Linux: Command and Conquer

Open Source Community ASU

Acknowledgments

First Edition

Habiba Yousri, MennaTullah Waleed, Aly Essam, Ibrahim Yasser, and Abdelrahman Khattab, members of the Linux committee, wrote and edited this manual.

Ahmed Yasser, OSC's vice president, who made this manual possible.

Mo'men Kadry and Tony Medhat also oversaw the development of this manual.

Contents

Acknowledgments i

First Edition i

1 Open Source **1**

Philosophy 1

Open Source Community ASU 2

Fun Facts 2

2 OS and Kernel **4**

Operating System (OS) 4

Resource Management 4

Kernel 5

Creator 5
GNU Project 6
Popular Linux Distributions 6

3 Introduction to Command Line 8
The Shell 8
Understanding Commands 8
Command Breakdown 9
Linux File System Hierarchy 10
'/' Directory 11
Top Level Directories 12
Navigating the System 13
Absolute Path 13
Relative Path 14
Exploring the System 16

4 Files and Directories 18
Manipulating Files and Directories 19

5 Compression and Archiving 23
Compression 23
Archiving 26

6 Getting Software **28**

Package Managers 28

What is a Package? 29

Package Contents 29

Inner Workings Overview 30

Commands 30

7 Getting Help **32**

CHAPTER 1

Open Source

Philosophy

Open Source is a software development movement that encourages open collaboration, free development, and the belief that the source code of software should be available to anyone.

This movement has improved the software industry in an unlimited way.

Open Source Community ASU

Open source community ASU is a community started in 2013 by a group of students who were passionate about the ideas of open source, Linux, and software development.

Our vision and duty are to emphasize the importance of using open source tools and developing your soft skills as a way to improve your skills for your career path.

Fun Facts

- We started with only 7 students (Geeks) in 2013, at Said Abdelwahab Hall, with laptops and banners, telling other students about Linux.
- We are the first open source community created in Egypt and have contributed to open source events like Hacktoberfest.
- We are not only a technical community; we also have committees for soft skills and design, such as:
  - Blender 3D Committee
  - UI/UX & Design Committee
  - Content Creation Committee (CCC)
- We usually hold events and workshops:
  - Get to Git' & Get to Linux' events
  - Linux Distribution Party
  - El Salakhana
  - OSC Summer Training
  - Technical Workshops: Linux, Docker, Game Development, and Web
  - Non-Technical Workshops: Blender, UI/UX & Design, CCC

- We are the community that created Employment Fair back in 2017. It was held at FCIS back then.
- We use only open-source tools as a way to emphasize the importance and capabilities of open source tools.

CHAPTER 2

OS and Kernel

Operating System (OS)

It is a system program that provides an interface between the user and the computer, as it manages the hardware and software running on it.

Resource Management

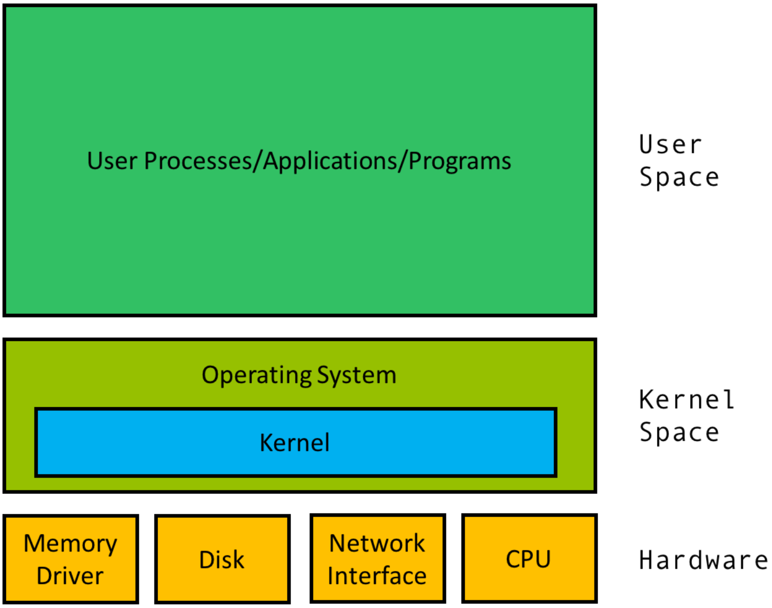
There are several different computer programs running at the same time that need access to your CPU, memory, and storage, so the OS coordinates all this to make sure that every program gets only what it needs.

