2.3 Management of User and Groups

2.3.1 Create user attributes

To manage users and groups on a Linux server and set attributes such as login names and encrypted passwords, you can use the following commands and steps. These commands are typically executed with root or superuser privileges.

1. Creating a User with login name

To create a user with specific attributes (e.g., login name and password), you can use the `useradd` command.

Example:

→sudo useradd -m -s /bin/bash username

- `-m`: Create the user's home directory.

- `-s`: Specify the shell (e.g., `/bin/bash` for Bash shell).

- `username`: Replace with the desired login name for the user.

2. Setting a Password

After creating the user, you can set an encrypted password using the `passwd` command.

Example:

→sudo passwd username

You will be prompted to enter the password twice. The password is encrypted automatically.

3. Creating Groups

To create a group, use the `groupadd` command.

Example:

→sudo groupadd groupname

- groupname: Replace with the desired name for the group.

4. Adding Users to Groups

You can add users to groups using the **`usermod`** command.

Example:

sudo usermod -aG groupname username

- aG: Adds the user to the group without removing them from other groups.

- groupname: The name of the group.

- username: The name of the user.

5. Viewing User Information

You can check the details of a user and their groups with the `id` command.

Example:

→id username

6. Creating Users with Encrypted Passwords

To create a user with an encrypted password in one step, you can use the **`useradd`** command with the `-p` option, although it's generally recommended to use **`passwd`** for security purposes.

→sudo useradd -m -p $(openssl passwd -crypt 'password') username

- `-p`: Specify the encrypted password.

- `openssl passwd -crypt 'password'`: Generate an encrypted password (replace `'password'` with the actual password).

7. Deleting Users and Groups

To delete a user or group, you can use the `userdel` or `groupdel` commands.

Example (delete a user):

➔sudo userdel -r username

- `-r`: Remove the user's home directory as well.

Example (delete a group):

→sudo groupdel groupname

This process will allow you to create and manage users, assign groups, and set encrypted passwords on a Linux server.

1. **Set Encrypted password**

Encryption is the process of converting information or data into a format that is unreadable by anyone who does not have the proper decryption key or password. It is a fundamental method used to protect sensitive information from unauthorized access.

Key Concepts of Encryption

1. **Plaintext**: The original, readable data or message that needs to be protected (e.g., a password, a message, or a file).
2. **Ciphertext**: The result of encryption. It's a scrambled version of the plaintext that cannot be easily understood by anyone without the key.
3. **Encryption Algorithm**: The mathematical method or process used to transform plaintext into ciphertext. Common encryption algorithms include AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman), and SHA (Secure Hash Algorithm).
4. **Encryption Key**: A secret value (typically a string of numbers or letters) that is used by the encryption algorithm to scramble the data. The same or a different key is needed to decrypt the data back into readable form.
5. **Decryption**: The reverse process of encryption, where ciphertext is converted back into its original, readable form (plaintext) using a key.

Types of Encryptions:

1. **Symmetric Encryption**:
    - Uses the **same key** for both encryption and decryption.
    - Faster but requires that both the sender and receiver share the same key securely.
    - Common algorithms: AES, DES (Data Encryption Standard).

Example:

- You encrypt a message with a password. The person receiving the message must know the same password to decrypt it.

2. **Asymmetric Encryption**:
    - Uses a **pair of keys**: a public key for encryption and a private key for decryption.
    - More secure for transmitting data, as only the recipient's private key can decrypt the message encrypted with their public key.
    - Common algorithms: RSA, ECC (Elliptic Curve Cryptography).

Example:

- You send an encrypted email to someone using their public key. Only they can decrypt it using their private key.

Real-World Uses of Encryption

- **Protecting Passwords**: When you create a password on a website, it is usually encrypted before being stored in the database. This way, even if the database is compromised, the passwords will remain protected.
- **Data Transmission**: When you browse websites using HTTPS, the data exchanged between your browser and the website is encrypted, preventing hackers from reading your sensitive information (e.g., credit card numbers).
- **File Encryption**: Sensitive files can be encrypted so that only authorized individuals with the correct key can access them.

