pages should include at least the **highlighted** sections. Arrange a new manual page so that sections are placed in the order shown in the list.

NAME
SYNOPSIS
CONFIGURATION [Normally only in Section 4]
DESCRIPTION
OPTIONS [Normally only in Sections 1, 8]
EXIT STATUS [Normally only in Sections 1, 8]
RETURN VALUE [Normally only in Sections 2, 3]
ERRORS [Typically only in Sections 2, 3]
ENVIRONMENT
FILES
VERSIONS [Normally only in Sections 2, 3]
CONFORMING TO
NOTES
BUGS
EXAMPLE
SEE ALSO

Where a traditional heading would apply, please use it; this kind of consistency can make the information easier to understand. If you must, you can create your own headings if they make things easier to understand (this can be especially useful for pages in Sections 4 and 5). However, before doing this, consider whether you could use the traditional headings, with some subsections (*.SS*) within those sections.

The following list elaborates on the contents of each of the above sections.

NAME The name of this manual page. See **man**(7) for important details of the line(s) that should follow the **.SH NAME** command.

SYNOPSIS briefly describes the command or function's interface. For commands, this shows the syntax of the command and its arguments (including options); boldface is used for as-is text and italics are used to indicate replaceable arguments. Brackets ([]) surround optional arguments, vertical bars (|) separate choices, and ellipses (...) can be repeated. For functions, it shows any required data declarations or **#include** directives, followed by the function declaration.

Where a feature test macro must be defined in order to obtain the declaration of a function (or a variable) from a header file, then the SYNOPSIS should indicate this, as described in **feature_test_macros**(7).

CONFIGURATION
Configuration details for a device. This section normally only appears in Section 4 pages.

DESCRIPTION
gives an explanation of what the program, function, or format does. Discuss how it interacts with files and standard input, and what it produces on standard output or standard error. Omit internals and implementation details unless they're critical for understanding the interface. Describe the usual case; for information on command-line options of a program use the **OPTIONS** section.

OPTIONS describes the command-line options accepted by a program and how they change its behavior. This section should only appear for Section 1 and 8 manual pages.

EXIT STATUS lists the possible exit status values of a program and the conditions that cause these values to be returned. This section should only appear for Section 1 and 8 manual pages.

RETURN VALUE
For Section 2 and 3 pages, this section gives a list of the values the library routine will return to the caller and the conditions that cause these values to be returned.

ERRORS For Section 2 and 3 manual pages, this is a list of the values that may be placed in *errno* in the event of an error, along with information about the cause of the errors. *The error list should be in alphabetical order*.

ENVIRONMENT
lists all environment variables that affect the program or function and how they affect it.

FILES lists the files the program or function uses, such as configuration files, startup files, and files the program directly operates on. Give the full pathname of these files, and use the installation process to modify the directory part to match user preferences. For many programs, the default installation location is in */usr/local*, so your base manual page should use */usr/local* as the base.

VERSIONS A brief summary of the Linux kernel or glibc versions where a system call or library function appeared, or changed significantly in its operation. As a general rule, every new interface should include a VERSIONS section in its manual page. Unfortunately, many existing manual pages don't include this information (since there was no policy to do so when they were written). Patches to remedy this are welcome, but, from the perspective of programmers writing new code, this information probably only matters in the case of kernel interfaces that have been added in Linux 2.4 or later (i.e., changes since kernel 2.2), and library functions that have been added to glibc since version 2.1 (i.e., changes since glibc 2.0).

The **syscalls**(2) manual page also provides information about kernel versions in which various system calls first appeared.

CONFORMING TO
describes any standards or conventions that relate to the function or command described by the manual page. For a page in Section 2 or 3, this section should note the POSIX.1 version(s) that the call conforms to, and also whether the call is specified in C99. (Don't worry too much about other standards like SUS, SUSv2, and XPG, or the SVr4 and 4.xBSD implementation standards, unless the call was specified in those standards, but isn't in the current version of POSIX.1.) (See **standards**(7).)

If the call is not governed by any standards but commonly exists on other systems, note them. If the call is Linux-specific, note this.

If this section consists of just a list of standards (which it commonly does), terminate the list with a period ('.').

NOTES provides miscellaneous notes. For Section 2 and 3 man pages you may find it useful to include subsections (**SS**) named *Linux Notes* and *Glibc Notes*.

