Linux Commands Guide

Written by Mayakkanann Subhas (MK)

Introduction

In this document, we will group documents by functionality and provide an one-liner explanation to each command. This document serves as a reference and we will look at examples for each command in another hands-on guide.

> This document is a simple list of commands with brief explanation. Refer the manual pages of the corresponding commands using `man CMD` from the shell or online documentation for more information

---

Types of Commands

In this document, we have classified the commands based on the broader functionality these commands provide. This classification is based on the author's experience with the Linux system as a developer and data analyst. We cover general-purpose commands here with very little system admin commands

> Some commands may fit into multiple categories.

1. Basic commands
2. Files and Directories Access
3. File Permission
4. Viewing Files / Directories
5. Manipulating Files
6. Process and Job Management
7. Text Processing
8. Archival processing
9. Accessing files from remote servers
10. Miscellaneous commands

1. Basic Commands

This section covers the basic commands that we will encounter when we start learning the command line interface. These commands are there to get information about the system such as current date, current working directory, clear screen etc..

> These commands gives us a good start to get ourselves comfortable using the shell / CLI

#	Name	Description
1	`echo <STR>`	print the string on the screen
2	`pwd`	print current working directory
3	`date`	print current date and time
4	`clear`	clear the screen, use the `CTRL-L` key as alternate
5	`sleep <N>`	sleeps for **N** seconds
6	`cal`	displays current month's calendar in tabular format.
7	`man CMD`	displays help text for the **CMD** passed as argument.
8	`history`	display the list of commands ran so far
9	`passwd`	change password
10	`seq`	create number sequence
11	`exit`	terminate session

2. Files and Directories

This section covers commands that are used to create files and access the files. Unix treats everything as files, including hardware devices, network sockets etc.. We will deal with three types of files; **regular files** such as text and binary files, **directories** that are containers that can hold other files and **links** that are shortcuts to actual files and directories

> This set of commands are to create and access files and directories. Next sections have more commands to view and manipulate files and directories

#	Name	Description
1	`mkdir`	create directory
2	`cd`	change directory
3	`rmdir`	remove empty directories, wont work on directories with contents
4	`touch`	create an empty file if it doesn't exists
5	`ln`	create link to an existing file; can be **hard** or **soft** link
6	`ls`	list files and directories
7	`tree`	list a directory and its contents in a tree like format
8	`cp`	copy files and directories
9	`mv`	rename or move files and directories
10	`rm`	remove files and directories with contents
11	`unlink`	remove a file. a simple version of `rm` to remove one file
12	`du`	display the disk usage info of a file or a directory and its contents
13	`df`	display the free and used space of disks attached to the system

3. File Permissions

Everything in Unix is treated as files and every file has explicit permissions associated with it. The permissions determine who can have access to these files and what type of access they have. There are two types of access; **Authentication** is a way to gain access into the system using credentials and **Authorization** is how the system determines who have access to what.

Permissions can be explicitly given when we create directories using `mkdir -m <MODE>` whre **MODE** is an octal number representing the permissions. Permissions are given to various **actors** in the system and files can have different **ac-**

cess associated with them. The owner or the super user can perform various **actions** in terms of providing permissions to various *actors*.

Actors

Name	Abbr	Description
User	**u**	owner of the file
Group	**g**	default group the owner belong to
Others	**o**	users that are not part of the group
All	**a**	every user in the system

Actions

Name	Abbr	Description
Add	**+**	add permission
Remove	**-**	remove permission
Assign	**=**	discard existing permissions and assign new permissions

Access

Name	Abbr	Description
Read	**r**	reading from files / viewing contents of directories
Write	**w**	writing into files / create, delete files and directories
Execute	**x**	execute files (code) / view or modify metadata of files

Permissions can be assigned to **u**ser, default **g**roup and **o**thers as combination the above thre acceses; **r**ead, **w**rite and e**x**ecute. The combinations ranging from no permission to all permissions. There are 8 combinations dervived from these three type of access that can be represented in **Octal** notation. Numbers 4, 2 and 1 are assigned to **r**ead, **w**rite and e**x**ecute and the combinations of these access gets the other values 0, 3, 5, 6 and 7.

