***Let’s explore the concepts as part of Section 2.***

In the previous section, we learned about fundamentals of Linux.

* What is Operating system
* Structure of Linux Operating System
  + What is Linux Kernel.
  + What is CLI Linux
* Linux Distributions
* How to setup Linux machine on your personal computer
* And finally, we learned about package managers.

In this section, we will learn about folder structures.

By default, when you set up a Linux machine, there are a lot of folders available on that Linux machine. Let’s try to understand what’s the significance of those folders is.

So, for that we have learned in the previous section how to start a Linux system on your Windows machine or Mac OS using a docker container. Let’s try to recollect it.



What we learnt is to run this particular command. You also find this command in [GitHub Documentation](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/01-getting-started/04-setup.md). If you go to [01-getting-started](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/01-getting-started) and click on [**04-setup.md**](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/01-getting-started/04-setup.md)you will find this sophisticated docker command. And when you run this, it sets up a Linux container of Linux environment on top of your Windows machine or Mac OS as well.

The only thing is you just need to have Docker Desktop running.

So, copy that code, and change the path accordingly. So, I have used “C:/Users/deven/Downloads/ubuntu-container-demo” as the location on my personal computer that is Windows computer. So, this is the storage my container will hold.

Once you run this command, you will get a container id. I have already run it, and I have the container id.

***Thought Provoking:***

**Hey Devender!**

**How do I get that container id again? I have run the command yesterday and now I don’t have the container id.**

Very simple 😊!

You can just run this command ***docker ps*** and this shows the container details.

Just copy it and run ***docker exec -it <contaninername or container id> /bin/bash***

For some reason let’s say you don’t want to use this, or you don’t want to use WSL as well, you can spin up an EC2 instance on AWS or an Azure virtual machine on Azure.

Once you spin that up, just click on connect (EC2 Instance connect) and when we click on connect, you will get the same linux environment and you can use any of these approaches. Either you can go for WSL or Docker environment on your Windows or Mac OS, or you can use an EC2 instance on the Windows machine, the option is yours 😊.

***Thought Provoking:***

**Hey Devender!**

**I now logged into Linux environment, and I see something here *root@ubuntu-dev:/#***

**What exactly is it?**

So, let’s first understand it. It has a meaning. It’s not that it doesn’t have a meaning.

So, what that line says is, it says ***root@ubuntu-dev:/#***

The first ***root*** is basically the user with which you logged into the Linux environment.

***Thought Provoking:***

**Hey Devender! What is the user?**

When you use your windows laptop on a day-to-day basis, you are quite used to using administrative users. So, when you buy a laptop, you become an administrative user, or you will be asked to set up a username and password. Every day when you use your windows laptop, you use that administrative user.

However, in real time when you use a Linux server in your organization, multiple people might be using that Linux Machine. So, by default you will get a user called root user, which is the administrative user on Linux.

However, your DevOps engineering team or system administrator team will set up multiple users. For example, for Devender, they might set up a user called Deven and for Manoj they might set up Mano. For different people, they will set up users and access for these users will be granted. Only the root user has administrative privileges. The rest of the user’s access is restricted according to what they do in the organization.

So, the first ***root*** is the user with which you logged in.

By default, when you create an environment using docker, you login as a ***root*** user.

At the same time, when you check EC2 instance here ***ubuntu@ip-172-31-82-183:~$*** the username is ubuntu. So, when you create an EC2 instance, what it does is along with the ***root*** user, it creates automatically another called ***ubuntu*** user.

We will learn more about how to create users, how to manage permissions of that user in our next section that is [***03-user-management***](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/03-user-management).

For now, let’s understand about ***root*** user.

This ***root*** says I’m logged in as an administrative user.

This ***ubuntu*** is basically your hostname or identity for your host. This doesn’t mean a lot in this particular case because you know anyways you logged into the system that you already know, unless you’re dealing 100 machines or 100 Linux environments.

And “***:”*** is just a separator. It doesn’t mean anything. But then you have this important thing ***/*** that is slash.

What is ***/*** ? 🡪 ***/*** Here is basically the path or your present working directory. And this particular thing has a lot of significance.

In this entire line ***root@ubuntu-dev:/#*** there are two things which are important. One ***root*** is the user with which you logged in and two is the path that you are currently in.

Now, what is this path? Very simple! In our previous section, we learned about hardware and when we learned about hardware, I have explained you something called as ***Filesystem***. File system in layman language is “filesystem is nothing but storage”. And for this storage, there is a parent location or there is a root location that is nothing but ***/*** slash.

So, by default when you get a storage, the path you get is just the ***/*** slash path. And all your files and folders by default are created in this particular location.

Just imagine Google Drive. For simple example, when you create a Google Drive, by default you get a parent location and within that location, you can create multiple folders, within each folder if you want you can create another folder. Within the folders you can see the files.

Similarly, when you buy a file system or when you get a Linux environment with a filesystem, the default path that you get is always the ***/*** slash path.

You can create any number of folders, any number of files within this particular filesystem that is ***/*** slash.

This particular thing **/** means you are at the root of your filesystem.

And now, this particular thing ***root@ubuntu-dev*** might change from system to system, environment to environment.

When I logged in using the docker environment, ***root@ubuntu-dev:/#*** this is the path that I have. But I login to EC2 instance, path ***ubuntu@ip-172-31-82-183:~$*** is very different. Now, let’s try to understand what is this path ***ubuntu@hostname: ~$*** here ***$*** is not much important as this just a separator and the “***:***” it’s a separator.

