OCTOBER 18, 2022 / #COMMAND LINE

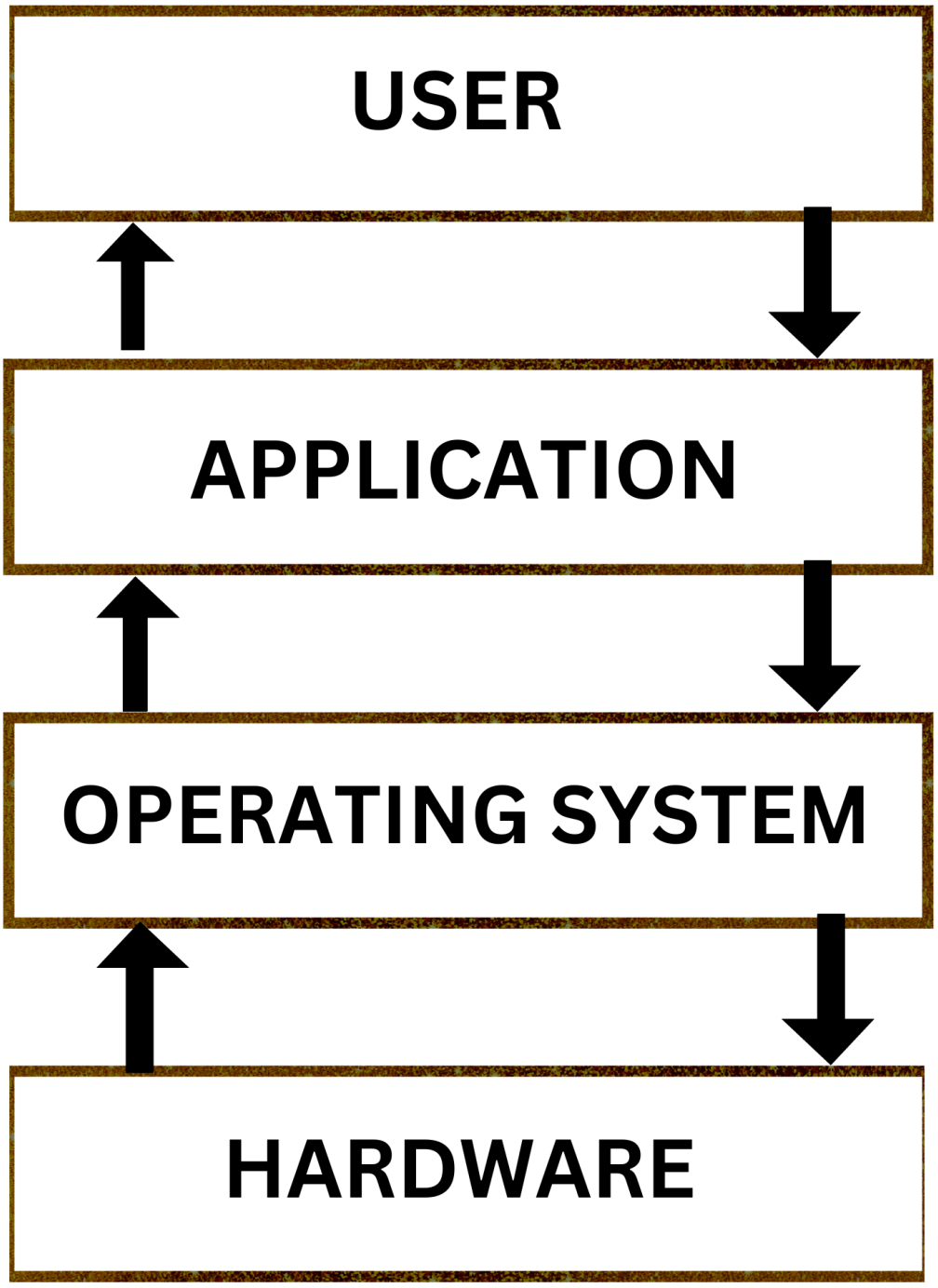
Linux Command Line Tutorial – How to Use Common Terminal Commands

Destiny Erhabor

you and your computer's hardware.

The operating system (or OS) is a piece of software that controls all other application programs and helps you manage the hardware and software of your computer.

Examples of popular operating systems are Windows, Linux, MacOS, and Android. In this tutorial, we'll focus on the Linux OS.



operating system diagram

in this article.