BUGS lists limitations, known defects or inconveniences, and other questionable activities.

EXAMPLE provides one or more examples describing how this function, file or command is used. For details on writing example programs, see *Example Programs* below.

AUTHORS lists authors of the documentation or program. **Use of an AUTHORS section is strongly discouraged**. Generally, it is better not to clutter every page with a list of (over time potentially numerous) authors; if you write or significantly amend a page, add a copyright notice as a comment in the source file. If you are the author of a device driver and want to include an address for reporting bugs, place this under the BUGS section.

SEE ALSO provides a comma-separated list of related man pages, ordered by section number and then alphabetically by name, possibly followed by other related pages or documents. Do not terminate this with a period.

Font conventions

For functions, the arguments are always specified using italics, *even in the SYNOPSIS section*, where the rest of the function is specified in bold:

int myfunction(int *argc***, char *****argv***);**

Variable names should, like argument names, be specified in italics.

Filenames (whether pathnames, or references to files in the */usr/include* directory) are always in italics (e.g., <*stdio.h*>), except in the SYNOPSIS section, where included files are in bold (e.g., **#include <stdio.h>**). When referring to a standard include file under */usr/include*, specify the header file surrounded by angle brackets, in the usual C way (e.g., <*stdio.h*>).

Special macros, which are usually in upper case, are in bold (e.g., **MAXINT**). Exception: don't boldface NULL.

When enumerating a list of error codes, the codes are in bold (this list usually uses the **.TP** macro).

Complete commands should, if long, be written as in an indented line on their own, for example

```
man 7 man-pages
```

If the command is short, then it can be included inline in the text, in italic format, for example, *man 7 man-pages*. In this case, it may be worth using non-breaking spaces ("\ ") at suitable places in the command. Command options should be written in italics, e.g., *−l*.

Expressions, if not written on a separate indented line, should be specified in italics. Again, the use of non-breaking spaces may be appropriate if the expression is inlined with normal text.

Any reference to the subject of the current manual page should be written with the name in bold. If the subject is a function (i.e., this is a Section 2 or 3 page), then the name should be followed by a pair of parentheses in Roman (normal) font. For example, in the **fcntl**(2) man page, references to the subject of the page would be written as: **fcntl**(). The preferred way to write this in the source file is:

```
.BR fcntl ()
```

(Using this format, rather than the use of "\fB...\fP()" makes it easier to write tools that parse man page source files.)

Any reference to another man page should be written with the name in bold, *always* followed by the section number, formatted in Roman (normal) font, without any separating spaces (e.g., **intro**(2)). The preferred way to write this in the source file is:

.BR intro (2)

(Including the section number in cross references lets tools like **man2html**(1) create properly hyperlinked pages.)

Spelling

Starting with release 2.59, *man-pages* follows American spelling conventions; please write all new pages and patches according to these conventions.

Example Programs and Shell Sessions

Manual pages can include example programs demonstrating how to use a system call or library function. However, note the following:

* Example programs should be written in C.
* An example program is only necessary and useful if it demonstrates something beyond what can easily be provided in a textual description of the interface. An example program that does nothing other than call an interface usually serves little purpose.
* Example programs should be fairly short (preferably less than 100 lines; ideally less than 50 lines).
* Example programs should do error checking after system calls and library function calls.
* Example programs should be complete, and compile without warnings when compiled with *cc −Wall*.
* Where possible and appropriate, example programs should allow experimentation, by varying their behavior based on inputs (ideally from command-line arguments, or alternatively, via input read by the program).
* Example programs should be laid out according to Kernighan and Ritchie style, with 4-space indents. (Avoid the use of TAB characters in source code!)

For some examples of what example programs should look like, see **wait**(2) and **pipe**(2).

If you include a shell session demonstrating the use of a program or other system feature, boldface the user input text, to distinguish it from output produced by the system.

Indentation of structure definitions, shell session logs, etc.

When structure definitions, shell session logs, etc. are included in running text, indent them by 4 spaces (i.e., a block enclosed by *.in +4n* and *.in*).

EXAMPLE

For canonical examples of how man pages in the *man-pages* package should look, see **pipe**(2) and **fcntl**(2).

SEE ALSO

man(1), **man2html**(1), **groff**(7), **groff_man**(7), **man**(7), **mdoc**(7)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.