What’s important is again the user ***ubuntu*** followed by **~**. So, when you got EC2 instance, what it did apart from your ***root*** user, it created another user called ***ubuntu*** user and when you logged in to the EC2 instance, the default user is ***ubuntu***. This is for security purpose. EC2 instance did not want to grant ***root*** user or administrative permissions out of the box. That’s why it’s default sign in is ***ubuntu*** user. So, it will have less permissions when compared to ***root*** user.

Then what is this ***~***. So ***~*** basically is ***/home/ubuntu***.

So, docker environment gave you direct access to your filesystem that this ***root@ubuntu-dev:/#*** is your default location. Whereas, when you logged in to the EC2 instance, the default path is /***home/ubuntu***

***Thought Provoking:***

**Hey Devender!**

**How do you know ~ is /home/ubuntu?**

Basically ***~*** is the home directory of the any user that you logged in. So, if you logged in as ***ubuntu*** user ***~*** means /home/ubuntu.

Let’s say if you logged in as Deva as a user user ***~*** means /home/deva

***Thought Provoking:***

**Hey Devender!**

**What is this /home/ubuntu /home /deva?**

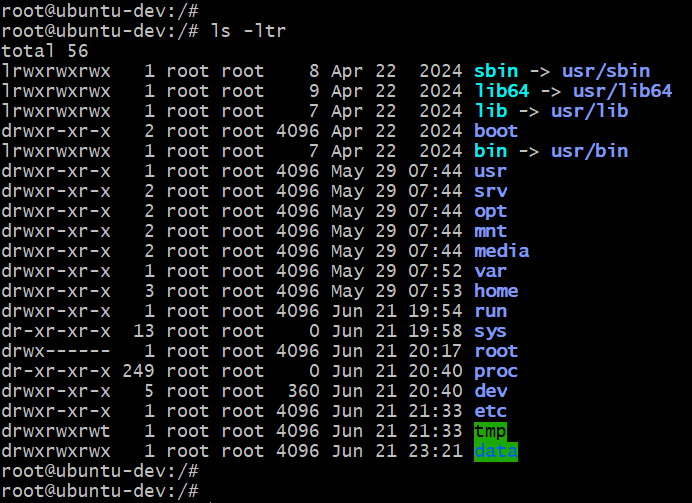
It’s very simple! 😊. As discussed, this is the parent location of the filesystem or root location of the filesystem.

Within this you will create a directory called ***home*** and within the ***home*** directory, you will create directories for every user on your Linux system. This comes by default when you create a user, it creates folder within the filesystem and within the ***home*** folder it creates a folder for the user that you have created. Again you will see this when you learn about user management [**03-user-management**](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/03-user-management)in the next section.

So, this is the difference between the docker environment entry point and the default entry point in the EC2 instance.

By now this should be clear.

Now, let’s understand, when you get a Linux environment, what are different folders that you get and what are different files, packages that you get.



If you just search for **ls -ltr** you will find various directories. So, ***ls*** command is basically *listing* command which will list all the folders and all the files in that particular directory. So, within your ***/*** root filesystem, you have so many directories.

***Thought Provoking:***

**Hey Devender! Why are these directories important?**

It’s very simple! 😊.

If you noticed, we entered ***ls*** command and it listed the files. Now, how does your Linux system know ***ls*** means to list the files? So, there has to be package and that package is provided in one of these folders.

So, basically, Linux environment has this package already installed in one of these folders. That’s why when we entered ***ls*** command, it listed the files.

Similarly, if I do ***mkdir test*** it creates a new folder called ***test***. But how does the Linux system know the meaning of ***mkdir***? There is a package in one of these folders and that let you create a new folder.

Similarly, all the administrative activities by default, any task that you need to perform, most of the things are setup in the Linux Environment. We learnt this in last section as well [01-getting-started/02-linux-structure.md](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/01-getting-started/02-linux-structure.md) If you look at the structure of Linux, on top of the Linux kernel, there are some system utilities and there are some system libraries along with the CLI.

CLI is something which we are already interacting. Apart from CLI, you have System Libraries and System Utilities. All of these System Libraries and System Utilities are basically present in these folders.

So, let’s try to understand the meaning of each folder

If you are on EC2 instance, what you need to do is you can just run this command ***sudo su -***

When you run the command, you will switch as ***root@ ip-172-31-82-183:~$*** to root user*.*

Then you can just run ***cd /*** just to take you to the root location of the filesystem and then you will find the same files exactly that you find the Linux environment created through docker.

***Thought Provoking:***

**Hey Devender!**

* **But what are these commands sudo su – doing?**
* **What is exactly cd / doing?**

It’s very simple! 😊.

You might know what is ***cd /*** doing. But you are not aware of what ***sudo su –*** doing. We will cover that in future sections and no need to worry about it at this point of time.

Now, let’s learn in docker environment instead of switching between different environments i.e; docker environment and EC2

Let’s start with the first folder 😊.

We have something called ***sbin***. What is ***sbin?***

******

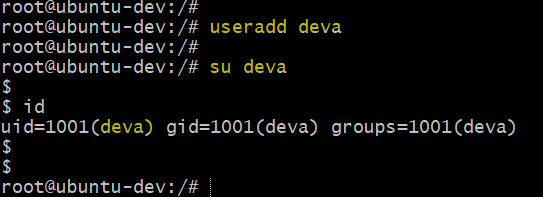
You can just go to***ls sbin*** and you will see there are a lot of system binaries in this particular folder. ***sbin*** means system binaries. So, what are this System Binaries? Basically, the commands or the binary files that you can use to manage your system.