Why Encryption is Important

- **Confidentiality**: Encryption ensures that sensitive information is only accessible to authorized users.
- **Data Integrity**: Encryption can also protect data from being tampered with during transmission.
- **Security**: In many fields, such as online banking, healthcare, and cloud storage, encryption is critical to ensuring the security of digital data.

Encryption plays a vital role in modern cybersecurity and protecting personal, corporate, and government data from unauthorized access

To set an **encrypted password** for a user on a Linux server, you can either use pre-encrypted password strings or let Linux automatically encrypt it via the `passwd` command. Below are several methods you can use to set encrypted passwords.

1. Using the `passwd` Command (Automatic Encryption)

This is the most straightforward and secure method where Linux will handle encryption.

Steps:

1. Create a user (if not already created):

→ **sudo useradd -m username**

2. Set the password for the user (it will prompt you to enter a password, which will be encrypted automatically):

→sudo passwd username

The password is stored in encrypted form in `/etc/shadow`, and the encryption is handled securely by the system.

2. Setting Pre-Encrypted Passwords with `useradd`

If you already have a pre-encrypted password string and want to add a user with that encrypted password, you can do so using the `useradd` command.

Steps:

1. Generate an encrypted password using the `openssl` command:

→openssl passwd -6 'plaintext-password'

- `-6`: Uses SHA-512 encryption.

- **Replace** `'plaintext-password'` with your actual password.

- The output will look like this:

➔6randomSalt$JYjIleNPhHgogSfghGJt.d0tW2tT7jifAUE

2. Create the user and assign the pre-encrypted password:

→ sudo useradd -m -p '6randomSalt$JYjIleNPhHgogSfghGJt.d0tW2tT7jifAUE' username

- `-p`: Passes the encrypted password.

- The system stores the password as-is in encrypted format in `/etc/shadow`.

3. Directly Editing `/etc/shadow`

This is an advanced method, where you manually update the encrypted password in the shadow file.

1. Generate an encrypted password using `openssl`:

→openssl passwd -6 'plaintext-password'

2. Open the `/etc/shadow` file with a text editor:

→ sudo nano /etc/shadow

3. Find the user and replace their current password field with the new encrypted password string.

4. Save and exit the file.

Example of a line in `/etc/shadow`:

username:6randomSalt$JYjIleNPhHgogSfghGJt.d0tW2tT7jifAUE:18693:0:99999:7:::

4. Using `chpasswd` for Bulk User Creation

You can also bulk-create users with encrypted passwords using the `chpasswd` command.

1. First, create a text file with user:encrypted_password pairs, like so:

→username:6randomSalt$JYjIleNPhHgogSfghGJt.d0tW2tT7jifAUE

2. Then, pass this file to the `chpasswd` command:

sudo chpasswd -e < user_password_file.txt

- `-e`: Indicates that the passwords are encrypted.

In Linux, **UID (User ID) and GID (Group ID)** are numerical identifiers associated with user accounts and groups. These IDs are used by the system to manage file ownership, permissions, and other security aspects. Let's explore them in more detail:

1. What is **UID (User ID)**

- The **UID** is a unique number assigned to each user on a Linux system.

- It is used by the system to identify the user and control access to system resources (files, processes, etc.).

- Each user has their own UID, and no two users can have the same UID on the same system.

Common UID Ranges:

- 0: The root user (administrator) has UID 0, which has full control over the system.

- 1-999: Reserved for system users and services (e.g., `daemon`, `syslog`, etc.).

- 1000 and above: Typically assigned to regular users created by system administrators.

Check UID for a User:

You can check the UID of a specific user using the following command:

→id username

This will display the UID and other information about the user.

→id john

Output:

uid=1001(john) gid=1001(john) groups=1001(john),27(sudo)

In this example, user "john" has a UID of `1001`.

2. What is GID (Group ID)?

- The **GID** is a unique number assigned to each group on a Linux system.

- It is used to define group memberships, which in turn control access to resources.

