***Let’s explore the concepts as part of Section 6.***

Let’s try to understand

* What are File Permissions?
* Why is it required on a Linux Server?

To understand it quickly, let’s recap a concept that we learned in previous section i.e. **user management**.

So, what we learned about **user management**?

* We learned that Linux is a multi-user system.
  + That means on a linux server, you can setup multiple users and these users can share access to the server.

***Thought Provoking:***

**Hey Devender!**

* **Why multiple users need access to Linux Server?**
* **Is it a real time thing?**

Yes 😊!

For example, if you take an organization called ORG, and assume these organization deploys applications onto a Linux server, quite obviously the developers of the organization, QA/QE engineers, DevOps engineers and multiple other engineers would need access to the Linux Server and you cannot grant all of them with the admin user and admin privileges or maybe ***root*** user and password.

So, what does Linux administrator does is to create a unique user for everyone that requests access to the server.

For example, if a developer called ***dev1*** requests access to the server, Linux administrator creates Linux user for them with name ***dev1.*** Similarly, a unique user for ***dev2***, **QE1, QE2 ...** for any number of people that request access to the server. And this process of creating users, creating groups and managing users and groups is called as ***user management***

***Thought Provoking:***

**Hey Devender!**

* **I have understood what is** *user management***. But why** *file permissions* **are required on a Linux server?**

If you take the same example of above, so, on a Linux server multiple users. And what if all the users on the Linux server have access to all the files and folders on the Linux server. i.e. ***dev1*** user has access to all the files and folders created by default, created by other users. Similarly, ***QE1*** has access to all the files and folders created by default, created by ***dev1***.

This is going to be a huge problem.

Why?

What if a user deletes a important folder like **/etc** or ***/sbin***? This is going to corrupt the entire file system, not only important folders.

Imagine a user called ***developer*** and this user creates a script in ***/tmp*** folder. Maybe a script for automating their day-to-day activities and accidentally a user called ***QE*** gets to the ***/tmp*** folder and deletes the file that is created by ***developer***user.

Now, this would impact day-to-day productivity of the user called ***developer.***

In any of these cases either

* Corrupting the file system
* Deleting or updating the files that are created by other users.

It is a problem to the **user management** featureof Linux.

***User Management*** is a very good feature. But what if one user is messing up with the files of the other user or a user is deleting the important files from the file system?

There is no point of ***user management***.

So, to solve this problem, Linux came up with the concept called ***File Permissions.***

So, **file permissions** is a feature that compliments ***user management*** of Linux. And File Permissions is out of the box. You don’t have to install any libraries or you don’t have to install anything. It comes out of the box.

And what Linux does out of the box? – It sets up basic file permissions on all the files and folders.

So, whenever you create a file, Linux automatically sets up file permissions on that file or even on the folder. However, this is a very basic thing.

So, Linux creates same file permissions on all the files and all the folders. If you want to modify it according to your organization requirement or maybe according to your personal requirement, then you need to know how the file permissions work in Linux.

And Linux offers some commands like ***chmod, chown*** using which you can change the default file permissions.

So, understand this carefully.

* By default, Linux provides permissions on each and every file or Linux configures file permissions on every file and folder.
* If you want to modify it according to your requirements, you can use commands like ***chmod*** and ***chown***.

So, this is an important thing that Linux supports to complement the user management feature.

***Thought Provoking:***

**Hey Devender!**

* **Can we see how Linux sets up the file permissions?**

We can definitely see that 😊!

So, for that I will create 2 users. A user called ***developer,*** a user called ***QE***.

Using the ***developer*** user within ***/tmp*** folder I will create a script file.

Then I will show you, if I login as QE user I cannot update this script file, I cannot delete this script file as well.

All that I can do as a **QE** user, I can only read the file.

So, by default, one user can read the files that are created by other users but cannot update or delete the files.

Once we understand this, then I will show you how you can modify it.

Maybe as a ***QE*** user, as per your organizational requirement, you shouldn’t be even able to read the file. I will show you how to modify the file permissions as well.

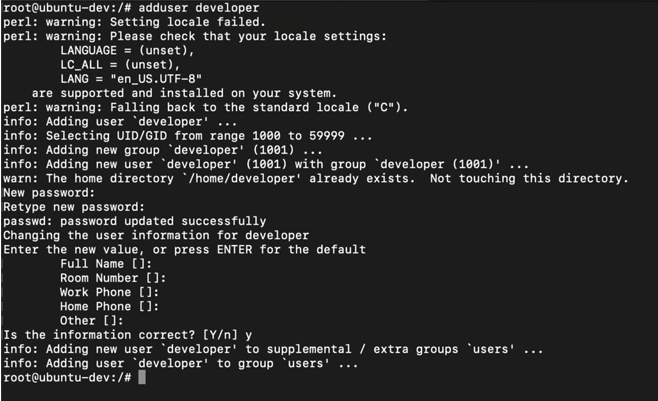
First, let’s understand this through the example.

