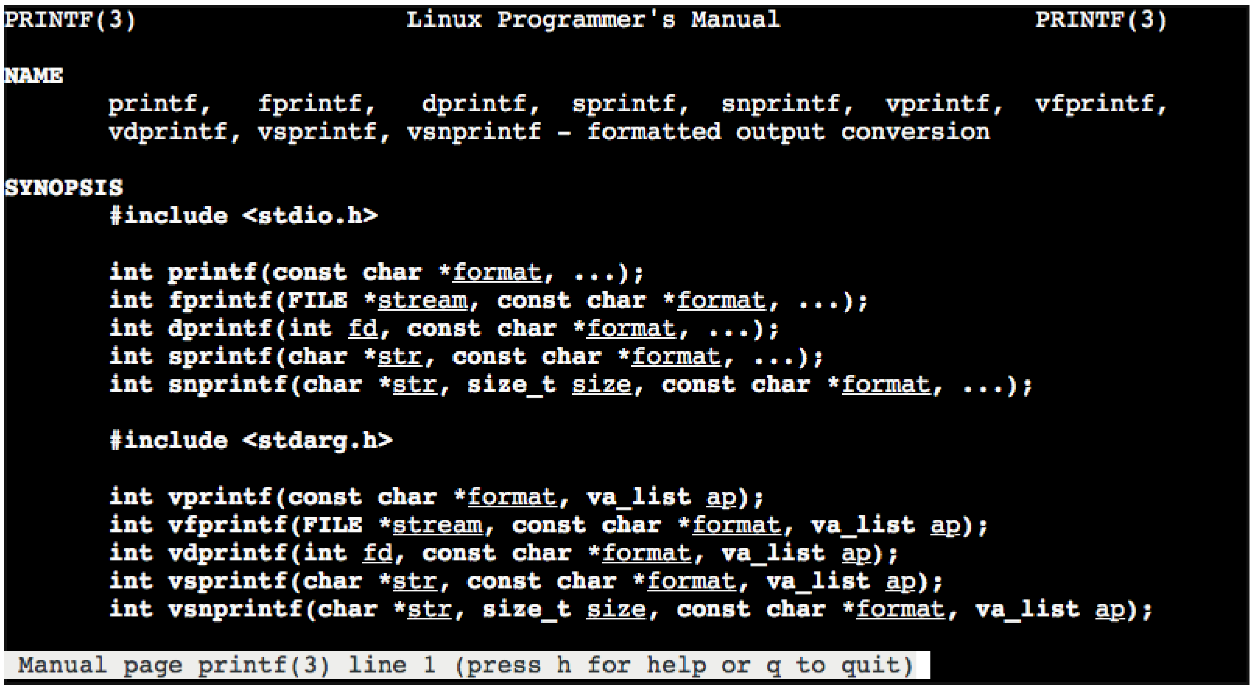
Getting Help: man Pages

/run/media/lijunjie/资料/Learning Video/writing-running-fixing-code/02_compiling-and-running/02_first-example/06_getting-help-man-pages_instructions.html

As you write your own programs, you will want to learn some commands that provide built-in help. The first, and most versatile of these is the man command (which is short for “manual”). This command displays the manual page (“man page” for short) for whatever you request—commands, library functions, and a variety of other important topics. For example, if you type *man -S3 printf,* then your computer will display the man page for the *printf* function from the C library.

Like many UNIX commands, *man* takes arguments on the command line to specify exactly what it should do. In the case of *man* , these arguments (typically) specify which page (or pages) you want it to display for you. In the example above, we gave the *man* command two arguments: *-S3* and *printf* . The *-S3* argument tells *man* to look in section 3 of the manual, which is dedicated to the C library.



```
PRINTF(3)                  Linux Programmer's Manual                 PRINTF(3)

NAME
       printf,   fprintf,   dprintf,  sprintf,  snprintf,  vprintf,  vfprintf,
       vdprintf, vsprintf, vsnprintf - formatted output conversion

SYNOPSIS
       #include <stdio.h>

       int printf(const char *format, ...);
       int fprintf(FILE *stream, const char *format, ...);
       int dprintf(int fd, const char *format, ...);
       int sprintf(char *str, const char *format, ...);
       int snprintf(char *str, size_t size, const char *format, ...);

       #include <stdarg.h>

       int vprintf(const char *format, va_list ap);
       int vfprintf(FILE *stream, const char *format, va_list ap);
       int vdprintf(int fd, const char *format, va_list ap);
       int vsprintf(char *str, const char *format, va_list ap);
       int vsnprintf(char *str, size_t size, const char *format, va_list ap);

 Manual page printf(3) line 1 (press h for help or q to quit)
```

