Read Your Manuals!

You may be telling yourself right now, "Linux is so hard! There are a lot of commands and even more command options! There is no way I can master all of these commands and commit them to memory." If this is what you think, believe me, you are smart. It's insane to remember all the Linux commands that exist, even the most experienced Linux administrator would never be able to remember all commands, not even Linus Torvalds himself!

So wait? If that's the case, what is the solution then? The answer lies in the beautiful world of Linux documentation. Linux is very well documented to the extent that it's hard to get lost in it. There are a variety of tools in Linux that help you in not just remembering the commands, but also in understanding how to use them.

Having met a lot of Linux professionals throughout my career, I noticed that the most skilled Linux administrators are not the ones who remember, but the ones who know how to make the most use of the Linux documentation. Ladies and gentlemen, I highly recommend you fasten your seatbelt and read this chapter carefully. I promise you that the fear in your heart will go away soon!

The four categories of linux commands

All Linux commands must fall into one of these following four categories:

1. **An executable program**: Which is usually written in the C programming language. The [cp] command is an example of an executable command.
2. **An alias**: Which is basically another name for a command (or a group of commands).
3. **A shell builtin**: The shell supports internal commands as well. The [exit] and [cd] commands are two examples of a shell builtin command.

```
<!-- -->
```

4. **A shell function**: These are functions that help us achieve a specific task and are essential in writing shell scripts. We will cover this in more detail later, for now, just know they exist.

Determining a command's type

You can use the [type] command to determine the type (category) of a command. For example, if you want to know the type of the [pwd] command you can simply run the [type pwd] command:

```
elliot@ubuntu-linux:~$ type pwd
pwd is a shell builtin
```

So now you know that the [pwd] command is a shell builtin command. Now let's figure out the type of the [ls] command:

```
elliot@ubuntu-linux:~$ type ls
ls is aliased to `ls --color=auto'
```

As you can see, the [ls] command is aliased to [ls --color=auto]. Now you know why you see a colorful output every time you run the [ls] command. Let's see the type of the [date] command:

```
elliot@ubuntu-linux:~$ type date
date is /bin/date
```

Any command that lives in [/bin] or [/sbin] is an executable program. Therefore, we can conclude that the [date] command is an executable program as it resides in [/bin].

Finally, let's determine the type of the [type] command itself:

```
elliot@ubuntu-linux:~$ type type
type is a shell builtin
```

It turns out the [type] command is a shell builtin command.

Finding a command's location

Every time you run an executable command, there a file somewhere on the system that gets executed. You can use the [which] command to determine the location of an executable command. For example, if you want to know the location of the [rm] command, you can run the [which rm] command:

```
elliot@ubuntu-linux:~$ which rm
/bin/rm
```

So now you know that [rm] lives in the [/bin] directory. Let's see the location of the [reboot] command:

```
elliot@ubuntu-linux:~$ which reboot
/sbin/reboot
```

As you can see, the [reboot] command lives in the [/sbin] directory.

What does the command do?

You can use the [whatis] command to get a brief description of what a command does. For example, if you want to know the purpose of the [free] command, you can run the [whatis free] command:

```
elliot@ubuntu-linux:~$ whatis free
free (1)             - Display amount of free and used memory in the system
```

As you can see, the [free] command, as we already know, displays the amount of free and used memory in the system. Cool! Now let's see what the [df] command does:

```
elliot@ubuntu-linux:~$ whatis df
df (1)               - report file system disk space usage
```

Finally, let's see what the [which] command does:

```
elliot@ubuntu-linux:~$ whatis which
which (1)            - locate a command
```

As we already know, [which] displays a command's location.

Help for shell builtins

If you play around enough with [man] pages, you may notice that a lot of shell builtin commands do not have a [man] page. For instance, there is no [man] page for the [cd] or the [exit] commands:

```
elliot@ubuntu-linux:~$ type cd
cd is a shell builtin
```

```
elliot@ubuntu-linux:~$ man cd
No manual entry for cd
elliot@ubuntu-linux:~$ type exit
exit is a shell builtin
elliot@ubuntu-linux:~$ man exit
No manual entry for exit
```

That's because shell builtin commands do not have [man] pages, but do not freak out just yet! You can still find help on how to use shell builtins by using the [help] command. For example, to get help on how to use the [exit] command, you can run:

```
elliot@ubuntu-linux:~$ help exit
exit: exit [n]
    Exit the shell.

    Exits the shell with a status of N. If N is omitted, the exit status
    is that of the last command executed.
```

Similarly, to get help on how to use the [cd] command, you can run the [help cd] command:

```
elliot@ubuntu-linux:~$ help cd
cd: cd [-L|-P] [dir]
    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.
```

The info page

The GNU project launched the [info] pages as an alternative documentation to the [man] pages. The GNU project once claimed that [man] pages are outdated and needed replacement and so they came up with the [info] pages.

You can view the [info] page of any command by running:

```
info command_name
```

For example, to view the [info] page of the [ls] command, you can run the [info ls] command:

```
elliot@ubuntu-linux:~$ info ls

Next: dir invocation, Up: Directory listing

10.1 'ls': List directory contents
==================================

The 'ls' program lists information about files (of any type, including directories).
Options and file arguments can be intermixed arbitrarily, as usual.

For non-option command-line arguments that are directories, by default 'ls' lists the
contents of directories, not recursively, and omitting files with names beginning with
'.'. For other non-option arguments, by default 'ls' lists just the file name. If no
non-option argument is specified, 'ls' operates on the current directory, acting as if
it had been invoked with a single argument of '.'.

By default, the output is sorted alphabetically, according to the locale settings in
effect.(1) If standard output is a terminal, the output is in columns (sorted
vertically) and control characters are output as question marks; otherwise, the output
is listed one per line and control characters are output as-is.

Because 'ls' is such a fundamental program, it has accumulated many options over the
years. They are described in the subsections below; within each section, options are
listed alphabetically (ignoring case). The division of options into the subsections is
not absolute, since some options affect more than one aspect of 'ls''s operation.
```

The /usr/share/doc directory

The [/usr/share/doc] directory is another excellent place to look for help in Linux. This directory has very intensive documentation; it doesn't just show you how to use a command; sometimes, it will even show the name and contact information of the authors who developed the command. Moreover, it may also include a [TODO] file that contains a list of unfinished tasks/features; contributors usually check the [TODO] files to help fix bugs and develop new features.

To demonstrate, let's go to the [nano] documentation directory:

```
elliot@ubuntu-linux:~$ cd /usr/share/doc/nano
elliot@ubuntu-linux:/usr/share/doc/nano$ pwd
/usr/share/doc/nano
```

Now list the contents of the directory to see what's inside:

```
elliot@ubuntu-linux:/usr/share/doc/nano$ ls

changelog.Debian.gz  copyright  examples
```

As you saw in this chapter, Linux has a variety of helpful tools available at your disposal; so make sure you utilize them!

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. You need to know if the [echo] command is a shell builtin or an executable program, which command would you run?
2. Display the location of the [uptime] command executable file.
3. Show a brief description of the [mkdir] command.
4. You forgot how to use the [mv] command, what are you going to do?
5. You forgot which command is used to display the calendar, what are you going to do?
6. The [history] command is a shell builtin and so it doesn't have a man page. You want to clear your history but don't know how. What are you going to do?

True or false

1. The command [whereis] is used to locate commands.
2. You can use [man -p] and [apropos] interchangeably.
3. You can use the [whatis] command to get a brief description of a command.
4. You can use the [type] command to determine if a command is an alias, shell builtin, or an executable program.