Let me give you some simple examples.

There is something here called ***useradd***. So, what is ***useradd?*** This particular command or this particular binary helps you to create users.

As a Linux administrator or somebody owning this Linux machine. If you want to create users, you will use this command. And when you run this command, this actually goes to this particular folder ie; ***/sbin/useradd*** and this ***useradd*** command gets executed and a user is created.

For example, ***useradd deva***



When I run this, a user called deva is created on this Linux machine. And I can go to that user by using ***su*** command ***su deva***It ask for password or it will directly switch to the user because I didn’t provide the password at the time of user creation. Press ctrl+d to switch back to root user.

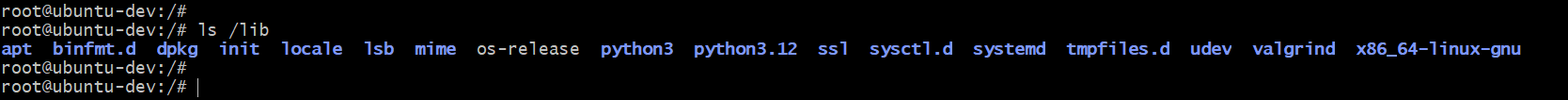
To recollect, what exactly are the files present inside ***sbin***. This are nothing but System Binaries which helps you manage your system as an administrator.

For example, ***useradd*** to add the user ***groupadd*** to add a group, ***userdel*** to delete users, ***groupdel*** to delete groups ***add-shell*** to add a new shell ***chcpu*** check the cpu, ***chmem*** check memory. All of these are present in ***sbin*** system binary folder.

I have also provided detailed documentation. If you want to summarize, just go to [**README.md**](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/02-folder-structure/README.md)file of [**02-folder-structure**](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/02-folder-structure) you will find ***/sbin*** is Sytem binaries for administrative commands.

Now let’s go to the next folder. Again, I will just run ***ls -ltr*** and then let’s look at ***lib***

The name ***lib*** name itself explains it is library folder. ***sbin*** is System binariesand ***lib*** is basically libraries.



What exactly are inside of this library file? Just to ***ls lib*** and you will see different kind of libraries that are available here and these libraries are used by your Linux Kernel. So, as a user, you don’t use these libraries, these are used by Linux Kernel for making system calls with the hardware or basically to execute its action.

Then you have a folder called ***boot.*** So, ***boot*** is basically for booting your Linux Machine that is starting or restarting your Linux machine. So, when your Linux machine restarts or maybe it starts, whatever inside the boot folder are basically helps the Linux machine during the restart. Basically, the commands that it require while it is starting or actions that it needs to execute during the start of the machine.

If I do ***ls /boot*** I’ll not find anything on the container environment, whereas if I run the same command on EC2 instance, there are a lot of commands on the EC2 instance. Why?

EC2 instance is a virtual environment, that is you know this actually restarts your Linux environment whereas this on container related Linux environment is just simulation. So, that’s why you don’t find anything inside the ***/boot*** on container related Linux environment.

If you restart the container environment, it doesn’t do any action. Whereas, if you start EC2 instance or you just stop the instance and restart it, there are certain actions that it performs and that is basically using the files which is inside the ***/boot***.

Then you have ***/bin***

So, there is slight difference between ***/sbin*** and ***/bin***. So, what is the difference between ***/sbin*** and ***/bin***?

***/sbin*** stands for System Binaries. Whereas, ***/bin*** stands for user binaries.



So, if I go to ***/bin*** you will find commands here as well. But these are not administrative commands or these are not system commands. Maybe you can grant access to these commands to your regular users as well.

For example, if I just use ***date,*** it will just print the date and time on the Linux machine. Now, this doesn’t have to be administrative action:

Anybody can get access to these particular binaries or this particular commands. Similarly, let’s say user wants to check difference between two files. There is one command in linux which is called ***diff*** command. Now same the checking differences between two files doesn’t have to be an administrative access.

So, you can give access to anyone on this command. Similarly, there is ***link*** command and command like ***mount*** which is used to mount a filesystem or may be ***sed***. ***sed*** is basically for an operation on the file system or modifying a particular line on a particular file.

So, all of the command that you have or all the binaries that you have inside ***/bin*** are non-administrative. If you are a Linux administrator, what you can simply do is you can grant permissions to ***/sbin*** only for system administrators whereas you can grant access ***/bin*** to every user.

So, that is why Linux segregated commands in two different folders. If it didn’t segregate commands in two different folders, everyone might have action to every command on the Linux machine.

Let’s recollect 😊!

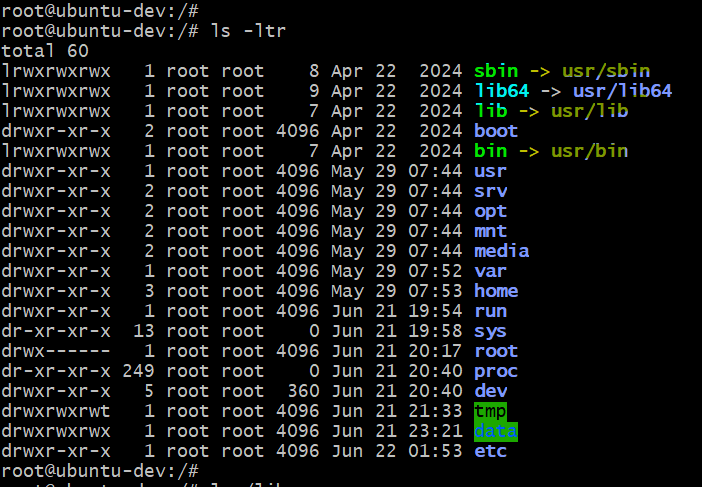
Within the Linux system that you create, basically there are a lot of binaries

* What are binaries?
  + Binaries are nothing but files which are executable.
  + In simple sense they can be considered as commands.

