NAME
manpages -- An introduction to manual pages

DESCRIPTION
Manual pages (often shortened to "man pages") are a means of providing documentation on the command line. Most manual pages describe low-level programming interfaces, command-line tools, and file formats.

This manual page is intended to help you learn about manual pages, their purpose, their style, and their overall layout so that you can better take advantage of the content that they provide.

MANUAL PAGE STYLE
Because manual pages are written by software engineers from many different companies and organizations around the world, the format of these manual pages varies somewhat from page to page, as does the style.

In general, however, manual pages are written to be as concise as possible. While this makes them a somewhat difficult place to start as a new programmer, this is intentional. They are not intended to replace conceptual documentation such as books on programming. They are intended primarily to provide a quick reference for people who are already somewhat familiar with the subject.

Some manual pages provide links to outside websites. You can often find more thorough conceptual documentation, sample code, and other useful information at these websites.

Manual pages also link to other manual pages. When they do, these links are in the form **name(section).** For example, the manual page **ls(1)** describes the **ls** command, and appears in section **1** of the manual. The manual sections are described below.

MANUAL PAGE SECTIONS
The manual is divided into sections. Each section covers a particular subject area. The major manual page sections are:

1	General User Commands
2	System Calls
3	Library Routines (*)
4	Special Files and Sockets
5	File formats and Conventions
6	Games and Fun Stuff
7	Miscellaneous Documentation
8	System Administration
9	Kernel and Programming Style
n	Tcl/Tk

(*) Excludes library routines that merely wrap system calls. Those routines are covered in section 2.

The majority of commonly used commands appear in sections 1 and 8 of the manual. To learn more about these commands, read the individual pages or read Shell Scripting Primer at <http://developer.apple.com/mac/library/documentation/opensource/conceptual/shellscripting>.

Most programming information is found in sections 2 and 3.

These subject areas may be further subdivided. For example, manual section 3 was originally intended to hold documentation in the standard C library but has been expanded to include functions in other C language libraries, such as section 3ssl (OpenSSL functions).

Section 3 has even been expanded to include other programming languages. For example, section 3pm contains documentation for Perl modules).

It is common to have multiple manual pages with the same name but different section numbers. This usually occurs when a command shares a name with a C function or system call that does something very similar. To avoid any confusion, manual pages are commonly referred to in the form name(number), in which the number is the section number.

MANUAL PAGE TOOLS
You can read manual pages in a number of ways. The most common way to read manual pages is with the man(1) tool from the command line. For example, typing "man man" displays the manual page for the man tool.

If there are multiple manual pages with the same name but different section numbers (for example, open(1) and open(2)), you can specify a section number when requesting the page. For example, the command "man 2 open" displays the manual page for the "open" system call from section 2 of the manual.

You can also read manual pages from within Xcode by choosing the "Open man page..." option in the Help menu, or within your choice of web browsers by going to <http://developer.apple.com/documentation/Darwin/Reference/ManPages/>.

In addition to searching for manual pages on the web, the manual page architecture also provides two useful command-line tools for searching the manual pages: whatis and apropos.

The whatis(1) command searches the manual page headings (command and function names) for a word. If that word appears in its entirety, it shows the matching name and abstract. For example, typing "whatis man" returns results for man and man.conf. This is mostly of interest if you want to read an abstract for a particular command.

The apropos(1) command is a much more user-friendly version of whatis. It searches the same database, but searches the manual page abstracts as well as the titles. Unlike the whatis command, apropos returns results for partial word matches.

NOTE: Both apropos and whatis depend on a database to provide information. This database is updated periodically. On fresh installations, however, it may not be present. If apropos and whatis are not working correctly, you should run the following command as an admin user:

```
sudo /usr/libexec/makewhatis
```

This will rebuild the database. Enter your admin password when prompted.

MANUAL PAGE STRUCTURE

Manual pages do not have a fixed structure. However, most manual pages do follow certain conventions.

Manual pages typically begin with a **NAME** section, which contains the name of a command or function and a short abstract.

Next, manual pages typically include a **SYNOPSIS** section, which describes how to use the command or function. For functions, the syntax generally contains the necessary include directives, followed by the function declarations themselves. For commands, the syntax is explained in **MANUAL PAGE SYNTAX.**

After the **SYNOPSIS** section, you will generally find a **DESCRIPTION** section, followed by an **OPTIONS** section (which explains the flags from the **SYNOPSIS** section.

You may find sections such as **ENVIRONMENT HISTORY, BUGS, CONFORMING TO, AUTHOR,** and **COPYRIGHT.**

Finally, most manual pages end with a section called **SEE ALSO,** which includes the names and section numbers of related manual pages.

MANUAL PAGE SYNTAX

In manual page syntax, anything in a normal text font is required text. Anything in a **boldface** font is a flag or a subcommand. Anything underlined is a user-specified argument such as a filename.

Any argument surrounded by brackets is considered to be optional. For example, [filename] would indicate an optional filename argument.

Flags, arguments, or subcommands separated by a vertical separator (|) are mutually exclusive. For example, if -a turns on an option and -b turns off the option, the syntax for this command might be **-a | -b.**

In some cases, you may even see entire groups of arguments wrapped with brackets and separated by a vertical separator. This is one way of showing that a command has more than one valid syntax. In other manual pages, this is expressed by having multiple lines in the synopsis, each of which begins with the command name. The separated format is more common (and more readable), but is not always possible for commands with particularly complex syntax.

Finally, the most important notational convention is the use of the ellipsis (...). This indicates that additional arguments may be added at this point. Depending on the author, you may see this written in one of two ways:

```
argument [ argument... ]
argument...
```

SEE ALSO

For more information on manual pages, see man(1), intro(1), intro(2), intro(3), intro(5), intro(7), intro(8), intro(9), and the developer documentation website at <http://developer.apple.com/documentation/Darwin/Reference/ManPages/>.

Reporting Problems

The way to report a problem with this manual page depends on the type of problem:

Content errors
Report errors in the content of this documentation with the feedback links below.

Bug reports
Report bugs in the functionality of the described tool or API through Bug Reporter.

Formatting problems
Report formatting mistakes in the online version of these pages with the feedback links below.