Kernel

A kernel is the core component of the operating system.

It's the first program of the operating system that is loaded into the main memory and remains in memory until the system is shut down.

The kernel acts as a bridge between the applications and the hardware, as it directly communicates with the hardware.



Creator

The Linux kernel was created by Linus Torvalds, one of the greatest developers.

In 1991, Linus Torvalds, a young Finnish computer science student, released the first version of the Linux kernel, a core component of the GNU/Linux operating system. Unlike proprietary operating systems like Windows or MacOS, Linux was open source, meaning anyone could view, modify, and distribute the source code. This open model of development attracted a community of developers who contributed their time and expertise to improve the software.

Keep in mind that there is a difference between Linux as an operating system and the Linux kernel. Linus only created the latter; the former was created in collaboration with the GNU project and Linus.

GNU Project

GNU is an acronym for: GNU’s Not UNIX. The founder of this project is a software engineer and researcher named Richard Stallman.

The release of the GNU project in 1983 was marked as one of the key events in the history of open source. The project aimed to create a free Unix-like operating system, and laid the groundwork for the free software movement. Stallman's philosophy of software freedom—freedom to use, study, modify, and distribute software—inspired a new generation of developers who believed in the power of open collaboration.

Popular Linux Distributions

Intended Use Case	Distribution
Usual users	Ubuntu - Pop OS!
Developer	Fedora - Ubuntu

Intended Use Case	Distribution
Artists & Music Producers	Ubuntu Studio
Corporate Organizations	Red Hat Linux - Fedora
Cyber Security	Kali Linux

CHAPTER 3

Introduction to Command Line

The Shell

When we speak of the command line, we are really referring to the shell. The shell is a program that takes commands and passes them to the operating system to carry out. Almost all Linux distributions supply a shell program from the GNU project called `bash`.

Understanding Commands

Commands are programs that we run from the shell instead of a GUI application.

Command Breakdown

```
PROGRAM [OPTIONS] [ARGUMENT]
```

Commands usually breakdown to three parts:

- **PROGRAM**: is the program name or path in the system, examples: firefox, java, etc...
- [**OPTIONS**]: are special argument that alter the default behaviour of the program, and most of the time they are optional.
- [**ARGUMENT**]: are the arguments that the program operate on, and also most of the time they are optional.

Note: There are two different formats for *[OPTIONS]*:

- Short format: usually starts with a dash '-' followed by a single letter, example: `-a`.
- Long format: usually starts with a double dash '--' followed by a word, like `--all`.

Take for example the following commands:

`echo [OPTION] [STRING]` Displays string of text.

Option	Usage
`-n`	Do not output the trailing newline.
`-e`	Enable interpretation of backslash escapes.
`-E`	Disable interpretation of backslash escapes. (default)

```
osc@osc:~$ echo Hello World
Hello World
osc@osc:~$ echo -n "Hello, "
echo "World!"
Hello, World!
osc@osc:~$ echo -e "Hello\nWorld"
Hello
World
osc@osc:~$ echo -E "Hello\nWorld"
Hello\nWorld
```

`clear` Clears the screen.

```
osc@osc:~$ echo Hello World
Hello World
osc@osc:~$ clear
```