So, out of the box there are hundreds of commands. But some of the commands are administrative commands. Whereas some f the commands are non-administrative commands.

So, if a user executes these commands (non-administrative) there is not severe action. Whereas, if you grant access to administrative commands to regular users, they might create dummy users, they might create passwords to the users and they might probably hack the system or corrupt the system.

So, that’s why Linux segregated them to ***/sbin*** directory (for administrative commands) and ***/bin*** directory (for non-administrative commands). ***/sbin*** directory you can grant access to admins and ***/bin***directory to everyone.



In fact, all of these binaries are within a parent folder which is ***/usr***. If you look for ***/usr/sbin*** you will find administrative commands and ***/usr/bin*** you will non-administrative commands. That’s why when you see ***sbin***, ***sbin***is a shortcut for ***/usr/sbin*** and ***bin*** is a shortcut for ***/usr/bin***

***Thought Provoking:***

**Hey Devender!**

* **What is this shortcut?**

Imagine you have a movie or you have a game that you play regularly on your windows machine and this is in one of the directories something like C:\Users\deven\Downloads\Movies\Hollywood\pubg.exe. Every time you don’t want to the path and then run the movie or game application. What you simply do is if you right click on the particular file and you say *Create Shortcut***.** And when you create this on your home directory or any location that you provide, a shortcut is created to the file.

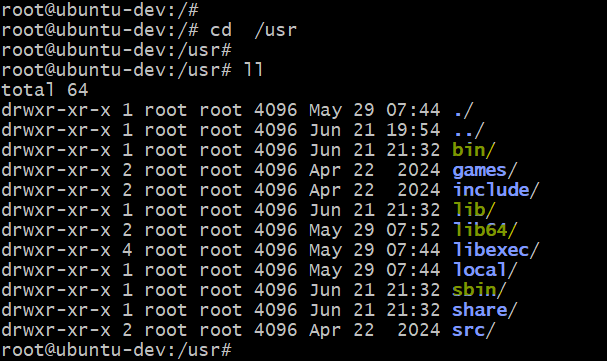
When you execute the shortcut, it will actually run this command C:\Users\deven\Downloads\Movies\Hollywood\pubg.exe, file is executed from this location C:\Users\deven\Downloads\Movies\Hollywood\pubg.exe itself.

We are using the same concept in Linux as well. However, in Linux, we don’t right click or something. We use a command called ***ln***. We will learn about it later sections. But these are the shortcuts.

/sbin -> /usr/sbin

/lib -> /usr/lib

/bin -> /usr/bin



If you go to ***/usr*** folder you will realize that. It has ***bin*** directory, it has lib*directory, it has* **sbin** *directory.* You have ***share***, let’s say you are using the Linux environment for sharing the files or basically using it has a common storage location, you can store that file within the ***/usr/share***. ***/usr/src/*** is just like any source code that you have any common thing that you want to share in terms of the source. But the important things within /***usr*** directory are basically ***sbin bin lib*** and shortcuts are created for all three of them.

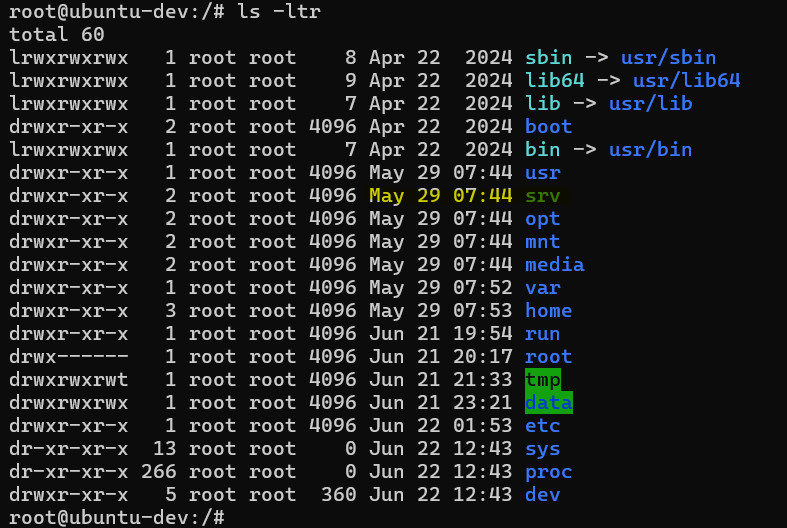
***Recap:***

Till now we learned

* What is ***sbin***
* What is ***bin***
* What is ***boot***
* What is ***usr***

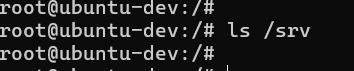
At any point of time, if you want to revise these topics, go to the repository [02-folder-structure](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/02-folder-structure) and here [**README.md**](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/02-folder-structure/README.md)you have the complete explanation of what I have covered in the section.

Now, let’s see the other folders.

What is ***srv***?