Let’s get back to Linux terminal in container environment.

In the previous sections I have explained you how to create a Linux container environment. If you haven’t followed, you can go to [04-setup.md](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/blob/main/01-getting-started/04-setup.md) documentation. However, if you are not interested in a container environment, you can use EC2 instance, Azure virtual machine, WSL, Oracle Virtual Box. In any of these the concept remains the same because they are just things to connect to your Linux server or a Linux virtual environment.

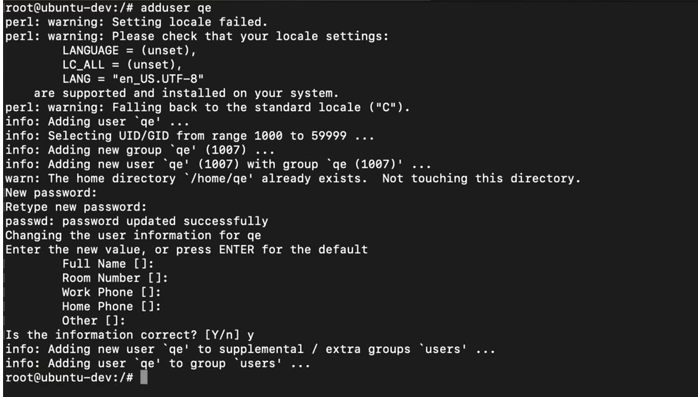
So, first thing first. I need to create two Linus user, user1 is ***developer.***

How do you create users? ***adduser developer***

******

Let me setup a password and I will leave the details as blank.

Similarly, let me create another user called ***qe***. Again, I will setup password of **qe** user and leave the details as blank.



Now, let’s verify if both the users are created.

How do you verify it?  ***cat /etc/passwd.*** So, when I do this, you will notice that ***developer*** user is created and ***qe*** user is also created.

Let us take two tabs, one of the tabs, I will switch to ***developer*** and in another tab, I will switch to ***qe***.That means, one session you’re connected to container environment as ***developer*** user and another session you’re connected as ***qe*** user.

Using ***developer*** user, I will get into /tmp folder and create helloworld.sh.

*#!.bin/bash*

*Echo “hello world!”*

As ***qe*** user get into /tmp folder, you can see the file helloworld.sh created and owned by ***developer*** *user.*

**Can I read the file?** cat helloworld.sh 🡪 I will be able to read the file.

**Can I update the script file?** Ideally, I should not be able to do it because ***developer*** created it and ***developer*** knows the script and my knowledge as ***qe*** is very limited and because of that’s the script of someone else, I shouldn’t be able to update it.

**Can I delete the file *helloworld.sh?*** rm helloworld.sh. It asks are you sure you want to remove a file? Yes 🡪 rm: cannot remove ‘helloworld.sh’: Operation not permitted.

So, this proves

* Linux by default comes with file permissions or there are default file permissions that are set up by Linux on every file and folder that is created.
* If you want, you can modify it.

***Thought Provoking:***

**Hey Devender!**

* **How can I modify?**

You can modify it 😊!

Now, let’s say.

As a ***qe*** user, I should not be able to read the helloworld.sh file as well.

Till now if I do cat helloworld.sh as ***qe*** user, I’m able to read the file.

Now, can I change this? Absolutely 😊!

There is a command called ***chmod***.

Don’t worry, I’m trying to explain it very clearly. But just I’m trying to show it through the demo. Then we will learn the permissions, what exactly is the command that I’m executing.

Just see what is happening now.

What I’ll do is ***chmod o= helloworld.sh*** as **developer** user and perform ***cat helloworl.sh*** as the ***qe*** user.

It says cat: helloworl.sh: Permission denied

So, with one command, I changed the permissions on the file that are set up by Linux filesystem 😊. This proves Linux sets up basic file permissions and as users we can modify the permissions on the files.

Now, how exactly it works?

***Thought Provoking:***

**Hey Devender!**

* **I need to understand how the file permissions on the Linux Environment work?**

If we go back to the terminal and to ***ls -ltr*** (as developer user). Till now we just know that ***ls -ltr*** lists the files and using this command you can see all the files and folders on this particular environment or in the particular directory.