Output:

```
osc@osc:~$
```

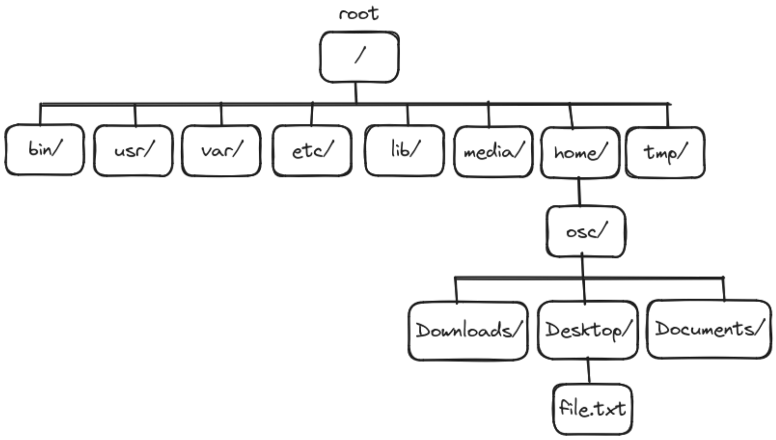
Linux File System Hierarchy

If you're new to the Linux system, you may be confused by how it references files and directories. Before exploring the Linux system, it helps to have an understanding of how it's laid out.

Linux stores files within a **single directory structure**, called a **virtual directory**. The virtual directory contains file paths from **all the storage devices installed on the computer**, merged into a **single directory** structure. The Linux virtual directory structure contains a single base directory, called the **root**. Directories and files beneath the root directory are listed based on the directory path used to get to them.

The tricky part about the Linux virtual directory is how it incorporates each storage device. The first hard drive installed in a Linux system is called the root drive (/). The root drive contains the virtual directory core. Everything else builds from there. On the root drive, **Linux can use special directories as mount points**. Mount points are directories in the virtual directory where you can **assign additional storage devices**. Linux causes files and directories to appear within these mount point directories, even though they are physically stored on a different drive.

Often system files are physically stored on the root drive. User files are typically stored on a separate drive or drives, as shown below.



'/' Directory

The '/' directory or the ”root” directory is where everything begins on Linux.

No matter what you want to access, where it is, it will somehow connect to the root directory.

The '`/`' character is also used as a directory separator in file names. For example, if `etc` is a subdirectory of the `/` directory, you could refer to that directory as `/etc`. Likewise, if the `/etc` directory contained a file named `issue`, you could refer to that file as `/etc/issue`.

Top Level Directories

Directory	Content / Description
`/`	The root of the virtual directory. It is the **starting point** for the file system hierarchy
`/boot`	Boot directory, where **boot files** are stored (e.g, Linux kernel and other static files of the boot loader).
`/media`	Media directory, a common place for mount points used for **removable media**.
`/mnt`	Mount directory, another common place for mount points used for **removable media**.
`/etc`	**System configuration** files directory.
`/bin`	Binary directory, where many **essential user command binaries** are stored.
`/sbin`	System binary directory, where many **system administration binaries** are stored.
`/usr`	User binary directory, where the **applications and files used by users** are stored (/usr/ is the second major section of the filesystem (secondary hierarchy).
`/tmp`	Temporary directory, where **temporary work files** can be created and destroyed (these temporary files are generally deleted when the system is restarted).

Directory	Content / Description
`/var`	Variable directory, for **files that change frequently** which handled by services, such as logs, queues, caches, and spools.
`/home`	Home directory, where Linux creates **normal user** directories (non-root users).
`/root`	The home directory for the **root user** (administrative superuser).

Navigating the System

`pwd` Prints the absolute path of the current working directory.

```
osc@osc:~/Downloads$ pwd
/home/osc/Downloads
```

`cd [DIRECTORY]` Change directory.

To change our working directory, we use the `cd` command.

```
osc@osc:~$ cd /home/osc/Downloads
osc@osc:~/Downloads$
```

Now , our working directory here is `Downloads`.

Absolute Path

An absolute path begins with the root directory '`/`' and follows the tree branch by branch until the path to the desired directory or file is completed.

For example, there is a directory on our system in which most of our system's programs are installed. The directory's path is ”`/usr/bin`”. This means from the root directory (represented by the leading slash '`/`' in the path), there is a directory called ”usr”, which contains a directory called ”bin”.

```
osc@osc:~$ cd /usr/bin
osc@osc:/usr/bin$ pwd
/usr/bin
osc@osc:/usr/bin$
```

Relative Path

Where an absolute path starts from the root directory '`/`' and leads to its destination, a relative path starts from the working directory.