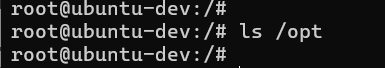
So, these shortcuts are you know these names actually have a meaning. If you want to remember that, ***usr*** is basically users. Similarly, ***srv*** is basically server.

So, you have a folder called ***srv*** where you can store configuration files or any important information related to your web servers. By default, when you create some web servers, it will actually store some file inn the ***srv***.



By default, it doesn’t have anything if I do **ls /srv***.*By default there are no files here because I don’t have any web server installed on this Linux environment.

Let’s see what is ***/opt***



Again, in ***/opt*** there is nothing. ***/opt*** is very important folder in your Linux environment. When you join an organization or when you are working in a team and you want to install some third-party dependencies.

***Thought Provoking:***

**Hey Devender!**

* **Can you give me some examples?**

Let’s say you want to install a particular version of **Java** or you want to install a third-party tool called **Krish.** What you will do is, you will use this folder called ***/opt***.

So, if I have to install a custom version of **java**, I will go to ***/opt*** and then as an administrator, I can do it. Or if you are a non-admin user, you need to switch to root or you need to take root permissions or you at least need to have access to ***/opt*** folder. You will learn that in user management [**03-user-management**](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/03-user-management).

Anyways, you will go to ***/opt*** folder and you will create a directory here, let’s say you are looking to install a tool called custom-tool. So, you have to **mkdir custom-tool**.You will switch to that custom-tool **cd custom-tool** and all the files related to that **custom-tool** like the executable things, any shell scripts related to it or documents related to it, you will place it in ***/opt/custom-tool.***

So, this ***/opt*** is a common location for all your third-party dependencies.

***Thought Provoking:***

**Hey Devender!**

* **Why can’t I do within my user?**

For example, I have a user called ubuntu.

Why can’t I go to ***ls /home/ubuntu*** or ***cd ~***. As already explained, both are same. Why can’t I go to ***cd ~*** and copy the files related to the **custom-tool** or create a shell script related to that **custom-tool.** If you do that, who will have access to that **custom-tool?** Only people who have access to ***/home/ubuntu***.

In general, this particular directory **/home/ubuntu** is very specific to your user. You might want to gran access to your home directory to everyone in the organization. Maybe you are saving some personal files here or maybe you have some personal scripts here.

That’s why for third-party dependencies; you need a common location and that common location is ***/opt***. You know this is where people knowing Linux and people just using Linux differ. Sometimes people install third-party dependencies in random locations, some people just install in **/tmp.** Some people just create a random directory called XYZ and within that they start installing third-party dependencies.

But person who knows Linux, they would create and use that within ***/opt*** because this is a common practice in Linux.

Then you have a folder called ***/mnt***.

So, **mnt** means mount.

Why is this important?

You know, system administrators in Linux, they usually have a task which is adding more space to the file system.

This Linux environment is created, after sometime let’s say initially it is created with 20GB storage. After sometime 20GB is completely utilized and you want to add 30GB. So, Linux administrator can do it. But for doing it what Linux administrator needs to do first is they need to procure a 30GB disk somewhere for a temporary or a permanent disk and then they have to mount it or first add it to this temporary location **/mnt**.

So, they will first mount it at ***/mnt*** and then they will add this disk to the filesystem. For doing this there are commands. We will learn that in [10-disk-management](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/10-disk-management) section for disk management.

For now, just remember ***/mnt*** is a folder that is used by the Linux administrators to mount new volumes. Volumes are nothing but new disks.

So, every folder has a meaning, you have to understand.

* **/sbin** 🡪 ***System Binaries***
* **/bin 🡪 Regular Binaries**
* **/mnt 🡪 mounting a volume**

So, you need to know what each folder is doing and use the specific folders for working on a specific task.

Then you have ***/media***. You can skip it. This is just for adding any audio files or MP4 files usually as DevOps engineers or developers you might not use this folder called **/media.**

People who are using this as their personal machine like some people use Ubuntu on their personal laptops, they might be using **/media** section. But usually we don’t use it.

Then you have a folder called ***/var***

So, what is that **/var?**

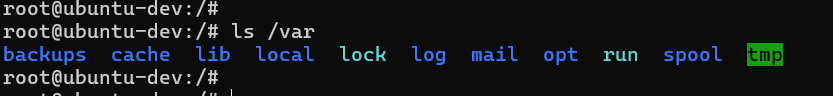
**/var** is basically used to store log files or may be some libraries (not the regular libraries but some third-party libraries or libraries that you download from internet). But mostly **/var** is used for log files. Sometimes when you install a server or when you install a third-party application, it might add the scripts to **/var/lib** followed by any folder that they want.

***Thought Provoking:***

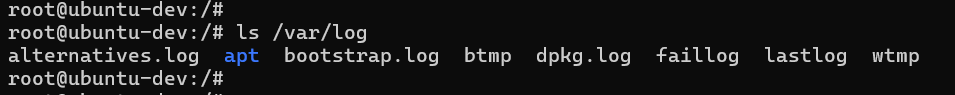
**Hey Devender!**

* **Why kind of log files?**

Let’s say you install a web server. Now, this webserver like Apache or http, the logs of the server might log to **/var/log.** You can find them by default within the **/var** folder as well.



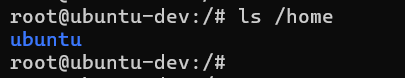
If I do **ls /var** you will find there is a **log** folder, there is a **lib** folder, there is a **cache** folder and multiple other thigs.