Table of Contents

Why Learn the Linux Command Line?

There are lot of reason why you should learn about the Linux command line. Some of these are:

- **More Control Over Your Machine**: You have a great deal of power and control with the command line. You can run

- **It's Faster**: You can complete tasks much more quickly with the basic commands in your toolbox than you could with a Graphical User Interface (GUI). Just keep in mind that it might be slower while learning the CLI.
- **Automate Many Tasks**: You may speed up your work by using a single command to create 10,000 files, each with a unique name. With a GUI, this process is laborious.
- **Available Everywhere**: The instructions you issue will automatically run similarly on Linux and Mac computers. And with a little tweaking, they will also function on Windows.
- **Basic requirement**: You NEED to use the command line if you want to advance your knowledge in any coding-related technology field, including development, data analysis, devops engineering, system administration, security, machine learning engineering, and others.

History of Operating Systems

Most OSs are generally divided into two families: Unix-based and Microsoft NT descendants.

Unix was an OS developed in the mid 1960s. It's the "grandparent" of many modern operating system that we frequently use now, such as Linux.

The Unix operating system was a closed source project (meaning its code and files weren't made public). And this led to the rise of the "Free software" movement led by Richard Stallman. It argued that

Microsoft NT descendants were proprietary graphical operating systems that Microsoft created. The Windows NT descendants don't natively have similar Linux commands, unlike Unix and Unix-based Operating Systems, which do. Instead, Microsoft NT has its own set of commands and default shells.

Microsoft NT's offspring includes Windows, Xbox OS, Windows Phone/Mobile, and others.

The Rise of the GNU Project

Richard Stallman wanted to create a free software alternative to Unix. He worked with some other developers in 1984 to create a full operating system that would be free. So they started working on the GNU project.

At same time, another developer called Linus Torvalds was creating his own kernel known as Linux. At that time, many GNU pieces were completed but they lacked a kernel. Torvalds combined his kernel with the existing GNU components to create a full OS.

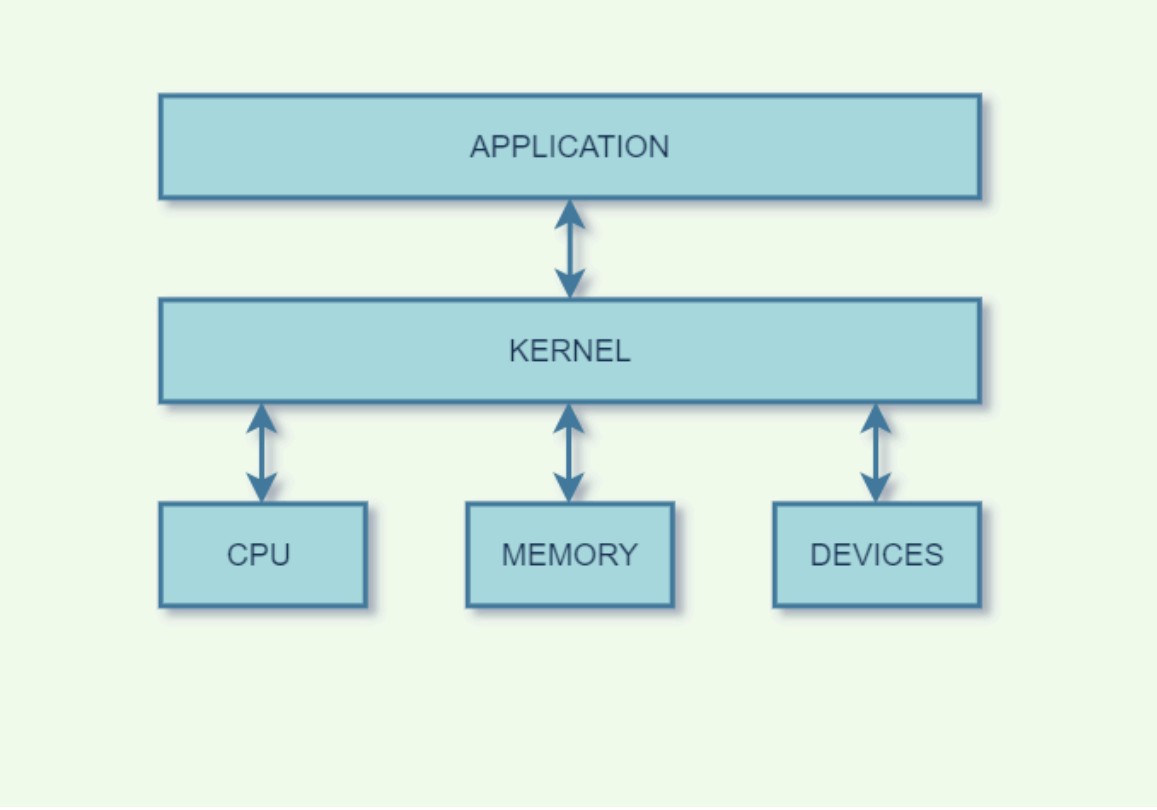
Some developers strongly feel that the name should be GNU/Linux instead of just Linux, as it reflects the joining of the Linux kernel with the GNU project.

