


Insider Tips: Documentation in Unix and Linux

THE NOTE KEEPER



System admins routinely refer to documentation for help with obscure command line options. In this month's Admin Workshop, we examine the programs and formats behind the command line documentation you read everyday.

BY MARC ANDRÉ SELIG

Most Linux users end up at the command line sooner or later. Command line programs are typically more flexible and efficient than their point&click counterparts. But they do have one disadvantage: innumerable options and parameters that nobody can remember. The expansive list of options often associated with a command line program makes detailed and accessible documentation invaluable to any administrator.

Documentation should avoid complexity: sophisticated presentations are definitely not the kind of reference you need for software applications, and most developers prefer not to waste time prettifying their documentation. These pragmatic considerations have brought forth several forms of online documentation that represent different compromises between readability and ease of creation.

Manual Pages

Manual pages, or manpages for short, are the most commonly used medium. Manpages are quick and easy for a developer to create, and they are easy to keep up to date. In fact, the Linux environment includes special tools that automatically update manpages by referring to the changes in the source code. Manpages are easy to install and extremely user-friendly.

A manpage like the one shown in Figure 1 is a single document that covers a specific topic, typically a single command. Programs developed specifically for this task help users find and display

manpages on the command line, in a GUI, or on the web.

The underlying file format of the manpage comes from Roff, a text layout system that dates back to the early 70s (see the source code in Listing 1). In the modes used by manpages, the format only supports basic formatting, such as headings, highlighting, or enumeration. This kind of simplicity guarantees that anyone will be able to create a manpage with a minimum of effort.

Any recent Linux distribution will have a liberal smattering of manpages – even minimal configurations, as manpages don't need a lot of space. Besides this, many program packages come complete with their own manpages. The Linux Documentation Project [1] has supplementary pages for basic subjects such as `libc` and the kernel.

Users can call a manpage by passing the name of the manpage, typically the command name – as an argument to the `man` program: `man bash`. The “GUI Front-Ends” show some alternatives views.

Navigating the Manpage Jungle

Each manpage is assigned to a specific section: Table 1 has an overview of the eight most important sections. In addition to the numbered sections, there are a number of sections organized by letter. These are typically the letters n (for new), l (local), p (public), and o (old). Many Linux systems put the manpages for the Tcl programming language in n, for example.

Manuals are stored below dedicated directories [2] which makes them easy to find. There is a subdirectory for each section: `man1`, `man2`, and so on. The `/etc/manpath.config`, and the `MANPATH` environmental variable, tell `man` where to look for documentation.



Figure 1: Manpages give users a quick and convenient way of viewing documentation.

Distributing the manpages across multiple directories was a welcome change in the early days of Unix. The UFS file system common at the time took a big performance hit the more files you stored in a directory.

Depending on the configuration, `man` stores formatted manpages (so called cat pages) in directories set up for this purpose. This arrangement removes the need to format the manpage the next time a user calls it – an obvious advantage on older or low-end machines that do not have very much CPU power. Some systems mount the `/usr` partition read-only to improve security. In this

case you would need to store cat pages below `/var`.

Additional folders provide localized manpages. For example, the English page for `chmod` would be stored in `/usr/share/man/man1/chmod.1`, whereas the German version would reside in `/usr/share/man/de/man1/chmod.1`.

Apropos, What is...?

If you are looking for a specific command and aren't sure if you have found the right one, the `apropos` and `whatis` commands can be lifesavers. `apropos` checks the headings of all stored manpages for the specified argument and even finds arguments in substrings.

`whatis` uses a similar approach but only finds complete matches in whole words within headings. Both `apropos` and `whatis` use a binary database for fast information retrieval. Depending on your distribution and Unix version, the `makewhatis` or `mandb` commands are responsible for creating the database file. The command is run by `cron` or `anacron` [3] at regular intervals to keep the directories up to date.

Do-It-Yourself

Internally, `man` mainly uses the Roff tools `nroff`, or `groff`. On GNU systems, `nroff` simply calls `groff` to emulate the behavior of the legacy `nroff` command. `nroff` is used for ASCII output, whereas the GNU tool can handle other file formats, such as Postscript and HTML.