- Each group has its own GID, and users can be part of multiple groups, each with its own GID.

Primary and Secondary Groups:

- Primary Group (GID): Every user has a primary group, which is typically the same as their username. This group is assigned when the user is created.

- Secondary Groups: Users can belong to additional groups, which allow them to access shared resources.

Check GID for a Group:

To check the GID of a specific group, you can use the **`getent`** command or check the **`/etc/group`** file.

→getent group groupname

3. How to Set UID and GID When Creating a User

When creating a new user, you can explicitly set the UID and GID using the `useradd` command.

Example:

→sudo useradd -u 1500 -g 1500 john

- `-u 1500`: Assigns UID 1500 to the user "john".

- `-g 1500`: Assigns GID 1500 to the user "john"'s primary group.

If you don't specify the UID and GID, the system will automatically assign the next available numbers.

4. Viewing UID and GID for Users and Groups

Viewing All Users and Their UIDs:

User information is stored in the **`/etc/passwd`** file. You can view it like this:

→cat /etc/passwd

Sample output:

john:x:1001:1001::/home/john:/bin/bash

Here:

- `john`: Username.

- `x`: Placeholder for the password.

- `1001`: UID.

- `1001`: GID.

- `/home/john`: Home directory.

- `/bin/bash`: Shell.

Viewing All Groups and Their GIDs:

Group information is stored in the **`/etc/group`** file. You can view it like this:

→cat /etc/group

Sample output:

john:x:1001:

Here, **`1001`** is the GID for the group **"john".**

5. Changing a User's UID and GID

To change the UID or GID of an existing user, use the **`usermod`** command.

Changing UID:

→sudo usermod -u 1501 john

Changing GID:

→sudo usermod -g 2000 john

6. File Ownership and Permissions

File ownership and permissions are tied to UIDs and GIDs. Each file has:

- Owner (UID of the user who owns the file).

- Group (GID of the group that owns the file).

To view file ownership:

→ls -l filename

Example output:

-rw-r--r-- 1 john john 1024 Sep 9 12:34 myfile.txt

Here:

- `john john`: First "john" refers to the file's owner **(UID),** and the second "john" refers to the group (GID).

Summary of Commands

-View UID and GID for a user:

→ id username

-View all users and UIDs:

→ cat /etc/passwd

- View all groups and GIDs:

→ cat /etc/group

- Create a user with specific UID and GID:

→ sudo useradd -u UID -g GID username

- Change the UID or GID for an existing user:

→ sudo usermod -u new_UID username

→sudo usermod -g new_GID username

2.3.2 Create group attributes

In Linux, groups are used to manage user permissions and access control. Here's how to manage group attributes, including group name, Group ID (GID), group members, group password, group ownership, group permissions, and group expiration date.

1. GID (Group ID) and name

1. Create a Group

To create a new group with a specified group name and GID, you can use the groupadd command:

Syntax:

→ sudo groupadd -g GID groupname

Example:

→sudo groupadd -g 1002 developers

This creates a group called developers with a GID of 1002.

2. Add Members to a Group

You can add a user to a group using the usermod command:

Syntax:

→ sudo usermod -aG groupname username

Example: sudo usermod -aG developers johndoe

This adds the user johndoe to the developers group. The -aG option appends the user to the group without removing them from other groups.

To verify the members of a group, you can check the /etc/group file:

→cat /etc/group | grep groupname

3. Group Password (Optional)

Linux groups can have passwords, although this is not commonly used. You can set a group password with the gpasswd command:

Syntax:

→*sudo gpasswd groupname*

Example: sudo gpasswd developers

After this, you will be prompted to set a password for the group. Users can join the group by using the newgrp command and providing the password

4. Group Ownership

Group ownership defines which group owns a file or directory. You can set group ownership of a file using the chown command:

Syntax:

→ sudo chown :groupname filename

Example: sudo chown :developers /var/www/html

This changes the group ownership of /var/www/html to the developers group.

5. Group Permissions

You can set group permissions on files and directories using the chmod command. Permissions are represented as rwx (read, write, execute) for users, groups, and others.