How Linux Works and its Basic Components

What is a kernel?

A **kernel** is a part of an OS that facilitates interactions between the hardware and software. It's an essential element of an operating system for a computer.

The core of the OS alone is responsible for providing all other components with necessary services. It helps with device control, networking, file system management, process and memory management, and it acts as the main interface between the OS and the hardware.



kernel

exposes the services of the OS to users or other programs. The shell takes your commands and gives them to the OS so it can perform them.

It's named a shell because its the outer layer around the OS – like the shell around an oyster!

What is the Terminal?

A terminal is a program that runs a shell. This is where we run most of our commands that tell the OS what to do.

You install the terminal in the following ways on different operating systems:

- **Linux Distro users** – the Bash shell is installed by default
- **Mac Users** – Terminal is installed by default and can execute similar Linux commands
- **Windows Users** – Download Windows Subsystem for Linux (WSL) or use `git bash` and run all Linux command from there.

What are Linux Distributions?

Linux distributions (popularly called **distros)** are flavors of the Linux operating system. These distros are built based on Linux's open source software.

Some examples of these are:

more slowly than Ubuntu and Mint. It is one of the most reliable Linux distributors as a result.

Debian is the foundation of Ubuntu, which was created to expeditiously enhance Debian's fundamental components and make it more user-friendly.

Ubuntu was created by Canonical in 2004 and gained popularity immediately. Canonical wants Ubuntu to be used as a simple, command-line-free graphical Linux desktop. It's the most well-known Linux distribution.

Ubuntu is simple for beginners to use. It has a large number of pre-installed applications and convenient repository libraries.

Red Hat Family

Red Hat is a professional Linux distributor. Red Hat Enterprise Linux (RHEL) and Fedora are their products, both of which are open source.

Fedora offers faster updates and no support, but RHEL is thoroughly tested before release and supported for seven years after the release.

Red Hat uses trademark law to stop the redistribution of its software. Red Hat Enterprise Linux source is used in CentOS, a community effort that eliminates all of Red Hat's trademarks and makes it publicly available. In other words, it is a free version of RHEL and offers a long-lasting reliable platform.

SUSE Family

SUSE Linux was primarily developed in Europe and is of German origin. The name SUSE is an acronym for "Software und System-Entwicklung." SUSE is one of the oldest commercial distributions still in use because the initial version debuted in early 1994.

Fedora Family

This is a project that offers the most recent software versions and mostly focuses on free software. It uses 'upstream' applications instead of developing its own desktop environment. It comes with the GNOME3 desktop environment by default. Although less reliable, it offers the newest information.

How to Choose a Linux Distribution

DISTRIBUTION	REASON TO USE
Ubuntu	It works like Mac OS and easy to use.
CentOS	If you want to use red hat but without its trademark.
Fedora	If you want to use red hat and latest software.
Red hat enterprise	Used commercially.
OpenSUSE	It works same as Fedora but slightly older and more stable
Arch Linux	It is not beginner friendly.

Basic Linux Commands to Run in the Terminal

The `whoami` command

This command prints the name of the currently logged in user to the terminal session.

```
caesarsage@caesarsage:$ whoami
```

The `man` command

This command prints the **manual** or information about a command, configuration files, and so on. This command is very useful when it comes to getting more information about any command.

```
caesarsage@caesarsage:$ man whoami
```

The `clear` command

Clears all previous commands that were run in the current terminal. This clears the screen from previous commands in the terminal.

How to open files

Mac Users:

```
open <filename or directory name>
```

The open command lets you open a file or directory in the graphical user interface (GUI) outside the terminal.

Linux Users

```
xdg-open <filename or directory name"
```

Windows WSL users

You can open files in a similar way to Linux, but you need to install the xdg-open package.

