File Permission Setup

**2 - File Permission:**

Task: Create a file/folder and give it the fine-grained access to it. Research which user need how much permissions. Document the all the process and your research.

# Step 1: Create a Folder

**Objective:** Create a folder named "my\_folder" as the target for fine-grained access control.

## Open Terminal:

 Launch your terminal by pressing

**Ctrl + Alt + T**

## Create a Folder:

on most Linux systems.

Execute the following command to create a folder named "my\_folder":

mkdir my\_folder

## Explanation:

 The

**mkdir**

command is used to create a new directory.

 "my\_folder" is the name of the directory we want to create.

## Expected Output:

 No explicit output is generated for a successful directory creation. You can

verify by executing to list the contents of the current directory.



**ls**

# Step 2: Fine-Grained Access Control

**Objective:** Implement fine-grained access control by setting specific permissions for owner, group, and others.

File and folder permissions are managed through three entities: owner, group, and

**chmod**

others. Permissions are assigned using the

## Owner Permissions:

command.

The owner has the most control over the file/folder. They can read, write, and execute. Execute the following command to give full control to the owner:

chmod 700 my\_folder

 This command sets read (4), write (2), and execute (1) permissions for the owner, and no permissions for group and others.

## Explanation:

is a command used to change file/folder permissions.  "700" is the numeric representation in octal notation.

bashCopy code

chmod 500 my\_folder

**chmod**

## Expected Output:

 No explicit output is generated upon success.

## Group Permissions:

 The group associated with the file/folder can be assigned specific permissions. To give read and execute permissions to the group, execute:

 This command sets read (4) and execute (1) permissions for the group, and no permissions for the owner and others.

## Explanation:

 The second digit in the numeric representation represents group permissions.

## Expected Output:

 No explicit output is generated upon success.

## Others Permissions:

 Permissions for users who are not the owner or in the group can be specified. To give read-only access to others, execute:

chmod 400 my\_folder

 This command sets read (4) permission for others, and no permissions for the owner and group.

## Explanation:

 The third digit in the numeric representation represents others' permissions.

## Expected Output:

 No explicit output is generated upon success.

# Step 3: Verify Permissions

**Objective:** Confirm that the permissions have been set as intended.

## Check Permissions:

 Execute the following command to verify the permissions:

ls -l

## Explanation:

 The

**ls -l**

command lists detailed information about files/folders in the

current directory, including permissions.

## Expected Output:

You will see output similar to the following:This line indicates the permissions for "my\_folder," where "user" is the owner, "group" is the associated group, and the permissions are set according to the specified values.

drwx------ 2 user group 4096 Jan 6 12:00 my\_folder

By following these steps, you can create a folder with fine-grained access control, understand numeric representation and octal notation, and learn how to verify and change file permissions on a Linux system.

**Research on File Permissions:**

# Numeric Representation:

File permissions in Linux can be represented numerically using a concise system where each permission type is assigned a numeric value. The numeric values are assigned as follows:

 Read: 4

 Write: 2

 Execute: 1

These values can be added together to grant multiple permissions. For example:  Read + Write (4 + 2) = 6 (Read and Write permissions)

 Read + Write + Execute (4 + 2 + 1) = 7 (Read, Write, and Execute permissions)

Numeric representation simplifies the expression of permissions, making it easier to understand and manipulate them using numeric values.

# File Permission Categories:

File permissions are categorized into three groups:

1. **Owner:** The user who owns the file or folder.
2. **Group:** The group associated with the file or folder.
3. **Others:** Users who are not the owner or in the associated group.

The command is used to modify these permissions. Permissions can be set

**chmod**

independently for each category, allowing fine-grained control over who can read, write, and execute a file or folder.

# Octal Notation:

Octal notation is a compact representation of file permissions using three digits. Each digit represents the permission value for owner, group, and others, respectively. The order is always owner, group, and others. The values are calculated based on the sum of read (4), write (2), and execute (1) permissions. For example:

 Read + Write + Execute (4 + 2 + 1) = 7

 Read + Execute (4 + 1) = 5

 Write only (2) = 2

Using octal notation simplifies the setting of permissions, providing a concise way to express the desired access levels.

# Default Permissions:

When a new file or folder is created, default permissions are assigned based on the system's umask value. The umask value determines which permissions are subtracted from the default permissions. For example, if the umask is set to 022, the default permissions for a file would be 644 (666 - 022), granting read and write permissions to the owner and read-only permissions to group and others.

Understanding default permissions is crucial for ensuring security and controlling access to newly created files and folders.

# Changing Ownership:

The command is used to change the ownership of a file or folder. This

**chown**

command allows users to specify a new owner and, if needed, a new group. For example:

chown new\_owner:new\_group my\_folder

This command changes the owner to "new\_owner" and the group to "new\_group," providing flexibility in managing file and folder ownership for effective access control.

In summary, these aspects of file permissions and ownership provide a foundational understanding of how access is controlled and managed in a Linux environment.