Syntax:

→sudo chmod g+rwx filename

Example: sudo chmod g+rw /var/www/html

This grants the group read and write permissions on the /var/www/html directory.

Alternatively, you can use numeric representations:

→sudo chmod 770 /var/www/html

This gives full permissions (rwx) to the owner and group, but no permissions to others.

6. Group Expiration Date

Groups themselves do not have expiration dates, but you can expire or set expiration dates for users in a group using the chage command.

Syntax (to set user expiration):

→ sudo chage -E YYYY-MM-DD username
example: sudo chage -E 2024-12-31 johndoe

This sets the expiration date of the user johndoe to December 31, 2024. After this date, the user will no longer be able to log in.

Summary of Commands:

- **Create group:** sudo groupadd -g GID groupname
- **Add user to group:** sudo usermod -aG groupname username
- **Set group password:** sudo gpasswd groupname
- **Set group ownership:** sudo chown :groupname filename
- **Set group permissions:** sudo chmod g+rwx filename
- **Set user expiration date:** sudo chage -E YYYY-MM-DD username
- **Modify group information:** sudo groupmod -g newGID groupname or sudo groupmod -n newgroupname oldgroupname

These commands give you full control over managing groups and their attributes on a Linux server.

2.3.3 Manage users and groups

Manage Users

1. Create a User

You can create a user account using the useradd command. This command also allows you to specify attributes such as home directory, default shell, and more.

Syntax:

→sudo useradd -m -s /bin/bash username

- -m: Creates a home directory for the user.
- -s /bin/bash: Sets the default shell to Bash.
- username: The name of the new user.

Example: sudo useradd -m -s /bin/bash johndoe

This creates a new user johndoe with a home directory (/home/johndoe) and sets Bash as the default shell.

2. Set a Password for a User

After creating a user, you need to set a password for the account:

Syntax:

→sudo passwd username

Example:

→sudo passwd johndoe

You will be prompted to enter and confirm the password.

3. Modify a User

You can modify user attributes using the usermod command.

Syntax:

→sudo usermod [options] username

- -c: Set a comment (usually the full name of the user).
- -d: Change the user's home directory.
- -s: Change the user's login shell.
- -g: Change the user's primary group.
- -G: Add the user to supplementary groups.

Example:

→sudo usermod -c "John Doe" -s /bin/zsh johndoe

This sets the full name to "John Doe" and changes the default shell to Zsh.

4. Delete a User

You can delete a user and optionally remove the user's home directory using the **userdel** command.

Syntax:

→sudo userdel -r username

-r: Removes the user's home directory and mail spool.

Example: sudo userdel -r johndoe

This deletes the user johndoe and their home directory.

5. List Users

To view all users on the system, you can view the contents of the /etc/passwd file:

→cat /etc/passwd

Managing Groups

1. Create a Group

You can create a new group using the **groupadd** command.

Syntax:

→sudo groupadd groupname

Example: **sudo groupadd developers**

This creates a group called developers.

2. Modify a Group

You can change a group's name or GID using the groupmod command.

Syntax (change GID):

→sudo groupmod -g newGID groupname

Syntax (change group name):

→sudo groupmod -n newgroupname oldgroupname

Example:

→sudo groupmod -g 1050 developers

→sudo groupmod -n devteam developers

This changes the GID of the developers group to 1050 and renames the group to devteam.

3. Delete a Group

You can delete a group using the groupdel command.

Syntax:

→sudo groupdel groupname

Example: **sudo groupdel developers**

This deletes the developers group.

4. Add Users to a Group

You can add a user to a group using the usermod command with the -aG option.

Syntax:

→sudo usermod -aG groupname username

Example: **sudo usermod -aG developers johndoe**

This adds the user johndoe to the developers group.

5. List Groups

To view all groups on the system, you can view the contents of the /etc/group file:

→cat /etc/group

This file contains information about all groups on the system.

6. Check User's Group Membership

To see which groups a user belongs to, use the groups command:

→ groups username
Eample: groups johndoe

This shows all the groups johndoe is a member of.