Example for Linux and Windows users:

```
caesarsage@caesarsage:$ xdg-open clean-code-architecture.pdf
```

Now that we have covered the basic commands, let's learn a few other commands you'll use a lot.

How to Work with Directories in Linux

Directories are like folders, and you can create, delete, and perform all functions on them through your system interface with a mouse or cursor.

Here we will be doing something similar but from the comfort of our terminal. The following commands let you perform different operations on directories:

- `pwd` (present working directory)
- `cd` (current directory)
- `ls` (list)
- `mkdir` (make directory)
- `rmdir` (remove directory)

Let's look at what each one does:

The `pwd` command

Whenever you feel lost in the filesystem, call the `pwd` command to know where you are. It takes no argument.

```
casesarsage@caesarsage:~/Documents/github.com$ pwd
```

It should print the current folder/directory path where you currently are.

Directory). Just like going back and forth between folders when using the GUI.

```
caesarsage@caesarsage$: cd Documents/articles
```

This command takes me to a folder called articles inside my Documents folder.

Let's see what else you can do with `cd`.

`cd ~`

The cd is also a shortcut to get back into your home directory. Just typing cd without a target directory will put you in your home directory. Typing `cd ~` has the same effect.

```
caesarsage@caesarsage:~/Documents/github.com$ cd ~
```

This takes your to you home directory from the github.com folder

`cd ..`

To go to the parent directory (the one just above your current directory in the directory tree), type `cd ..`:

The `ls` command

Inside a folder you can list all the files that the folder contains using the `ls` command. It takes no arguments.

```
caesarsage@caesarsage:~/Documents/mycatfolder$ ls
```

Just like with `cd`, there are some other options you can use with `ls`:

`ls -a`

A frequently used option with ls is -a to show all files. Showing all files means including the hidden files.

When a file name on a Linux file system starts with a dot, it is considered a hidden file and it doesn't show up in regular file listings. This command will show those files.

`ls -l`

Many times you will be using options with ls to display the contents of the directory in different formats or to display different parts of the directory.

Typing just ls gives you a list of files in the directory. Typing `ls -l` gives a long listing and permission as (**rwx** - read, write, execute).

create your own directories/folders with **mkdir**.

You have to give at least one parameter to `mkdir` – the name of the new directory to be created. Think before you type a leading / .

```
caesarsage@caesarsage:~/Documents$ mkdir cats
```

The `rmdir <directoryName>` command

When a directory is empty, you can use **rmdir** to remove or delete the directory.

```
caesarsage@caesarsage~/Documents$ rmdir cats
```

`rmdir -p <directoryName>`

When you want to delete nested directories, you can used the **-p** flag. You use `rmdir -p` to recursively remove directories. This is similar to creating nested directories with **mkdir -p**.

```
caesarsage@caesarsage:~/Documents$ rmdir -p articles/drafts
```

In this section, you'll learn how to recognize, create, remove, copy, and move files using the following commands:

- `touch`
- `rm`
- `cp`
- `mv`
- `rename`

The `touch <filename>` command

One easy way to create an empty file is with `touch` like this:

```
caesarsage@caesarsage:~$ touch file1.txt file2.md file3
```

The above creates three files (text and markdown files).

The `rm <filename>` command

When you no longer need a file, use rm to remove it.

Note that unlike some graphical user interfaces, the command line in general does not have a waste bin or trash from where you can recover files. When you use `rm` to remove a file, the file is gone. So be careful when removing files!

Here are some more specific ways to use `rm`:

`rm -v <filename>`

This flags gives you feedback of what it did (deleting a file).

`rm -i <filename>`

To prevent yourself from accidentally removing a file, you can type `rm -i`. This will show a prompt to confirm if you really want to delete the file or not.

`rm -rf <filename>or<directory>`

By default, `rm -r` will not remove non-empty directories. However rm accepts several options that will allow you to remove any directory.

The `rm -rf` statement is famous because it will erase anything (providing that you have the permissions to do so). When you are logged in as root, be very careful with `rm -rf` (the f means force and the r means recursive), since being root implies that permissions don't apply to you. You can literally erase your entire file system by accident.