The '`.`' notation refers to the working directory, and the '`..`' notation refers to the working directory's parent directory.

The working directory here is `usr`.

```
osc@osc:/usr/$ cd ./bin
osc@osc:/usr/bin$ pwd
/usr/bin
```

In almost all cases, we can omit the `./`.

```
osc@osc:/usr/$ cd bin
osc@osc:/usr/bin$ pwd
/usr/bin
```

Command	Usage
`cd ..`	Change the current working directory to the parent directory of the current directory.
`cd`	Change the working directory to your `home` directory.
`cd -`	Changes the working directory to the previous working directory.
`cd ~user_name`	Changes the working directory to the `home` directory of user_name.

`ls [OPTION] [FILE]` Lists directory contents.

```
osc@osc:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public
    Templates  Videos
```

Besides the current working directory, we can specify the directory to list:

```
osc@osc:~$ ls /home
osc
```

Option	Usage
`-l`	Displays detailed information about files.
`-a`	List all files, including the hidden files.

Option	Usage
`-t`	Sort files by modification time, with the newest files appearing first.
`-r`	Reverse the order of the sort to display files in reverse order.
`-S`	Sort files by size, with the largest files appearing first.

Exploring the System

`cat [OPTION] [FILE]` Display contents of a file, concatenate files and print on the standard output.

```
osc@osc:~/Documents$ cat myFile
hello world
```

Option	Usage
`-n`	Number the lines of the output.
`-s`	Suppress repeated empty output lines.
`-E`	Display a dollar sign ($) at the end of each line.

`file` Determine file type.

```
osc@osc:~/Documents$ file myFile
myFile: ASCII text
```

`type` It is used to find out whether it is a built-in or external binary file.

```
osc@osc:~$ type cd
cd is a shell builtin
```

CHAPTER 4

Files and Directories

In Linux, most of the operations are performed on files. And to handle these files, Linux has directories, also known as folders, which are maintained in a tree-like structure. Though these directories are also a type of file.

Linux has three types of files:

- **Regular Files:** It is the most common file type in Linux. It includes files like – text files, images, binary files, etc. Such files can be created using the `touch` command. They consist of the majority of files in the Linux/UNIX system. The regular file contains ASCII, or Human-readable text, executable program binaries, program data, and much more.

- **Directories:** These are the files that store the list of file names and the related information. The root directory `/` is the base of the system, `/home/` is the default location for the user's home directories, `/bin` is for Essential User Binaries, `/boot` is for static boot files, etc. We could create new directories with `mkdir` command.
- **Special Files:** Represents a real physical device, such as a printer, which is used for IO operations. Devices or special files are used for device Input/Output on UNIX and Linux systems. You can see them in a file system like an ordinary directory or file.

Manipulating Files and Directories

`touch FILE` Creates a new file;
if the file exists, it updates the timestamp.

To Create a new files `file1.txt`, `file2.txt`, `file3.txt`:

```
osc@osc:~$ touch file1.txt file2.txt file3.txt
osc@osc:~$ ls
file1.txt file2.txt file3.txt
```

`mkdir [OPTION] DIRECTORY` Creates a new directory,
provided it doesn't exists.

You can create a new directory whose name is `Test`:

```
osc@osc:~$ mkdir Test
osc@osc:~$ ls
Test
```

`mkdir -p Directory/SubDirectory1/SubDirectory2` Creates nested directories.

You can create nested directories, `world`, which is inside the `hello`, which is inside the `program`:

```
osc@osc:~$ mkdir -p program/hello/world
osc@osc:~$ ls
program
osc@osc:~$ ls program/
hello
osc@osc:~$ ls program/hello
world
```

`mv [OPTION] SOURCE DESTINATION` Moves a file to a new location.