2.4 Application of files and directories permissions

2.4.1 Description of permission types

What are the three permission groups in Linux?

First, you must think of those nine characters as three sets of three characters (see the box at the bottom). Each of the three "rwx" characters refers to a different operation you can perform on the file.

1. **Owners:** These permissions apply exclusively to the individuals who own the files or directories.

2. **Groups:** Permissions can be assigned to a specific group of users, impacting only those within that particular group.

3. **All Users:** These permissions apply universally to all users on the system, presenting the highest security risk. Assigning permissions to all users should be done cautiously to prevent potential security vulnerabilities.

| --- | --- | --- |
| rwx | rwx | rwx |
| user | group | other |

What are the three kinds of file permissions in Linux?

There are three kinds of file permissions in Linux Read, write, and execute.

Letters	Definition
'r'	"read" the file's contents.
'w'	"write", or modify, the file's contents.
'x'	"execute" the file. This permission is given only if the file is a program.

Symbols: **`+`, `-` and `=`**Option in Linux File Permission

Operators	Definition
`+`	Add permissions
`-`	Remove permissions
`=`	Set the permissions to the specified values

User, group, and others Option in Linux File Permission

Reference	Class	Description
`u`	**user**	The user permissions apply only to the owner of the file or directory, they will not impact the actions of other users.
`g`	**group**	The group permissions apply only to the group that has been assigned to the file or directory, they will not affect the actions of other users.

Reference	Class	Description
`o`	**others**	The other permissions apply to all other users on the system, this is the permission group that you want to watch the most.
`a`	**All three**	All three (owner, groups, others)

Reading the Security Permissions in Linux

For example: "rw- r-x r–"

- **"rw-"**: the first three characters `rw-`. This means that the owner of the file can "read" it (look at its contents) and "write" it (modify its contents). we cannot execute it because it is not a program but a text file.
- **"r-x"**: the second set of three characters "r-x". This means that the members of the group can only read and execute the files.
- **"r–"**: The final three characters "r–" show the permissions allowed to other users who have a UserID on this Linux system. This means anyone in our Linux world can read but cannot modify or execute the files' contents.

2.4.2 Settimg Permissions

Symbolic Notation

How to Change Permissions in Linux?

The command you use to change the security permissions on files is called "chmod", which stands for "change mode" because the nine security characters are collectively called the security "mode" of the file.
An example will make this clearer.

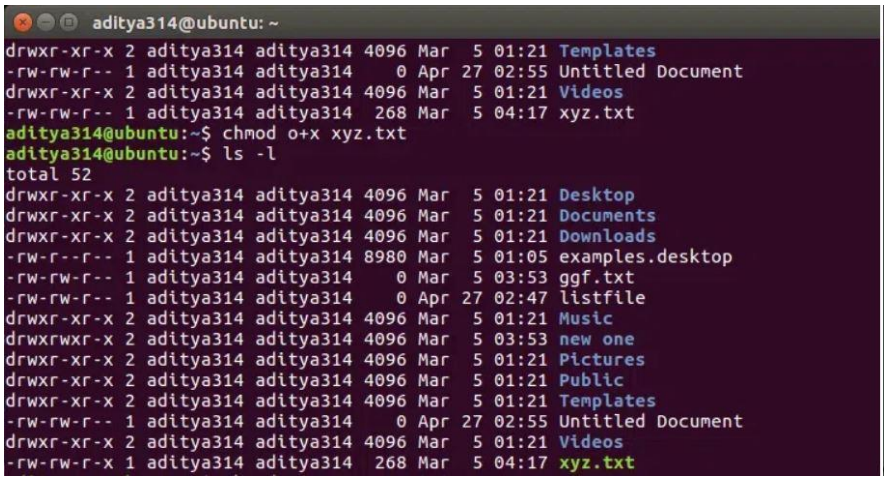
For example, if you want to give "execute" permission to the world ("other") for file "xyz.txt", you will start by typing.

chmod o
Now you would type a '+' to say that you are "adding" permission.

chmod o+
Then you would type an 'x' to say that you are adding "execute" permission.

chmod o+x
Finally, specify which file you are changing.