Listing 1: Manpage Source Code

```

01 .TH man 1 "14 May 2001"          14 [...]
   "2.3.19" "Manual pager utils"  15 .SH DESCRIPTION
02 .SH NAME                        16 .B man
03 man \- an interface to the     17 is the system's manual pager.
   on-line reference manuals     Each
04 .SH SYNOPSIS                    18 .I page
05 .\ " The general command line  19 argument given to
06 .B man                          20 .B man
07 .RB [\] \-c [\]|\[\] \-w [\]|\[\]  21 is normally the name of a
   \-tZT                          program, utility or function.
08 .IR device [\]                  22 The
09 .\ "                              23 .I manual page
10 [...]                            24 associated with each of these
11 .RI [\]|\[\] section \[\]       25 arguments is then found and
12 .IR page \ .\|\.\|\.\|\.\|\.\  displayed.
13 .\ "

```



Figure 2: xman allows admins to view the installed manpages with a neat graphical front-end. xman can also display a list of manpages for a specific section

The following command changes the source code of a manpage to ASCII, (or Latin 1 to be more precise), and uses the less pager to display the results: `nroff -man manpage.1 | less`.

The `-m` option specifies the macro package that `nroff` should use to interpret the formatting.

The `an` macro package contains typical manpage commands. Many Linux systems use the extended macro package, `andoc` instead (you need to type `-mandoc` to call `andoc`). The following command converts a manpage to Postscript: `groff -Tps -man manpage.1 > manpage.ps`. The `-T` option specifies the desired output format.

Creating a manpage is quite simple. `Roff` commands always start with a dot; the required options follow in the same line. There are just a handful of important formatting commands for man-

pages: `.TH` initiates a manpage and requires a short heading, the section number, the date, version, and extended heading as its arguments.

When output, this data is displayed in the header and footer lines. `.SH` gives you a sub-heading, `.B` is for bold, `.I` for italics (or underlined, depending on your terminal). An empty line ends a paragraph, and typing `.BR` gives you a line break.

Before you rush off to copy an original manpage and replace its content with your own, a

word of warning: headings (`.SH`) are normalized. The first section, NAME, has special significance, as its content is parsed by the databases described in the following sections.

Size Matters

The biggest advantage that manpages offer is that they are small and simple. However, things start to get more difficult when a developer needs to document more complex scenarios or software packages. Take the manpages for

the various shells, for example, which all describe a complete programming language along with the accompanying tools and tips; this is just too much of a good thing. The manpage for Bash comprises about 4000 lines (or 60 pages of hard copy) and is anything but intuitive.

Perl demonstrates one possible approach to avoiding manpage overkill by dividing the manpages into a number of sections, allowing users to load the sections they need. A welcome page provides navigational aids and a table of contents.

Info System

The info system is another alternative to the overkill of an overgrown manpage. Instead of the linear format adopted by manpages, info uses a tree structure with menus and submenus to organize the text (Figure 3). Additional cross references integrated with the tree allow users to jump back and forward quickly between related topics. And a well designed navigational system makes it easier for users to find their way around the tree structure.

Using a tree to distribute information reflects the way people use computers. If you start using a new scripting language, for instance, you will first need a short introduction, then you can skip the unnecessary topics and go directly to the information you need right now. A well-

Table 1: Man Sections

Section	Description
1	Executables or shell commands
2	System calls (kernel functions)
3	Library calls (functions in system libraries)
4	Special files (typically in /dev)
5	File formats and conventions, e.g. /etc/passwd
6	Games
7	Macro packages and conventions (such as <i>XF86Config</i> , <i>man</i>)
8	Sysadmin commands (such as <i>lsmold</i> , <i>ifup</i>)



Figure 3: Typing the info command at the command line launches Info, a simple Help system in a tree format.

