

Working With Commands

Up until now you have seen a number of commands and their mysterious options and arguments. In this lesson, we will try to remove some of that mystery. This lesson will introduce the following commands.

- [`type`](#) - Display information about command type
- [`which`](#) - Locate a command
- [`help`](#) - Display reference page for shell builtin
- [`man`](#) - Display an on-line command reference

What Are "Commands?"

Commands can be one of 4 different kinds:

1. **An executable program** like all those files we saw in `/usr/bin`. Within this category, programs can be *compiled binaries* such as programs written in C and C++, or programs written in *scripting languages* such as the shell, Perl, Python, Ruby, etc.
2. **A command built into the shell itself.** bash provides a number of commands internally called *shell builtins*. The `cd` command, for example, is a shell builtin.
3. **A shell function.** These are miniature shell scripts incorporated into the *environment*. We will cover configuring the environment and writing shell functions in later lessons, but for now, just be aware that they exist.
4. **An alias.** Commands that you can define yourselves, built from other commands. This will be covered in a later lesson.

Identifying Commands

It is often useful to know exactly which of the four kinds of commands is being used and Linux provides a couple of ways to find out.

`type`

The **`type`** command is a shell builtin that displays the kind of command the shell will execute, given a particular command name. It works like this:

```
type command
```

where “command” is the name of the command you want to examine. Here are some examples:

```
[me@linuxbox me]$ type type
type is a shell builtin
```

```
[me@linuxbox me]$ type ls
ls is aliased to `ls --color=tty'
```

```
[me@linuxbox me]$ type cp
```

```
cp is /bin/cp
```

Here we see the results for three different commands. Notice that the one for ls (taken from a Fedora system) and how the ls command is actually an alias for the ls command with the “-- color=tty” option added. Now we know why the output from ls is displayed in color!

which

Sometimes there is more than one version of an executable program installed on a system. While this is not very common on desktop systems, it's not unusual on large servers. To determine the exact location of a given executable, the **which** command is used:

```
[me@linuxbox me]$ which ls
/bin/ls
```

which only works for executable programs, not builtins nor aliases that are substitutes for actual executable programs.

Getting Command Documentation

With this knowledge of what a command is, we can now search for the documentation available for each kind of command.

help

bash has a built-in help facility available for each of the shell builtins. To use it, type “help” followed by the name of the shell builtin. Optionally, you may add the -m option to change the format of the output. For example:

```
[me@linuxbox me]$ help -m cd
```

```
NAME
    cd - Change the shell working directory.
```

```
SYNOPSIS
    cd [-L|-P] [dir]
```

```
DESCRIPTION
    Change the shell working directory.
```

```
Change the current directory to DIR.  The default DIR is the value of the
HOME shell variable.
```

```
The variable CDPATH defines the search path for the directory containing
DIR.  Alternative directory names in CDPATH are separated by a colon (:).
A null directory name is the same as the current directory.  If DIR begins
with a slash (/), then CDPATH is not used.
```

```
If the directory is not found, and the shell option `cdable_vars' is set,
```

the word is assumed to be a variable name. If that variable has a value, its value is used for DIR.

Options:

- L force symbolic links to be followed
- P use the physical directory structure without following symbolic links

The default is to follow symbolic links, as if `-L` were specified.

Exit Status:

Returns 0 if the directory is changed; non-zero otherwise.

SEE ALSO

bash(1)

IMPLEMENTATION

GNU bash, version 4.1.5(1)-release (i486-pc-linux-gnu)

Copyright (C) 2009 Free Software Foundation, Inc.

A note on notation: When square brackets appear in the description of a command's syntax, they indicate optional items. A vertical bar character indicates mutually exclusive items. In the case of the `cd` command above:

```
cd [-L|-P] [dir]
```

This notation says that the command `cd` may be followed optionally by either a “-L” or a “-P” and further, optionally followed by the argument “dir”.

--help

Many executable programs support a “--help” option that displays a description of the command's supported syntax and options. For example:

```
[me@linuxbox me]$ mkdir --help
```

Usage: mkdir [OPTION] DIRECTORY...

Create the DIRECTORY(ies), if they do not already exist.

-Z, --context=CONTEXT (SELinux) set security context to CONTEXT
Mandatory arguments to long options are mandatory for short options too.

- m, --mode=MODE set file mode (as in chmod), not a=rwx - umask
- p, --parents no error if existing, make parent directories as needed
- v, --verbose print a message for each created directory
- help display this help and exit
- version output version information and exit

Some programs don't support the “--help” option, but try it anyway. Often it results in an error message that will reveal similar usage information.

man

Most executable programs intended for command line use provide a formal piece of documentation called a *manual* or *man page*. A special paging program called **man** is used to view them. It is used like this:

```
man program
```

where “program” is the name of the command to view. Man pages vary somewhat in format but generally contain a title, a synopsis of the command's syntax, a description of the command's purpose, and a listing and description of each of the command's options. Man pages, however, do not usually include examples, and are intended as a reference, not a tutorial. As an example, let's try viewing the man page for the **ls** command:

```
[me@linuxbox me]$ man ls
```

On most Linux systems, **man** uses **less** to display the manual page, so all of the familiar **less** commands work while displaying the page.

README And Other Documentation Files

Many software packages installed on your system have documentation files residing in the `/usr/share/doc` directory. Most of these are stored in plain text format and can be viewed with **less**. Some of the files are in HTML format and can be viewed with your web browser. You may encounter some files ending with a “.gz” extension. This indicates that they have been compressed with the **gzip** compression program. The gzip package includes a special version of **less** called **zless** that will display the contents of gzip-compressed text files.

© 2000-2019, [William E. Shotts, Jr.](#) Verbatim copying and distribution of this entire article is permitted in any medium, provided this copyright notice is preserved.

Linux® is a registered trademark of Linus Torvalds.