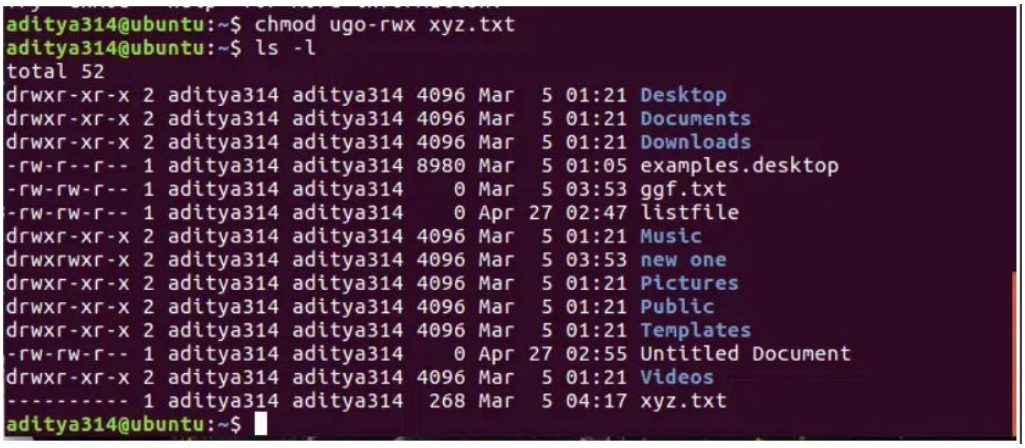
chmod o+x xyz.txt
You can see the change in the picture below.



You can also change multiple permissions at once. For example, if you want to take all permissions away from everyone, you would type.

chmod ugo-rwx xyz.txt

The code above revokes all the read(r), write(w), and execute(x) permission from all user(u), group(g), and others(o) for the file xyz.txt which results in this.



Another example can be this:

chmod ug+rw,o-x abc.mp4

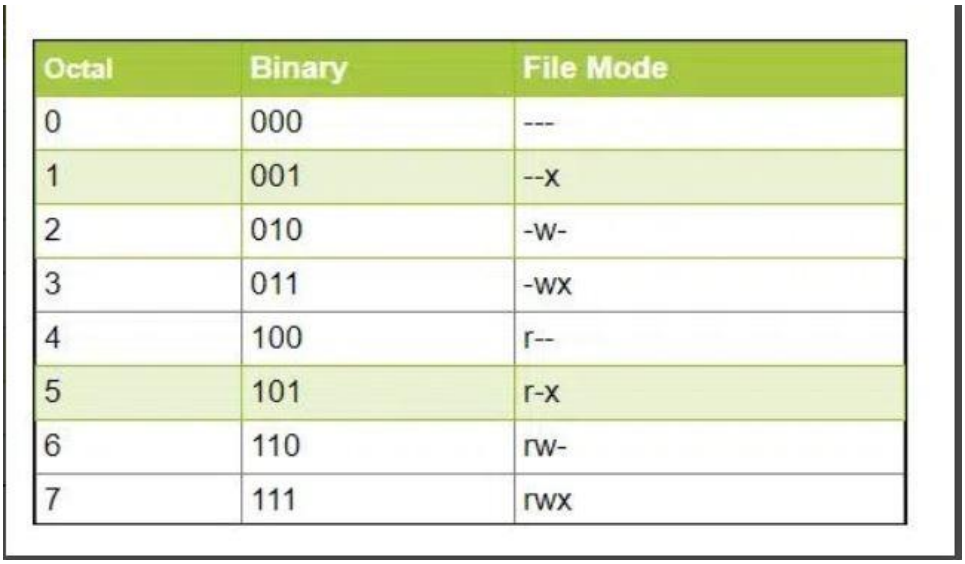
The code above adds read(r) and write(w) permission to both user(u) and group(g) and revoke execute(x) permission from others(o) for the file abc.mp4.

