25 Essential Linux Commands That You Should Know :

TABLE OF CONTENTS

Linux is a family of open-source Unix operating systems based on the Linux Kernel. They include Ubuntu, Fedora, Debian, openSUSE, and Red Hat. Using Linux to manage a **Virtual Private Server** (VPS) is common practice.

However, we recommend utilizing the command-line interface (CLI) because it's quicker and offers more control. Tasks that require multiple

steps on the GUI can be done in a matter of seconds by entering commands into the CLI.

By the end of this article, we will be able to learn most commonly command that are being used in Linux :

Firstly, What is a linux command ?

A Linux command is a program or utility that runs on the CLI – a console that interacts with the system via texts and processes. It's similar to the Command Prompt application in Windows.

Syntax for a Linux Command :

Here's what a Linux command's general syntax looks like:

CommandName [option(s)] [parameter(s)]

A command may contain an option or a parameter. In some cases, it can still run without them. These are the three most common parts of a command:

- **CommandName** is the rule that you want to perform.
- **Option** or **flag** modifies a command's operation. To invoke it, use hyphens (**–**) or double hyphens (**—**).
- **Parameter** or **argument** specifies any necessary information for the command.

NOTE : Linux commands are case-sensitive.

Here is the list of basic Linux commands:

1. sudo command

Short for superuser do**, sudo** is one of the most popular basic Linux commands that lets you perform tasks that require administrative or root permissions.

Here's the general syntax: **sudo (command**)

2. pwd command

Use the **pwd** command to find the path of your current working directory. Simply entering **pwd** will return the full current path – a path of all the directories that starts with a forward slash (**/**). For example, **/home/username**.

The **pwd** command uses the following syntax: **pwd**

3. cd command

To navigate through the Linux files and directories, use the **cd** command. Depending on your current working directory, it requires either the full path or the directory name.

- **cd ..** moves one directory up.
- **cd-** moves to your previous directory

4. ls command

The **ls** command lists files and directories within a system. Running it without a flag or parameter will show the current working directory's content.

To see other directories' content, type **ls** followed by the desired path. For example, to view files in the **Documents** folder, enter:

ls /home/username/Documents

5. cat command

Concatenate, or **cat**, is one of the most frequently used Linux commands. It lists, combines, and writes file content to the standard output. To run the cat command, type **cat** followed by the file name and its extension. For instance:

cat filename.txt.

6. cp command

Use the **cp** command to copy files or directories and their content. Take a look at the following use cases.

To copy one file from the current directory to another, enter **cp** followed by the file name and the destination directory. For example:

cp filename.txt /home/username/Documents

7. mv command

The primary use of the **mv command** is to move and rename files and directories. Additionally, it doesn't produce an output upon execution.

Simply type **mv** followed by the filename and the destination directory. For example, you want to move **filename.txt** to the **/home/username/Documents** directory:

mv filename.txt /home/username/Documents.

*You can also use the **mv** command to rename a file:*

mv old_filename.txt new_filename.txt

8. mkdir command

Use the **mkdir** command to create one or multiple directories at once and set permissions for each of them. The user executing this command must have the privilege to make a new folder in the parent directory, or they may receive a permission denied error.

Here's the basic syntax:

mkdir [option] directory_name

For example, you want to create a directory called **Music**:

mkdir Music

9. rmdir command

To permanently delete an empty directory, use the **rmdir command**. Remember that the user running this command should have **sudo** privileges in the parent directory.

For example, you want to remove an empty subdirectory named **personal1** and its main folder **mydir**:

rmdir -p mydir/personal1

10. rm command

The **rm** command is used to delete files within a directory. Make sure that the user performing this command has write permissions.

Remember the directory's location as this will remove the file(s) and you can't undo it.

Here's the general syntax:

rm filename

To remove multiple files, enter the following command:

rm filename1 filename2 filename3

11. touch command

The **touch command** allows you to create an empty file or generate and modify a timestamp in the Linux command line.