Move `file1` into the `Test` directory:

```
osc@osc:~$ ls
Test file1
osc@osc:~$ mv file1 Test
osc@osc:~$ ls
Test
osc@osc:~$ ls Test
file1
```

You can move that file back to your home directory by using the **dot reference** '.' to refer to the current directory. Make sure you're in your `home` directory, and then run the `mv` command:

```
osc@osc:~$ mv test/file1 .
osc@osc:~$ ls
Test file1
```

Moving and renaming are both just adjusting the location and name for an existing item.

You can rename `file1` into `file2`:

```
osc@osc:~$ ls
file1
osc@osc:~$ mv file1 file2
osc@osc:~$ ls
file2
```

`cp [OPTION] SOURCE DESTINATION` Makes a new copy of an existing item.

Copies `source.txt` contents to a new file called `copy.txt`.

```
osc@osc:~$ ls
source.txt
osc@osc:~$ cat source.txt
I am a text file!
osc@osc:~$ cp source.txt copy.txt
osc@osc:~$ ls
source.txt copy.txt
osc@osc:~$ cat copy.txt
I am a text file!
```

`cp -r SOURCE_DIRECTORY DESTINATION` Copies entire directories.

Note: You must include the `-r` option to the command. This stands for "recursive", as it copies the directory, plus all of the directory's contents.

Copies the `test1` directory structure to a new structure called `test2`.

```
osc@osc:~$ ls
test1
osc@osc:~$ cp -r test1 test2
osc@osc:~$ ls
test1 test2
```

`rm [OPTION] FILE` Deletes a file.

To remove `test` file from the directory:

```
osc@osc:~$ ls
test
osc@osc:~$ rm test
osc@osc:~$ ls
```

`rmdir [OPTIONS] DIRECTORY` Removes empty directories.

```
osc@osc:~$ ls test/
example
osc@osc:~$ rmdir test/example
osc@osc:~$ ls test/
```

`rm -r DIRECTORY` Removes a **non-empty** directory.

To remove the `test` directory and everything within it:

```
osc@osc:~$ ls
test
osc@osc:~$ rm -r test
osc@osc:~$ ls
```

Option	Description
`-f`	Deletes by force and doesn't prompt the user.
`-r DIRECTORY`	Deletes a non-empty directory.
`-d DIRECTORY`	Deletes an empty directory.

CHAPTER 5

Compression and Archiving

Compression

Data compression is the process of encoding or modifying data using functions or an algorithm to reduce its size. Fundamentally, it involves re-encoding information using fewer bits than the original representation.

Data compression is important for reducing file size so that files can be transferred easily and use less space when stored.

Files in Linux can be compressed or decompressed using the `gzip` tool.

`gzip FILENAME` Compresses a file.

To compress a file named `file.txt` using `gzip`:

Note: You can use `ls -lh` to monitor file size before and after compression.

```
osc@osc:-$ ls -lh
total 4.0k
-rw-r--r--. 1 osc osc 4.0K Mar 30 16:47 file.txt
osc@osc:-$ gzip file.txt
osc@osc:-$ ls -lh
total 4.0K
-rw-r--r--. 1 osc osc 55 Mar 30 16:47 file.txt.gz
```

The file `file.txt` is now compressed and replaced with the compressed file `file.txt.gz`. The new compressed file now occupies 55 bytes only instead of 4 kilobytes.

To extract or decompress back `file.txt` from `file.txt.gz`:

```
osc@osc:-$ gunzip file.txt.gz
osc@osc:-$ ls -lh
total 4.0k
-rw-r--r--. 1 osc osc 4.0K Mar 30 16:47 file.txt
```

Note: You can decompress files compressed by `gzip` by adding the flag `-d` to the command, so it becomes `gzip -d file.txt.gz`.

There are plenty of flags that can be used, and there are some of them:

Option	Usage
`-k`	Keeps the original file and makes a new file for the compressed one.