Something like **chmod ug=rx,o+r abc.c** assigns read(r) and execute(x) permission to both user(u) and group(g) and add read permission to others for the file abc.

There can be numerous combinations of file permissions you can invoke revoke and assign. You can try some on your Linux system.

Octal Notation

You can also use octal notations like this.



Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Using the octal notations table instead of 'r', 'w', and 'x'. Each digit octal notation can be used for either of the group 'u', 'g', or'o'.

So, the following work is the same:

chmod ugo+rwx [file_name]
chmod 777 [file_name]

Both of them provide full read write and execute permission (code=7) to all the group.

The same is the case with this.

chmod u=r,g=wx,o=rx [file_name]
chmod 435 [file_name]

Both the codes give read (code=4) user permission, write and execute (code=3) for the group and read and execute (code=5) for others.

And even this…

chmod 775 [file_name]
chmod ug+rwx,o=rx [file_name]

Both the commands give all permissions (code=7) to the user and group, read and execute (code=5) for others.

Exercise

Q1: If we want to grants read, write, and execute permissions to the owner, and read and execute permissions to the group and others.

➔ **chmod 755 filename**

Q2: I change file permissions for multiple files at once?

➔ **chmod 644 *.txt**
Q3: If we want to changes the owner to “newowner” and the group to “newsgroup.”

➔ chown newowner:newgroup filename

Common Commands:

- **chmod**: Change permissions.
- **chown**: Change the owner of a file.
- **chgrp**: Change the group of a file.
- **ls -l**: View file permissions in symbolic form.

2.4.3 Perform permissions testing in Linux server

To perform permissions testing on a Linux server, you can follow these steps:

1. Check Existing Permissions

You can view the current permissions of a file or directory by using the ls -l command:

→ ls -l filename

This will display the permissions in symbolic notation, along with the owner, group, and other details of the file.

Example output:

→ -rw-r--r-- 1 user group 4096 Sep 14 10:00 filename

In this case:

- Owner: rw- (read, write)
- Group: r-- (read-only)
- Others: r-- (read-only)

2. Testing Permission Changes Using Symbolic Notation

Step 1: Create a test file

Start by creating a test file for permission changes:

→touch testfile.txt

Step 2: Modify file permissions using chmod

You can now test permission changes with symbolic notation.

Example 1: Give the owner execute permissions.

→ chmod u+x testfile.txt

→ls -l testfile.txt

This will add execute permission for the owner (rwx), keeping the others as read-only.

Example 2: Remove write permissions for the group.

→ chmod g-w testfile.txt

→ls -l testfile.txt

3. Testing Permission Changes Using Octal Notation

You can also test permissions using octal notation:

Step 1: Set full permissions for the owner, and read-only for the group and others

→chmod 744 testfile.txt

→ls -l testfile.txt

Output:

→ -rwxr--r-- 1 user group 4096 Sep 14 10:00 testfile.txt

4. Test File Access

You can now test access based on the permissions you've set:

Step 1: Switch to a different user

Use the su command to switch to another user or sudo to execute commands as another user.

Step 2: Attempt to read, write, or execute the file

Example: Trying to read a file without read permissions.

→ chmod 200 testfile.txt # Owner can only write, no read permissions

→cat testfile.txt

This should produce a "Permission denied" error.

Example: Trying to execute a file without execute permissions.

→chmod 644 testfile.sh # No execute permissions

→./testfile.sh

This should produce a "Permission denied" error.

5. **Test Directory Permissions**

You can test directory permissions in the same way as files. Directories require x (execute) permission to allow users to access the files within.

Example 1: Create a test directory and file

→mkdir testdir

→touch testdir/file.txt

Example 2: Remove execute permission from the directory for others

→chmod o-x testdir

→ls -ld testdir

Others will now be unable to access the directory contents:

→ **cd testdir** # "Permission denied"

Example 3: Give read and execute permissions for group and others

→chmod 755 testdir

→ls -ld testdir

End of Learning Unit 2.