The `cp <fileold> <newfile>` command

To copy a file, use `cp` with a file name and a new file name argument.

```
caesarsage@caesarsage:$ cp text2.md text2Copy.md
```

If the target is a directory, then the source files are copied to that target directory.

```
caesarsage@caesarsage:~$ mkdir dir3
caesarsage@caesarsage:~$ cp file2.md dir3
```

`cp -r directorySource dirTarget`

To copy complete directories, use `cp -r` (the -r option forces recursive copying of all files in all sub-directories).

```
caesarsage@caesarsage:~$ cp -r dir1/dir2  dir3
```

The `mv source destination` command

You can use the `mv` command to move and rename directories.

```
caesarsage@caesarsage:~/Documents/$ mv cat catFolder
```

```
caesarsage@caesarsage:~/Documents/$ mv newarticle.txt articles
```

You can use the following commands to look at the contents of text files:

- `head`
- `tail`
- `cat`
- `less`
- `echo`
- `wc`
- `grep`

The `head <file>` command

This command prints the first part of the files. By default it gives the first 10 lines of a file, but you can override that by adding the `-n` flag.

```
caesarsage@caesarsage:$ head /etc/passwd
```

The `tail <file>` command

This command prints the last 10 lines of a file. You can also override the default similarly by passing the `-n` flag.

The tail file also has an `-f` flag that helps you keep printing extra additions to a file. This is useful for logs and errors files that keep

```
caesarsage@caesarsage:$ tail /etc/passwd
```

The `cat <filename>` command

`cat` can add content to a file which makes it super powerful. In its simplest usage, cat prints a file's contents to the standard outputs.

```
caesarsage@caesarsage:$ cat file
```

You can print the content of multiple files as well.

And using the **operator >** (we will see what this does later – for now, know it takes terminal output into a file) you can concatenate the content of multiple files into a new file:

```
caesarsage@caesarsage:$ cat file2.txt file3.txt > combine.txt
```

You can also use it to create files:

```
caesarsage@caesarsage:$ cat > newfile.txt
```

The `less <filename>` command

```
caesarsage@caesarsage:$ less /etc/passwd
```

Use **b** to scroll one page, **G** to the go to end, **g** to go to the start and **q** to quit the cmd.

The `echo` command

This command prints to the output the argument passed to it.

```
caesarsage@caesarsage:$ echo 'Hello world'
```

The `wc <input>` command

`wc` stands for word count, and this command gives information about input (for example a file) like number of lines, number of words, number of bytes for content, and so on.

`wc -l`

This option prints only the newline count.

`wc -m`

This option prints only the character count.

`wc -c`

`WC -W`

This option prints only the word count.

The `grep` command

The command grep is probably the most widely used text manipulation command. It lets you filter the content of a file for display.

If, for instance, you want to see all lines that include the word output in your file, you could use cat and ask it to display only those lines.

```
caesarsage@caesarsage:$ cat /etc/snort/snort.conf | grep output
```

You will learn more about the pipe (|) operator in the next section.

Linux Command Operations

Some common commands you can use to manipulate Linux commands are:

- `>`: redirects standard outputs

Most of the commands we have seen so far print something out for us on the terminal. For example the PWD prints out our current directory, and so on.

```
caesarsage@caesarsage: whoami > file.txt
caesarsage@caesarsage: pwd > file.txt
caesarsage@caesarsage: cat > file.txt
```

- `>>`: redirects standard outputs and appends new contents.

Unlike the '>' operation, >> doesn't override previously stored output in a file.

```
caesarsage@caesarsage: whoami >> file.txt
caesarsage@caesarsage: pwd >> file.txt
caesarsage@caesarsage: cat file.txt
```

- `|`: this operator is called pipe.

This takes the output of one command and passes it as the input for another command. Here's how you use it:

```
caesarsage@caesarsage:$ cat /etc/snort/snort.conf | grep output
```

Summary

As always, I hope you enjoyed the article and learned something new. If you want, you can also follow me on LinkedIn or Twitter.

Cheers and see you in the next one! ✌️

Destiny Erhabor

Hi. Glad you could check out my profile. I'm Destiny, a chemical engineering student majorly focused on software development, technical writing, cloud computing, power platform and building tech communities. I have vast and extensive experience on the MERN stack. My experience includes developing scalable and fast applications while still applying and implementing more efficient algorithms with JavaScript and it's various technologies, Python and Databases. I have experience using cloud platforms like Microsoft Azure, Google Cloud Platform, and Amazon Web Services. With about three years of experience working remotely and working in different diverse teams of all sizes, I have contributed to a range of projects or programs of all sizes.