For example, enter the following command to create an HTML file named **Web** in the **Documents** directory:

touch /home/username/Documents/Web.html

12. locate command

The **locate command** can find a file in the database system.

Moreover, adding the **-i** argument will turn off case sensitivity, so you can search for a file even if you don't remember its exact name.

To look for content that contains two or more words, use an asterisk (*****). For example:

locate -i school*note

The command will search for files that contain the words **school** and **note**, whether they use uppercase or lowercase letters.

13. find command

Use the **find** command to search for files within a specific directory and perform subsequent operations. Here's the general syntax:

find [option] [path] [expression]

For example, you want to look for a file called **notes.txt** within the **home** directory and its subfolders:

find /home -name notes.txt

14. grep command

Another basic Linux command on the list is **grep** or global regular expression print. It lets you find a word by searching through all the texts in a specific file.

Once the **grep command** finds a match, it prints all lines that contain the specific pattern. This command helps filter through large log files.

For example, you want to search for the word **blue** in the **notepad.txt** file:

grep blue notepad.txt

The command's output will display lines that contain **blue**.

15. df command

Use the **df command** to report the system's disk space usage, shown in percentage and kilobyte (KB). Here's the general syntax:

df [options] [file]

For example, enter the following command if you want to see the current directory's system disk space usage in a human-readable format:

df -h

16. du command

If you want to check how much space a file or a directory takes up, use the **du** command. You can run this command to identify which part of the system uses the storage excessively.

Remember, you must specify the directory path when using the **du** command. For example, to check **/home/user/Documents** enter:

du /home/user/Documents

17. head command

The **head** command allows you to view the first ten lines of a text. Adding an option lets you change the number of lines shown. The **head** command is also used to output piped data to the CLI.

Here's the general syntax:

head [option] [file]

18. tail command

The **tail command** displays the last ten lines of a file. It allows users to check whether a file has new data or to read error messages.

Here's the general format:

tail [option] [file]

19. diff command

Short for difference, the **diff** command compares two contents of a file line by line. After analyzing them, it will display the parts that do not match.

Programmers often use the **diff** command to alter a program instead of rewriting the entire source code.

Here's the general format:

diff [option] file1 file2

20. tar command

The **tar command** archives multiple files into a **TAR** file – a common Linux format similar to **ZIP**, with optional compression.

Here's the basic syntax:

tar [options] [archive_file] [file or directory to be archived]

21. chmod command

chmod is a common command that modifies a file or directory's read, write, and execute permissions. In Linux, each file is associated with three user classes – **owner**, **group member**, and **others**.

Here's the basic syntax:

chmod [option] [permission] [file_name]

22. chown command

The **chown command** lets you change the ownership of a file, directory, or symbolic link to a specified username.

Here's the basic format:

chown [option] owner[:group] file(s)

For example, you want to make **linuxuser2** the owner of **filename.txt**:

chown linuxuser2 filename.txt

23. jobs command

A job is a process that the shell starts. The **jobs** command will display all the running processes along with their statuses.

Remember that this command is only available in **csh**, **bash**, **tcsh**, and **ksh** shells.

This is the basic syntax:

jobs [options] jobID

24. kill command

Use the **kill command** to terminate an unresponsive program manually. It will signal misbehaving applications and instruct them to close their processes.

To kill a program, you must know its process identification number (PID). If you don't know the PID, run the following command:

ps ux

After knowing what signal to use and the program's PID, enter the following syntax:

kill [signal_option] pid

25. ping command

The **ping command** is one of the most used basic Linux commands for checking whether a network or a server is reachable. In addition, it is used to troubleshoot various connectivity issues.

Here's the general format:

ping [option] [hostname_or_IP_address]

For example, you want to know whether you can connect to **Google** and measure its response time:

ping google.com

WRITTEN BY

Biswaraj Sahoo

— AWS Community Builder | DevOps Engineer | Docker | Linux | Jenkins | AWS | Git | Terraform | Docker | kubernetes

Empowering communities via open source and education.