Num	Permission	Description
`0`	`---`	no permission
`1`	`--x`	execute only
`2`	`-w-`	write only

Num	Permission	Description
3	`-wx`	write and execute
4	`r--`	read only
5	`r-x`	read and execute
6	`rw-`	read and write
7	`rwx`	read, write and execute

> Permissions can be given using the `chmod` command either by using **actors, actions and access** notation or using **octal** notation

View / Set permission and ownership of files

#	Name	Description
1	`umask`	view or set default permission
2	`chmod`	change permisison on files
3	`chown`	change owner of the file
4	`chgrp`	change group name associated with the file
5	`ls`	view permission info using `ls -l` option
6	`stat`	view perimssions and other file status such as size, date of creation, ...
7	`id`	get info about an user; uid, default group and other groups an user is associated with
8	`groups`	get group info of an user

The below commands will be used by SysAdmins, These are listed here for completeness

#	Name	Description
1	`useradd`	create new user id
2	`adduser`	same as `useradd`, some installation have this one
3	`usermod`	modify an existing user info
4	`userdel`	delete user
5	`groupadd`	creating new group
6	`groupmod`	modify an existing group info

#	Name	Description
7	`groupdel`	delete group

4. Viewing Files

In the previous section, we have looked into commands that are used to create files and directories and once created how to access and manipulate those files. In this section we will discuss various way in which we can view the files; entire file, part of the file, view page by page,...

#	Name	Description
1	`cat`	view file(s), displayed on the screen
2	`tac`	view file(s), records displayed in reverse order; last record first **LIFO**. May not be available in some OS versions
3	`rev`	view records in **reverse** order, like `cat` but each record is displayed in reverse order
4	`nl`	like `cat`, adds line number as prefix
5	`head`	display first 10 lines of file(s) by default
6	`tail`	display last 10 lines of file(s) by default
7	`less`	display contents of a file, one line at a time. use `spacebar` and `b` to page down and page up and `q` to quit display
8	`more`	like the `less` command; older version and limited navigation features
9	`od`	display files as ascii, octal, hexadecimal dump, useful in analyzing binary files
10	`wc`	display number of lines, words and characters of the file(s)

5. Manipulating Files

This section deals with commands that can manipulate the contents of files; actions such as **sort**, **slice**, **split**, **merge**,...

#	Name	Description

#	Name	Description
1	`cut`	create slices from each record
2	`paste`	merge multiple lines from a file into single line or merge multiple files line by line
3	`split`	split a file into smaller chunks. By default, each split file contains 1000 lines, we can use options to split by different line count or split by bytes size
4	`join`	merge already **sorted** files by keys, By default it performs an **inner** or **equality** join, only display records with matching keys
5	`sort`	sort files, a rich set of options are available
6	`uniq`	create unique records from an already **sorted** file
7	`diff`	compare two **sorted** files and display the records that are different
8	`column`	display file in tabular format
9	`comm`	compare two **sorted** files and display the unique records from file 1, file 2 and matched records in 3 columns. This command is useful to compare files with shorter records
10	`cmp`	compare two files byte by byte and display summary, from which byte/line there is a difference. We can limit the number of bytes to be compared

6. Process Management

Unix is a **multiuser** and **multitasking** operating system. Processes form the core of the system and they form the *logical unit* to manage resources needed for each processes. A **process** is a running instance of a command or a program. Each process needs computing resources such as CPU, memory. When a process is initiated, Unix assigns an unique number called **process id** also called as **PID**. We can query the status of running processes and if we specifically want get status of a process, we can use the **PID** to access it.

The Unix family of OS uses a `fork()` and `exec()` model to spawn a new process. The processes in Unix forms a tree-like structre. When the system boots up, it launches a process called `init` that gets the **PID 1** and from `init` other processes are created. Like the `root` directory in the file system tree structure,

`init` forms the entry to the proccess tree structre. In addition to the **PID**, the process also has **PPID**; *parent process id* that can be used to trace the process hierarchy as needed.