Option	Usage
`-l`	For compressed files it show: compressed size, uncompressed size, and the ratio between them.
`-r`	Used in directories to recursively compress all files inside them.
`-1 - -9`	Is used to regulate the speed and compression level, where `-1` is the fastest (less compression) and `-9` indicates the slowest (best compression). *default compression level is `-6`.*

To compress all files within a directory:

```
osc@osc:-$ ls -lh dir/
total 12K
-rw-r--r--. 1 osc osc 4.0K Mar 30 17:00 file1.txt
-rw-r--r--. 1 osc osc 4.0K Mar 30 17:00 file2.txt
-rw-r--r--. 1 osc osc 4.0K Mar 30 17:00 file3.txt
osc@osc:-$ gzip -r dir
osc@osc:-$ ls -lh dir/
total 12K
-rw-r--r--. 1 ocs ocs 56 Mar 30 17:00 file1.txt.gz
-rw-r--r--. 1 ocs ocs 56 Mar 30 17:00 file2.txt.gz
-rw-r--r--. 1 ocs ocs 56 Mar 30 17:00 file3.txt.gz
```

Note: `gzip -r` compresses every file in the directory individually.

Archiving

Archiving is the process of bundling multiple files or directories into a single file for future reference. Archiving directories can help compress them and save storage.

An **archive file** is a collection of files and directories that are stored in one file. **The archive file is not compressed** — it uses the same amount of disk space as all the individual files and directories combined.

`tar MODE [OPTION] -f ARCHIVE_NAME FILE...` Archives files or directories.

Mode	Usage
`-c`	Create a new archive.
`-x`	Extract files from an archive.
`-r`	Append specified pathanmes to the end of an archive.
`-t, --list`	List the contents of an archive.

Option	Usage
`-f`	Mandatory flag to create an archive with a specific name.
`-u`	Add a file to an existing archive.
`-z`	Compress the archive file using gzip.

NOTE: The mode must be specified before any other options. example: `tar -fc` this will cause error, the correct way: `tar -cf`.

To create a tar archived file from `dir/`:

```
osc@osc:-$ ls -lh dir/
total 12K
-rw-r--r--. 1 osc osc 4.0K Mar 30 17:00 file1.txt
-rw-r--r--. 1 osc osc 4.0K Mar 30 17:00 file2.txt
-rw-r--r--. 1 osc osc 4.0K Mar 30 17:00 file3.txt
osc@osc:-$ tar -cf dir.tar dir/
osc@osc:-$ ls -lh
total 24K
drwxr-xr-x. 2 tonym tonym 4.0K Mar 30 17:17 dir
-rw-r--r--. 1 tonym tonym  20K Mar 30 17:18 dir.tar
```

Now that you created the archive file, then you can compress it using `gzip`.

To unarchive `dir.tar`:

```
osc@osc:-$ tar -xf dir.tar
osc@osc:-$ ls -lh dir/
total 12K
-rw-r--r--. 1 osc osc 4.0K Mar 30 17:00 file1.txt
-rw-r--r--. 1 osc osc 4.0K Mar 30 17:00 file2.txt
-rw-r--r--. 1 osc osc 4.0K Mar 30 17:00 file3.txt
```

Note: `.tar` suffix is preferred, but it is not mandatory or used by `tar`.

CHAPTER 6

Getting Software

Package Managers

A package manager in Linux is a tool or a set of tools designed to simplify the process of installing, updating, configuring, and removing software packages on a Linux system. Linux distributions use different package managers, and the choice of package manager often depends on the distribution's specific package management system.

What is a Package?

A package refers to a compressed archive file containing software, an application, or a set of related files along with metadata. These packages are used for easy distribution, installation, and management of software on Linux-based systems. Different Linux distributions have their own package management systems, and the package format may vary among them.

The two most common package management systems are:

- **RPM (Red Hat Package Manager)**: RPM is used by Red Hat-based distributions such as Red Hat Enterprise Linux (RHEL), Fedora, CentOS, and others.
- **DPKG (Debian Package)**: DPKG is used by Debian-based distributions such as Debian, Ubuntu, and their derivatives.

In addition to these package management systems, many Linux distributions also use package managers such as `YUM`, `APT`, `Zypper`, and others to handle dependencies, updates, and installations.

Package Contents

The contents of an `RPM` package and `DPKG` package typically include:

- **Binary executables:** Precompiled program files that can be run on the target system.
- **Libraries:** Shared libraries required by the software.
- **Configuration files:** Settings and configurations specific to the installed software.
- **Documentation:** Information and manuals related to the software.