If I go t **/var/log,** within this, you might find some temporary logs or non-temporary logs. You can find some logs that are related to the system. But when you install Apache or HTTPD, it will create another folder inside **/var/log** and within that folder you have the log files related to the web servers or your third-party applications.

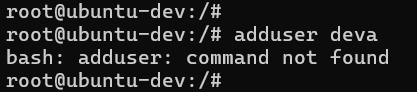
So, this was about ***/var.***

Now, let’s see next folder **/home.** We already covered **home.**



If I just do ls **/home** you have a folder for **ubuntu**. Why you have a folder for Ubuntu?

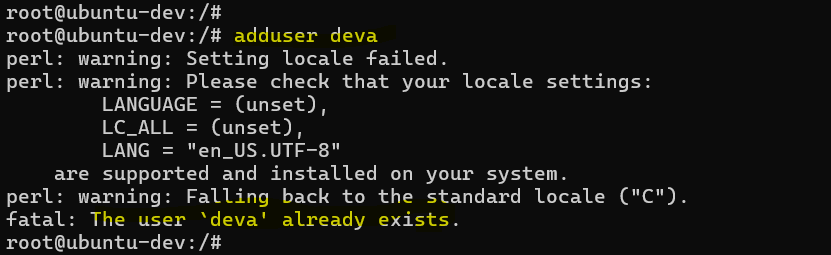
When you have created docker Linux environment, it created **ubuntu** user. However, you logged in as **root** user.



If I create another user, for example, if I do **adduser deva**. It says **adduser** command doesn’t exist.

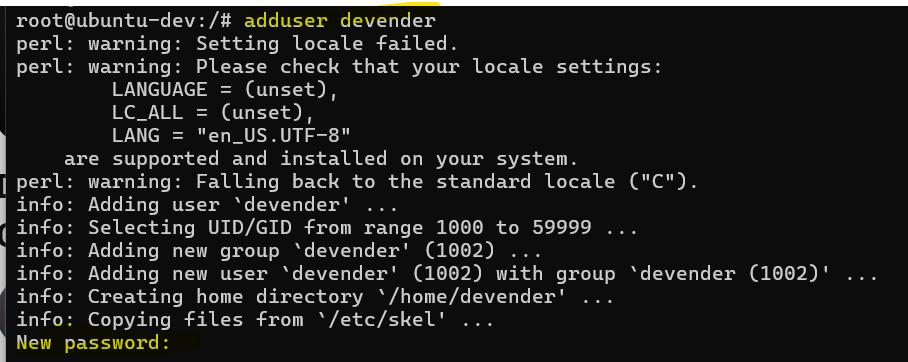


I can quickly do that. ***apt install adduser***.

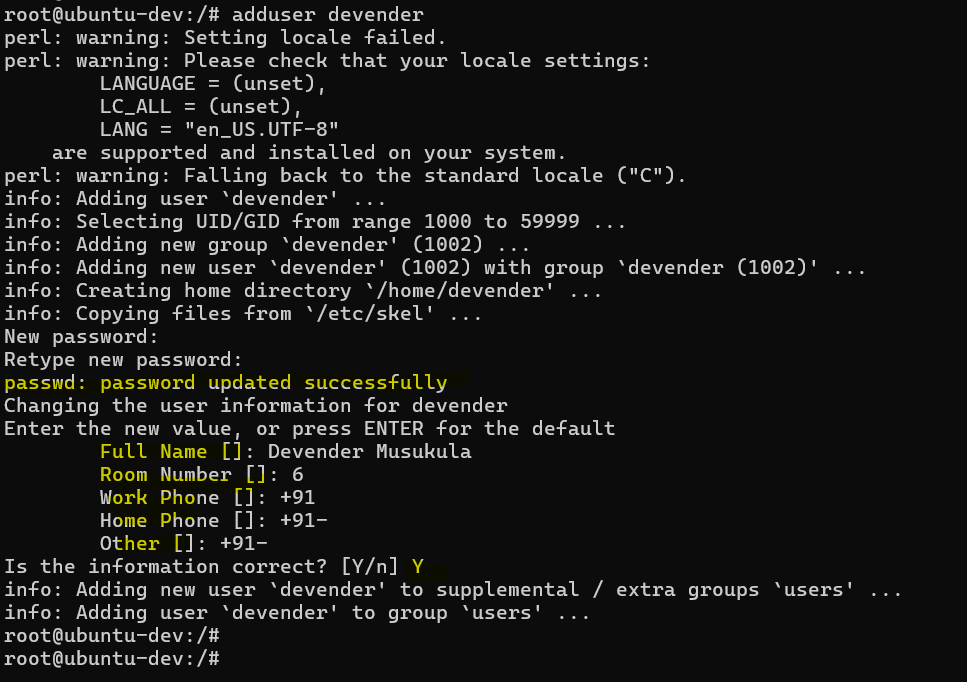


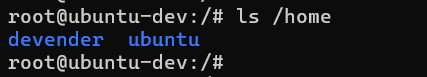
Now, if I do **adduser deva.** Ok it says user named **deva** already exists on the Linux machine. Let me do **devender.**

**Note:** There are two commands to create a user – adduser and useradd. Useradd command creates user but doesn’t create home directory.



So, it is asking for N**ew password:** I will just give a random password. Just enter random details for now.





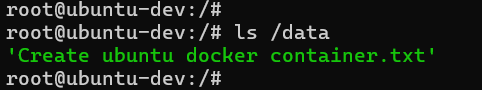
Ok, now it says **devender** is a user that is created. So, now within **/home** you will find **ubuntu** and **devender.** This is what we discussed a bit ago. Whenever you create a user using **adduser** command, you will see within the **/home,** **ubuntu** folder and **devender** folder or user folder is created. Similarly, **ubuntu** or the user folder is created.