Before we delve into the options and the details of the various sections of the manual, we will look at what the manual displays in a bit more detail. The figure above shows the output of *man -S3 printf* . This page actually has information not only for printf, but also for a variety of related functions, which are all listed at the top of the page. The SYNOPSIS section lists the *#include* file to use, as well as the functions’ prototypes. At the bottom of the screen is the start of the DESCRIPTION section which describes the behavior of the function in detail. This description runs off the bottom of the screen, but you can scroll up and down with the arrow keys. You can also use d and u to scroll a page at a time. You can also quit by pressing q. These are the most important and useful keys to know, but there are a variety of other ones you can use, which you can find out about by pressing h (for help).

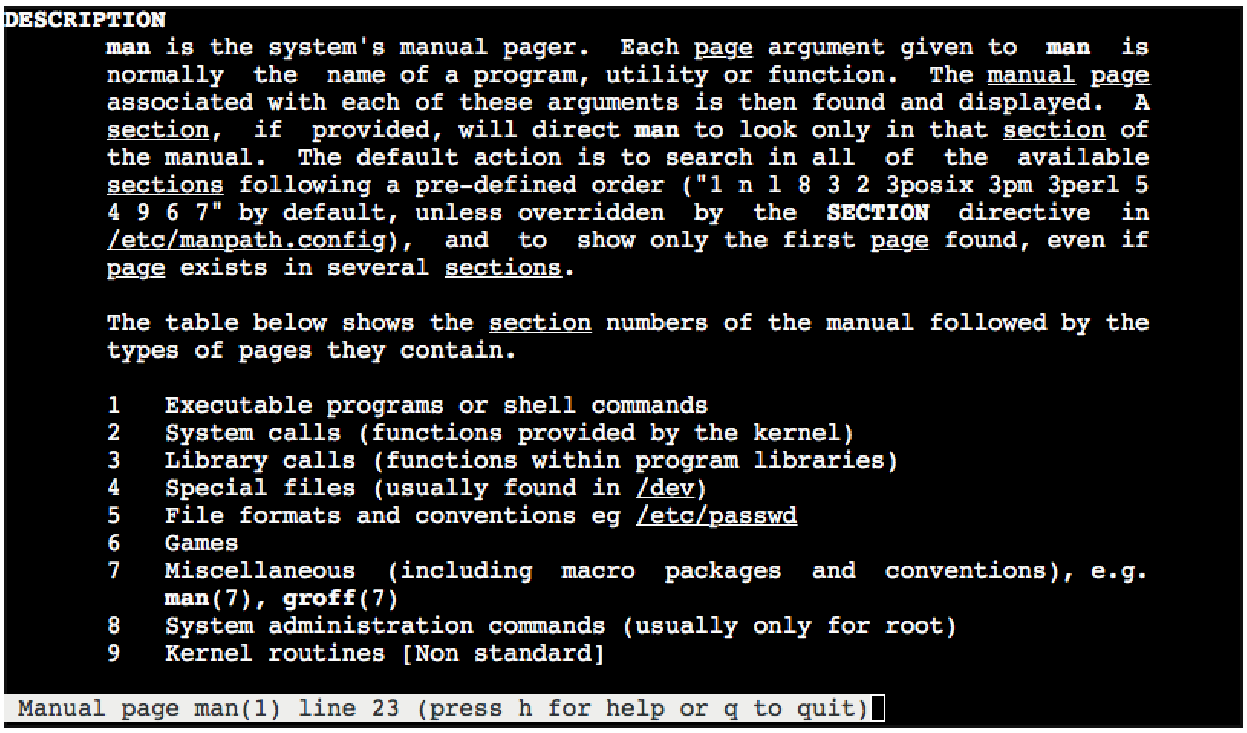
If you were to continue scrolling down through the man page for *printf,* you would find out everything you could ever want to know about it (including all the various options and features for the format string, what values it returns under various conditions, etc.). We are not interested in the details of *printf* for this discussion, only that the man page provides them.