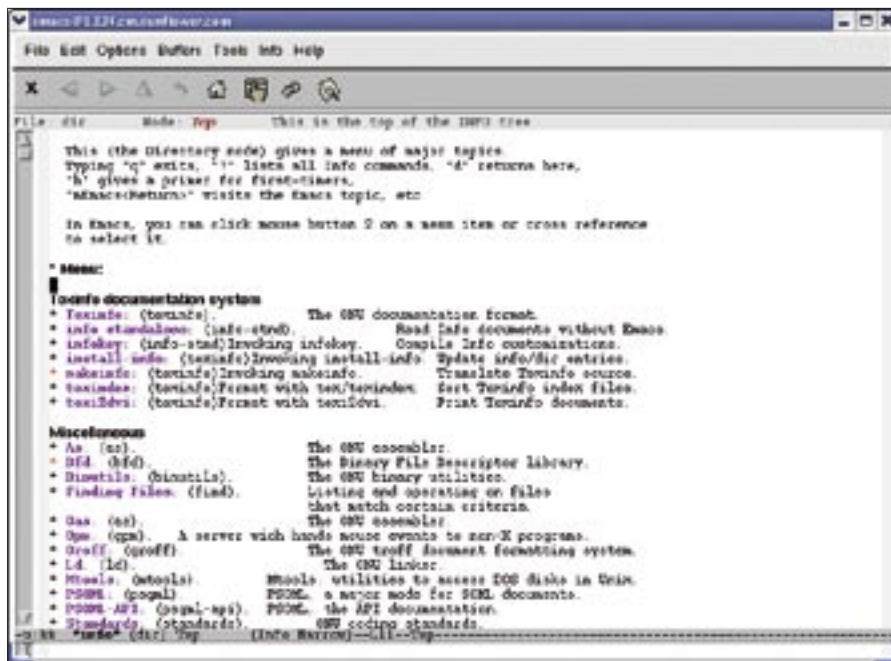


Figure 4: The Emacs editor allows users to browse the info system. Running this under X adds mouse support to the info menus.

organized and clearly arranged structure is a big help.

Info and the User

The *info* command allows users to navigate the info system. In addition to this, the Emacs and XEmacs editors both have info system navigation modes (Figure 4). In the most simple case, the user will want to browse the info pages from top to tail. Pressing the space key takes you forwards through the text, and pressing backspace takes you back.

Alternative navigational steps make more sense when a text branches off based on conditions. Menus are tagged as ** Menu:* and give you an overview of their subordinate nodes. You can press [M] to jump to a menu item. Simply type the first few letters of the menu label at the prompt, and press [Tab] to complete the entry. Moving the cursor to the menu item, and pressing [Enter], has the same effect.

Other shortcuts include [N] (for next – go to the next page), [P] (previous – go back one page), and [U] (up – go to the parent page). *see* tags point the way to cross references, and you can press [F] to follow them. Finally, [L] takes you back to the last page you viewed.

Info File Source Code

Before they are converted to a format that the info reader understands, info

source files (which are identifiable by the *.texi* file extension) look just like TeX documents with some additional tagging. TeX commands start with a backslash, \, as usual, whereas info commands use an at sign (@.)

To create your own *info* files, you need the *texinfo* program package, which contains detailed documentation on the *texi* format – in *info* file format, of course – among other things. *.texi* files can be converted to *.info* files using the *makeinfo* command. The files typically reside in */usr/info* or */usr/share/info*, as does the *dir* file, which gives you an

GUI Front-Ends

Graphical front-ends that support point&click operations are available both for manpages and for the info system. For example, *xman* is a simple but convenient interface. Administrators can press [Ctrl+S] to search for a manpage. Alternatively, you can display all the manpages in a specific section, as shown in Figure 2. If you prefer to use your browser for browsing documentation, you might like to try *man2html*. There is an enormous collection available online at [4], for example.

tkinfo displays info pages on the desktop, however, functions and appearance resemble the original (see Figure 3), which also supports mouse operations in Emacs, for example.

overview of the *info* files installed on your system. The *info* reader displays the *dir* file as a kind of “homepage” when launched.

You can also run TeX to convert *texi* files to DVI for output, or to convert to other formats such as PDF and Postscript. TeX is capable of reading *texi* files. The file calls a macro package in the first line (*\input texinfo*), and this allows TeX to interpret the *at* commands that follow. TeX returns a DVI file, along with a number of index files, as output. *makeinfo* and *co.* read the same *at* commands that TeX reads but return an *info* document for interactive, on screen output.

Readmes, HOWTOs, and FAQs

Faced with this choice of documentation tools, admins should make sure they don’t forget the simplest form of documentation. Text only files (typically labeled *README*) are provided with most programs and often contain vital information. Linux distributions typically hide these files below */usr/share/doc/*. There is a large collection of HOWTOs at [1], and a visit is worth the effort, as the collection could help you navigate many critical Linux tasks, such as setting up WLAN cards, burning DVDs, and setting up LDAP. ■

INFO

- [1] The Linux Documentation Project: <http://www.tldp.org>
- [2] Filesystem hierarchy standard for manual pages: <http://www.pathname.com/fhs/pub/fhs-2.3.html#USRSHAREMANMANUALPAGES>
- [3] Marc André Selig, “Cron, At, Anacron”: Linux Magazine 12/04, p. 66
- [4] Manpages on the web: <http://linux.ctyme.com>

THE AUTHOR

Marc André Selig spends half of his time working as a scientific assistant at the University of Trier and as an ongoing medical doctor in the Schramberg hospital. If he happens to find time for it, his current preoccupation is programming web based databases on various Unix platforms.