- **License information:** Details about the software's licensing terms.
- **pre-installation, post-installation scripts:** Custom scripts executed before and after installation to perform additional tasks.

Inner Workings Overview

The general working principles of a package manager can vary slightly depending on the specific operating system, but the core concepts are similar. Here's a general overview of how a package manager works:

- **Package Repository:**
  - A package manager relies on a centralized repository or multiple repositories where software packages are stored.
  - Each package in the repository contains the application or library, along with metadata like version information, dependencies, and configuration files.
- **Package Metadata:**
  - Metadata provides information about the software package, including its name, version number, dependencies, and a brief description.
  - Dependencies are other software packages that must be installed for the current package to work properly.

Commands

The `apt`, and 'dnf' commands are package management tools used in Debian-based, and Red Hat-based Linux distributions respectively, such as Ubuntu, Fedora, and CentOS.

The two previous commands allow users to install, update, upgrade, and manage software packages on their system.

Here are some commonly used `dnf` and `apt` commands:

Description	Debian	RPM
Search for package	`apt search PACKAGE ...`	`dnf search PACKAGE...`
View package details	`apt show PACKAGE...`	`dnf info PACKAGE...`
Install a package	`apt install PACKAGE ...`	`dnf install PACKAGE...`
Reinstall packages	`apt reinstall PACKAGE...`	`dnf reinstall PACKAGE...`
Remove packages	`apt remove PACKAGE ...`	`dnf remove PACKAGE...`
Upgrade a package	`apt upgrade PACKAGE ...`	`dnf upgrade PACKAGE...`
List installed packages	`apt list --installed`	`dnf list installed`
Upgrade system packages	`apt upgrade`	`dnf upgrade`

CHAPTER 7

Getting Help

`man [OPTIONS] [COMMAND]` Displays the manual pages for various commands and utilities.

It provides information about the command's syntax, options, usage, and often includes examples.

So let's view the manual page for the `ls` command:

```
osc@osc:~$ man ls
LS(1)    User Commands    LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...
```

```
DESCRIPTION
       List information about the FILEs (the current
    directory by default).  Sort entries alphabetically if
    none of -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for
    short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..
...
```

The following conventions apply to the SYNOPSIS section and can be used as a guide in other sections.

Convention	Usage
bold text	Type exactly as shown.
italic text	Replace with appropriate argument.
[-abc]	Any or all arguments within [ ] are optional.
-a|-b	Options delimited by | cannot be used together.
argument ...	Argument is repeatable.
[expression] ...	Entire expression within [ ] is repeatable.

`apropos` Lists several commands that match the keyword you used.

The list includes a short description of what the command does, works by looking through the entire description sections of the man pages for the matching keyword you provide with the apropos command.

The word apropos is derived from the French word ”à propos” which means ”about”.

Let's assume you want to copy a file but do not know which command to use. Simply use the apropos command followed by the task you want to complete.

```
osc@osc:~$ apropos copy
copy_file_range (2)  - Copy a range of data from one file to
    another
copysign (3)         - copy sign of a number
copysignf (3)        - copy sign of a number
copysignl (3)        - copy sign of a number
cp (1)               - copy files and directories
...
```

`--help` Displays helpful information about how a command is used and its arguments in a simplified manner.

Note: Most commands have the `--help` command argument or option.

Get more help on the `cp` command by typing:

```
osc@osc:~$ cp --help
Usage: cp [OPTION]... [-T] SOURCE DEST
  or:  cp [OPTION]... SOURCE... DIRECTORY
  or:  cp [OPTION]... -t DIRECTORY SOURCE...
Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

Mandatory arguments to long options are mandatory for short
    options too.
  -a, --archive                 same as -dR --preserve=all
      --attributes-only         don't copy the file data, just
     the attributes
      --backup[=CONTROL]        make a backup of each existing
     destination file
  -b                            like --backup but does not
    accept an argument
      --copy-contents           copy contents of special files
     when recursive
...
```