Now, within these folders, you can create files that are specific to you. But for that I need to specifically login as **devender** user or **ubuntu** user.

If I’m saving some files at ***/*** slash, that means everybody can access that files you are basically making that files public. If I’m storing that within my user directory, they are private to me.

So, this is about **/home**. Home itself explains it is home directory.

Now, let’s look at **/data**.



So, basically if you just do ***ls /data***, this is basically for storing any data.

Let’s say I want to share data with other people or I have data related to organizational billing information, organizational cloud cost or people in the organization, any data that can be shared with other people in the organization or other admins in the organization, you can put that in ***/data.*** It depends on how you restrict the access to the folder.

What’s next?

After **/data** you have set of temporary or volatile files or volatile folders. So, the data inside this are basically not permanent.

For example, you have **/proc**. So, **/proc** is basically virtual filesystem for your processes and system information. You will learn more about **/proc** and **/dev** when we use some scripts or when we head to the next sections. For now, you can just understand these are temporary folders where you can store files that are not permanent.

Similarly, ***/tmp*** as well. The name itself says temporary.

Imagine if you want to have a file that only lives for a certain time. Instead of deleting it after using, what you can do is you an create that files or folders within **/tmp.**

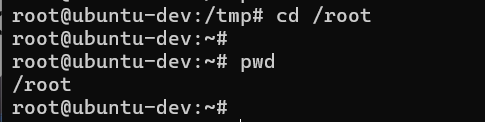
For example, if I go to **/tmp** and if I create a directory called **devender**.



Right now, you have this directory. But usually, every filesystem or any Linux environment, they have a time frame where they clear the temporary directory. Just like recently deleted images on your mobile. So, when you delete image on your mobile, it still stays on the recently deleted images and after a while may be after 30 days or 20 days or 1 day 2 days the files are deleted. Same is with **/tmp** as well.

So, when you are writing scripts and your scripts generates a log file, you want to delete that after a while, put that in the **/tmp**. You will learn more about **/proc** **/dev /sys** and **/tmp** when we head to simple scripts and next sections.

We know what is **/root**. It’s the home directory of root user. We are already within **/root** if I do **cd /root/** this becomes the root directory or the home directory of root.



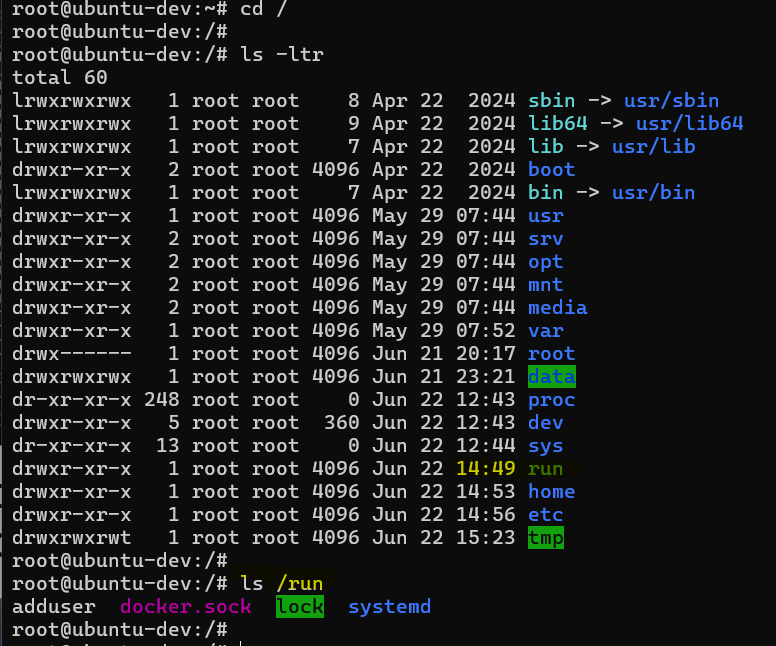
***Thought Provoking:***

**Hey Devender!**

* **You said home directory for ubuntu is /home/ubuntu.**
* **Similarly, home directory of devender is /home/devender**
* **But for root why is the home directory /root?**

So, **/root** is the only exception, the root home directory is **/root.** Rest any user that you create home directories starts with **/home** i.e. they are inside the **/home** folder.

Then you have **/test** basically a test folder, **/opt** we have already covered it. Then you have **/run.**

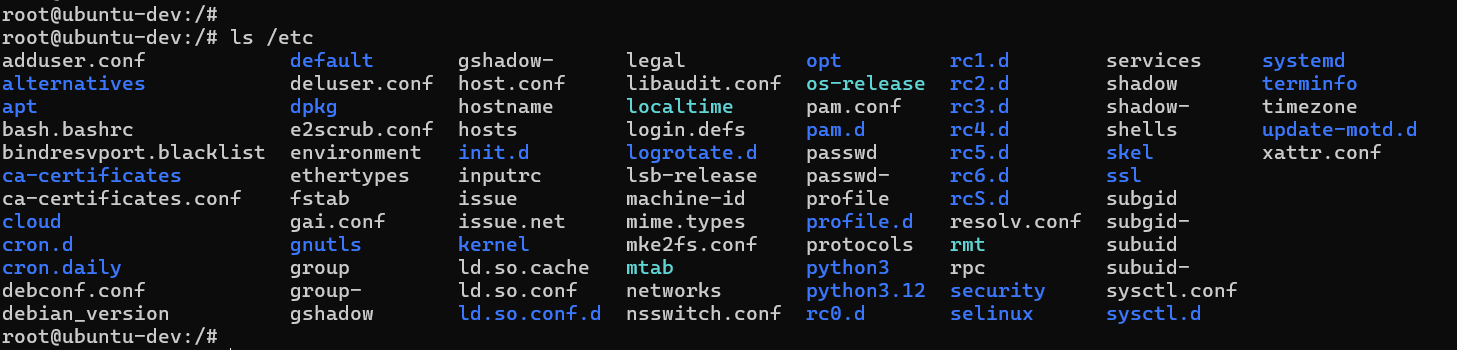


So, what’s inside **/run**?