#	Name	Description
1	`ps`	print process status information
2	`jobs`	prints the status of the background processes
3	`bg`	convert a foreground process into background process. Use `^Z` to suspend process first
4	`fg`	convert the background process to foreground process.
5	`kill`	forcefully terminate a running job using the **PID**
6	`kil-lall`	like `kill` but uses command name. may terminate more than one job
7	`nohup`	submit commands in **no hangup** mode, job continues to run even if session terminates
8	`dis-own`	similar to `nohup`, we simply `disown` a running job so that it continues to run even if session terminates
9	`top`	view process information in 5 seconds interval. stop command using `^C`
10	`free`	displays available, used and free main memory; *RAM*

Job Schedule We can schedule jobs to run in the future; either once or repeatedly. Unix provides a simple yet elegent syntax to schedule the jobs using combination of minutes, hours, day, month and day of the week.

#	Name	Description
1	`crontab`	schedule jobs
2	`at`	schedule one time jobs
3	`atq`	list jobs scheduled by `at`
4	`atrm`	remove job(s) scheduled by `at`

view crontab examples at https://crontab.guru/

7. Text Processing

Unix is very good with process text information. The support for **Regular Expressions** also called as **RegEX** makes operations like filter, search, replace easier. In addition to these operations, we have commands to validate, tranform, translate and aggregate information.

#	Name	Description
1	`tr`	translate input stream
2	`grep`	global regular expression and print. search files for patterns and text; supports *Basic RegEX (BRE)*
3	`egrep`	extended grep, supports *Extended RegEX (ERE)* syntax. we can also use `grep -E` instead
4	`fgrep`	fixed grep, search literals, faster than using grep as it doesnt have to validate search string as RegEX. we can also use `grep -F` instead
5	`sed`	**stream editor**, search and replace text. A predecessor of the **Vi** editor
6	`awk`	a programming language that can be used to write code snippets directly on the command line

8. Archival Process

This section covers the commands that can be used to compress files to save space and archive multiple files as a single unit. There are commands that can be used to view contents of the files without explicitly uncompressing the files

#	Name	Description
1	`zip`	compress one or more files, ends with `.zip` extension
2	`unzip`	umcompress one or more files created by the `zip` command
3	`zip-info`	get info about the contents of the **zip archive**
4	`gzip`	compresses a single file using *LZ77* coding
5	`gunzip`	uncompress the file created by the `gzip` command
6	`tar`	create archive files with or without compression

#	Name	Description
7	`gzcat`	displays content of a `.gz` file without compressing
8	`zgrep`	`grep` on `.gz` file; similar to `grep -Z`
9	`zegrep`	`egrep` on `.gz` file; similar to `grep -ZE`
10	`zfgrep`	`fgrep` on `.gz` file; similar to `grep -ZE`
11	`zip-grep`	`grep` on `.zip` file

9. Accessing Remote Servers

#	Name	Description
1	`ssh`	login to remote server
2	`scp`	secure copy from / to local machine and remote server or between two remote servers
3	`sftp`	secure FTP; *File Transfer Protocol'
4	`rsync`	syncronize the contents of directories between local and remote server

10. Miscellaneous Commands

#	Name	Description
1	`bc`	calculator
2	`dc`	calculator; expressions in reverse polish notation
3	`alias`	create shortcut to commands or display existing aliases
4	`unalias`	remove an existing alias
5	`uname`	get operating system info; name, processor architecture,...
7	`which`	
8	`whatis`	
9	`whereis`	
10	`users`	get the list of current users.
11	`who`	display all the users who have logged in

#	Name	Description
12	`uptime`	shows how long the system has been running

11. Advanced Commands

#	Name	Description
1	`find`	search for files recursively and perform operations on the results
2	`xargs`	build arguments from standard input or redirected output and pass it to a command