The manual includes pages on topics other than just the C library, such as commands. For example, the *ls* command. If you wanted to know more details of this command, you could do *man ls* to read about it. The manual page describes what command line arguments ls expects, as well as the details of the various options it accepts.

Unlike *printf* , we did not specify a section of the manual for ls. In fact, not specifying the section explicitly is the common case— *man* will look through the sections sequentially trying to find the page we requested. If there is nothing with the same name in an earlier section of the manual, then you do not need to specify the section. In the case of *ls* , the page we are looking for is in Section 1—which has information about executable programs and shell commands. In fact, when you run *man ls,* you can see that it found the page in section 1 by looking in the top left corner, where you will see *LS(1)* . The *(1)* denotes section 1 of the manual.

If we just type *man printf* we get the man page for the *printf command* from section 1 *("printf(1)")* . This page corresponds to the executable command *printf* which lets you print things at your shell. For example, you could type printf " *Hello %d\n" 42* at your shell and it would print out *Hello 42*. When writing a script, it might be useful to print information out such as this. Since *man* finds this page first, if we want the C library function printf (for example, if we are programming and need to look up a format specifier that we do not remember), we need to explicitly ask for section 3 with the *-S3* option, as section 3 has C library reference.

So far, we have seen two sections of the manual: 1, which is for executable programs and shell commands, and 3, which is for C library function reference. How would we find these out if we did not have this book handy? Also, how do we find out about the other sections of the manual? The *man* command, like most other commands has its own manual page too, so we could just read that. In fact, if we type *man man,* the computer will display the manual page for the man command. Scrolling down a screen or so into the DESCRIPTION section shows the following table of sections:



```
DESCRIPTION
       man is the system's manual pager.  Each page argument given to  man  is
       normally  the  name of a program, utility or function.  The manual page
       associated with each of these arguments is then found and displayed.  A
       section,  if  provided, will direct man to look only in that section of
       the manual.  The default action is to search in all  of  the  available
       sections following a pre-defined order ("1 n l 8 3 2 3posix 3pm 3perl 5
       4 9 6 7" by default, unless overridden  by  the  SECTION  directive  in
       /etc/manpath.config),  and  to  show only the first page found, even if
       page exists in several sections.

       The table below shows the section numbers of the manual followed by the
       types of pages they contain.

       1   Executable programs or shell commands
       2   System calls (functions provided by the kernel)
       3   Library calls (functions within program libraries)
       4   Special files (usually found in /dev)
       5   File formats and conventions eg /etc/passwd
       6   Games
       7   Miscellaneous  (including  macro  packages  and  conventions), e.g.
           man(7), groff(7)
       8   System administration commands (usually only for root)
       9   Kernel routines [Non standard]

 Manual page man(1) line 23 (press h for help or q to quit)
```

Scrolling down further in the manual will show various examples of how to use *man* , as well as the various options it accepts.

New users of the man system often face the conundrum that reading a *man* page is great for the details of something if you know what you need, but how do you find the right page if you do not know what you are looking for? There are two main ways to find this sort of information. The first is to use the *-k* option, which asks *man* to do a keyword search. For example, suppose you wanted to find a C function to compare two strings. Running the command *man -k* compare lists about 56 commands and C library functions that have the word “compare” in their description. You can then look through this list, find things that look relevant, and read their respective pages to find the details.

The other way to find things is to look in the SEE ALSO section at the end of another page if you know something related but not quite right. This section, which you can find at the end of each man page, lists the other pages which the author thought were relevant to someone reading the page she wrote.