But if you watch carefully, the first word of the output has a unique pattern **- r w - r w - - - -** so, this particular word represents file permissions on the files or the folders. So, this decides the permissions on files and folders on the Linux server

**But Devender, what exactly is this**

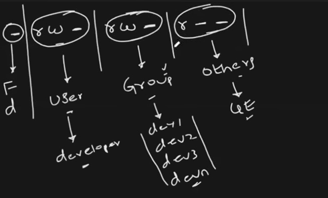
If I write this down. So, the permissions on the file **- r w -** and r **w -** and **r - -**

First character ‘**- ‘** represents it is a file or a directory.

r 🡪 read permission

w 🡪 write permission

x 🡪 execute permission

****

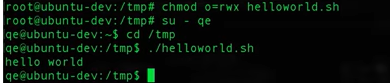
But as root user you can perform any operations.

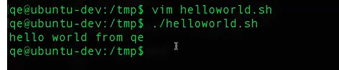












Using file permissions, you can control permissions or user group as well as others. And the command that we use is **chmod.**

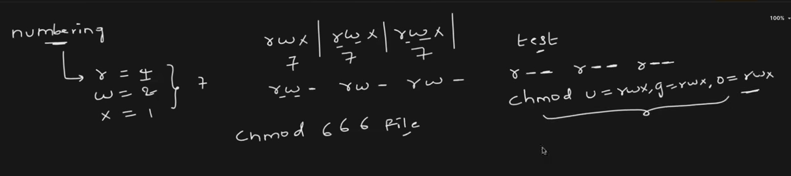
**But Devender, Is there any easy way to do it? u= g= o= its bit confusing. Is there any other way Linux supports?**

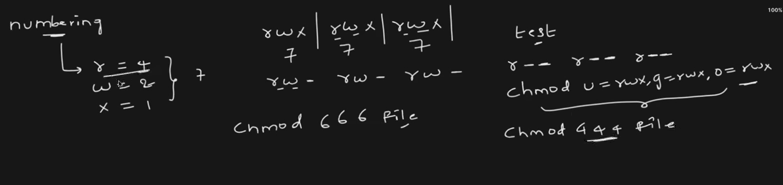
So, Linux supports numbering format.

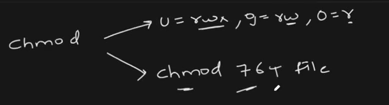
r 🡪 4

w 🡪 2

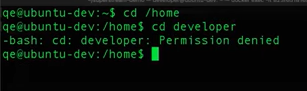
x 🡪 1





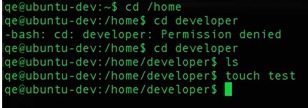


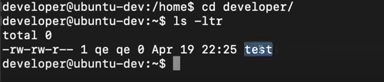
What about directories?









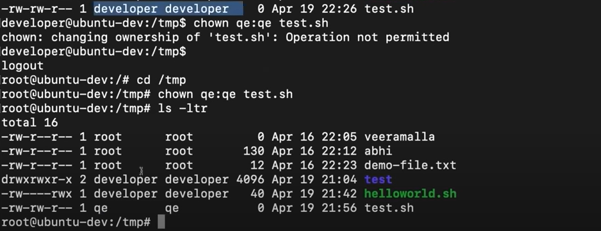


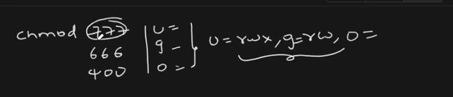
This is how Linux manages File permissions and as well as folder permissions.

Concept is exactly same for file and folders.

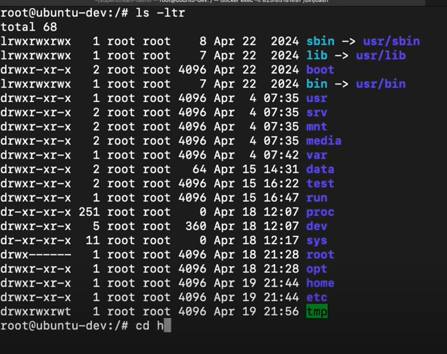
What about ***chown***?

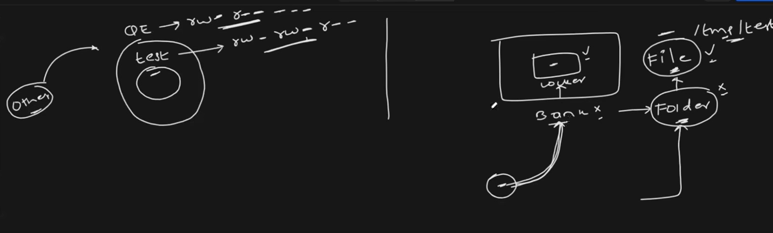






What if there are permissions on folders and there is no permission on file as well.







You might have permissions to a file but if you can’t get into the folder, there is no way you can access the file. So, permissions on the folder takes the highest priority.

Remember Bank and Locker concept 😊!

Try to go through [06-file-permissions](https://github.com/DevenderMusukula/Linux-KT-Related-Documentation/tree/main/06-file-permissions) which has more practical examples as well. Do try it out.