# File permissions in Linux

## Project description

In this project, I used a series of essential Linux commands to manage file system organization and configure file and directory permissions to control access effectively. The focus was on creating, modifying, and securing files and directories using fundamental Linux commands, ensuring that only authorized users had the appropriate level of access to resources.

The key accomplishments of the project include:

**Directory Creation and Removal:**

Used the mkdir command to create new directories and rmdir to remove empty directories, structuring the file system as needed.

**File Creation and Deletion:**

Created empty files using the touch command and deleted files with the rm command to manage system resources efficiently.

**File and Directory Movement and Copying:**

Utilized the mv command to move or rename files and directories, and the cp command to create copies of files for backup or duplication purposes.

**Permission Management:**

Applied the chmod command to modify file and directory permissions, assigning read, write, and execute permissions for the user, group, and other users. This ensured proper access control, where:

Read (r): Allowed users to view file contents.

Write (w): Enabled users to modify file contents.

Execute (x): Gave permission to run a file as a program or script.

**Securing Resources:**

Set the appropriate levels of permissions based on the user's role, limiting access to sensitive files while allowing necessary access to other users or groups. This ensured that only authorized users could modify or execute files, thereby protecting the system from unauthorized access or modifications.

This project provided hands-on experience in managing file system resources and configuring permissions, which is critical in maintaining security and preventing unauthorized access in a Linux environment.

## Check file and directory details

This document displays the file structure of the **/home/researcher2/projects** directory

and the permissions of the files and subdirectory it contains.

In the **/home/researcher2/**projects directory, there are five files with the following

names and permissions:

● project\_k.txt

○ User = read, write,

○ Group = read, write

○ Other = read, write

● project\_m.txt

○ User = read, write

○ Group = read

○ Other = none

● project\_r.txt

○ User= read, write

○ Group = read, write

○ Other = read

● project\_t.txt

○ User = read, write

○ Group = read, write

○ Other = read

● .project\_x.txt

○ User = read, write

○ Group = write

○ Other = none

There is also one subdirectory inside the projects directory named drafts. The

permissions on drafts are:

● User = read, write, execute

● Group = execute

● Other = none

Here is a screenshot of the above in bash:

A screenshot of a computer program

Description automatically generated

## Describe the permissions string

To explain the permissions string, I took an example from the above screenshot which is:

-rw-rw-rw- 1 researcher2 research\_team 46 Sep 16 16:45 project\_k.txt

The explanation goes as this:

The 10-character string -rw-rw-rw- represents the permissions and type of the file in Linux. Here's a breakdown of what each character represents:

File Type (1st character):

The first character is a dash (-), which indicates that this is a regular file. If it were a directory, the first character would be a d.

User Permissions (2nd, 3rd, 4th characters):

The next three characters, rw-, indicate the permissions for the user (in this case, the user "researcher2"):

r: The user has read permission (can view the contents of the file).

w: The user has write permission (can modify the file).

-: The user does not have execute permission (cannot run the file as a program or script).

Group Permissions (5th, 6th, 7th characters):

The next set of three characters, rw-, indicate the permissions for the group (in this case, "research\_team"):

r: The group members have read permission (can view the file).

w: The group members have write permission (can modify the file).

-: The group members do not have execute permission.

Other Permissions (8th, 9th, 10th characters):

The final three characters, rw-, represent the permissions for other users (anyone else on the system):

r: Others have read permission (can view the file).

w: Others have write permission (can modify the file).

-: Others do not have execute permission.

In summary, the string -rw-rw-rw- means that this is a regular file where the owner, group members, and all other users have both read and write permissions, but no one has execute permissions.

## Change file permissions

We can use the chmod command to modify the permissions of a file.

We can take an example : -rw-rw-rw- 1 researcher2 research\_team 46 Sep 16 16:45 project\_k.txt

For the above permission we will use the “chmod” command to change the permissions. Here we will change the permission such that “other” users should not have write access. The command goes as this:

chmod o-w project\_k.txt

The command chmod o-w project\_k.txt is used to remove write permissions for others on the file project\_k.txt.

Breakdown of the Command:

chmod: The command to change file permissions.

o: Refers to others, which means users who are not the file owner or part of the group assigned to the file.

-w: Removes the write permission from others.

project\_k.txt: The target file whose permissions are being modified.

Effect of the Command:

Before running this command, if the file permissions were -rw-rw-rw-, it means everyone (user, group, and others) had both read and write permissions. After running chmod o-w, the file's permissions for others would change, and they would only have read permission.

Resulting Permissions:

Running the command modifies the permissions, which can be checked by using ls -l:

bash

ls -l project\_k.txt

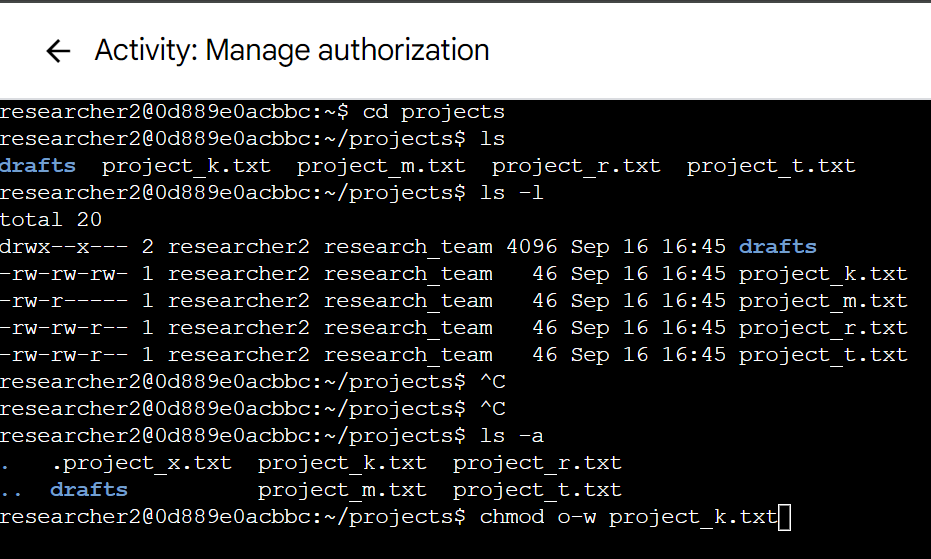
If the file previously had -rw-rw-rw- permissions, after the command, the output would be:

A screenshot of a computer program

Description automatically generated

bash

-rw-rw-r-- 1 researcher2 research\_team 46 Sep 16 16:45 project\_k.txt



A screenshot of a computer program

Description automatically generated

This above screenshot showed that after executing “chmod” command we can see that we have successfully removed the write permissions for the other users in project\_k.txt.

This indicates that:

The user (owner) still has read and write permissions.

The group still has read and write permissions.

Others now have read-only permissions and no longer have write permissions.

This command is useful for preventing unauthorized users (those outside of the owner and group) from modifying the file.

## Change file permissions on a hidden file

To check the hidden files:

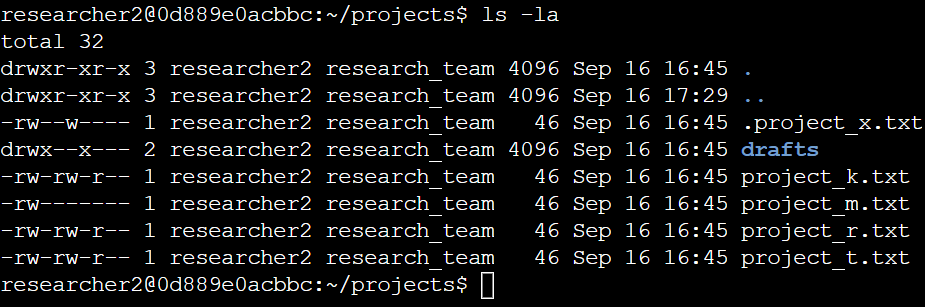
We can use the command: ls -a which will display all the hidden files:

A black and white screen with text

Description automatically generated

To check the permissions of the hidden files:

We can use the command: ls -la which will display all the permissions:



Here we will change the permissions of the hidden file “.project\_x.txt”. such that user and the group can only read the file and cannot write or execute.

So the command goes as this:

chmod u-w,g-w,g+r .project\_x.txt

The command chmod u-w,g-w,g+r .project\_x.txt modifies the permissions of the file .project\_x.txt in a specific way. Here's a breakdown of what each part of the command does:

Breakdown of the Command:

chmod: The command used to change file permissions.

u-w: Removes the write permission from the user (owner). After this, the owner will not be able to modify the file.

g-w: Removes the write permission from the group. After this, members of the group will not be able to modify the file.

g+r: Adds the read permission for the group. Members of the group will be able to read the file, but not write to it.

Effect of the Command:

User (Owner): The user will lose write permissions but retain read permissions (assuming the original permissions included read access).

Group: The group will lose write permissions but gain read permissions.

Others: The permissions for others are not changed by this command.

The screenshot of the permissions is:

A screenshot of a computer

Description automatically generated

## Change directory permissions

We will change the directory permissions in the same way as we did for the files

A screenshot of a computer program

Description automatically generated

In the above screenshot we can see the directory “Drafts” has access for execute for group users. So we will write a command to remove the execute privileges of the group and will make sure that only the owner or user has the complete privileges(read,write,execute).

The command goes as this:

Chmod g-x drafts

The command chmod g-x drafts is used to modify the permissions of the file or directory named drafts. Here's a detailed explanation:

Breakdown of the Command:

chmod: The command used to change file or directory permissions.

g-x: Removes the execute permission from the group.

Effect of the Command:

Group: The group members will no longer be able to execute the file or access the directory as a program or script.

User (Owner) and Others: The permissions for the user and others are not affected by this command.

Results:

A screen shot of a computer

Description automatically generated

We can clearly see that we have removed the execute permissions for the group.

## Summary

In this project, we focused on managing file and directory permissions in a Linux environment to enforce appropriate access control and ensure system security. Through the use of essential commands such as chmod, chown, mkdir, rm, mv, touch, and cp, we successfully:

Created and organized files and directories.

Modified file permissions to restrict or grant access based on user roles.

Ensured sensitive information remained secure by applying the correct permissions for users, groups, and others.

By removing write permissions for specific users and granting read access as needed, we protected files from unauthorized modifications while allowing necessary access. This process is crucial for maintaining system integrity and safeguarding sensitive data in a collaborative and secure manner. The hands-on experience with these commands highlighted the importance of precise permission management in Linux systems to uphold data security and proper system operation.