So, ***/run*** basically stores the runtime data of the processes. That is, while a process is running, if it has a runtime data, that is stored within the **/run** folder.

The **/home** we already covered. Finally, you have something called as ***/etc***

Although it is **/etc**is the final one, it is one of the most important folders of your Linux system.



Within **/etc** you have a lot of system configuration files.

If you are using a Windows machine, you can somewhat corelate this to the **C:\** folder.

In Windows, you have a C drive within which you have all the System Configuration files.

Similarly, in Linux you have a folder called ***/etc*** within which you have all the system configuration files.

What does that mean?

Using these files, you can configure or modify your system. These are just like your setting on your mobile phone or settings on your laptop.

Let’s see for example, **/etc/passwd** which is password. Using this you can modify password of any user on the Linux machine.

Let’s take another example, **/etc/hosts.** So, **hosts** is a file where you can add local DNS caching for your Linux Environment. Maybe you can say that this particular website has this particular ipaddress.

Another example, **/etc/os-release**. So, this is a file where you have the Linux environment OS details.



If I just do **cat /etc/os-release** all the information about your Linux environment, the operating system version of it everything can be seen.

You have these **rc** folders where you can add your executables and this is the priority of the executables.

So, everything that is related to configuration of your Linux system is added to the **/etc** folder.

To recall, **/etc** is just like settings on your mobile phone or your laptop. And this is the most important configuration files of your Linux system.

😊

These are the different folders that you have out of the box on your Linux environment. And as you proceed with next sections, we will start using more commands from these particular folders.

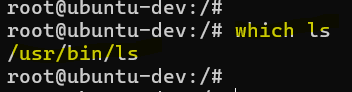
One final question that you might have

***Thought Provoking:***

**Hey Devender!**

* **If I run ls, you have explained that ls should be part od some of these folders, but I’m just running ls and its returns the output.**
* **I didn’t instruct the Linux machine that ls is in one of these folders.**
* **I did not say which is the correct path of ls command.**
* **So, how my Linux filesystem is able to execute it?**
* **How my Linux Kernel is able to execute it?**

Now, this is a very important question 😊



For example, **which ls** tells me what is the actual location of ***ls***.

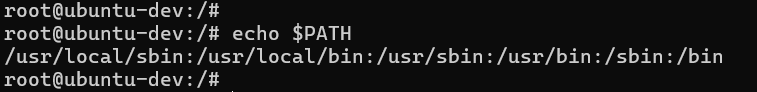
Ideally, I should have run this command **/usr/bin/ls** and only then it should return the output.

If I just run ***ls*** how my Kernel is able to give the same output? I didn’t even configure the shortcut.

So, there is something called as **PATH** on your Linux machine. This is also available on Windows. You might not use it regularly.

But what this **PATH** does?

So, this will tell the Linux system that when somebody enters the command or when somebody write something on the terminal, basically check in all the folders within the PATH and if it is in one of that folders, you can directly execute it.

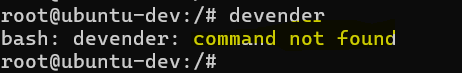


**echo $PATH**

You will see, within path***:*** is the separator but you have **/usr/local/sbin** **/usr/local/bin /usr/sbin /usr/bin /sbin** and  **/bin.**

Now, if **ls** command is in one of these folders, only then it will be able to execute. It is in **/usr/bin/ls** and **/usr/bin** is in **PATH.** That’s why **ls** command is executed.

Imagine, if this is not part of the **PATH** then when I run **ls** command, the kernel says “I don’t know what is this command”



If I just run the command **devender**, you will see the same output **command not found**. Because there is no binary called **devender** in one of these folders.

Now, if you want to add more folders to this (PATH) you can append PATH, you can add more paths to the PATH environment variable which we will learn in the future sections.

But for this section, what we learned is the folder structure of Linux. And this is more than enough.

If you want to revise these topics, you know where to go [**02-folder-structure**](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/02-folder-structure) and its made more simple in the documentation. If you go to the [02-folder-structure/README.md](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/02-folder-structure/README.md) file, I have categorized them into different sections

* So, first there is a section for the system configuration files a regular configuration file.
* Then you have Important System Directories
  + /etc
  + /var
  + /boot
  + /usr/
* Then you have application or user specific directories which is in
  + /home
  + /opt
  + /srv
  + /root
* Then the folders were categorized into temporary or volatile directories
  + /tmp
  + /run
  + /proc
  + /sys
  + /dev
* And then you have Mount points
  + /tmp
  + /media
  + /data

This [02-folder-structure/README.md](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/02-folder-structure/README.md) will make you learning easier after going through this section if you go through them, it will become a easy revision.

Thank you so much 😊!