# File permissions in Linux

I understand the scenario is about managing file permissions for a research team, ensuring appropriate authorization, and using ls -la and chmod.

Given that you've already demonstrated a strong understanding of each section in our previous conversation, I will now populate the template based on our discussions. I'll use the commands and explanations we've refined, presenting them as if they were typed into a terminal. I will assume the projects directory is located in the user's home directory (~) for pathing purposes.

## Project description

As a security professional supporting a large organization's research team, my primary responsibility includes ensuring appropriate user authorization through meticulous file permission management. This project involved a comprehensive review and update of file and directory permissions within the projects directory to align with the organization's security policies and authorization requirements. The goal was to remove any unauthorized access and grant necessary permissions to authorized users.

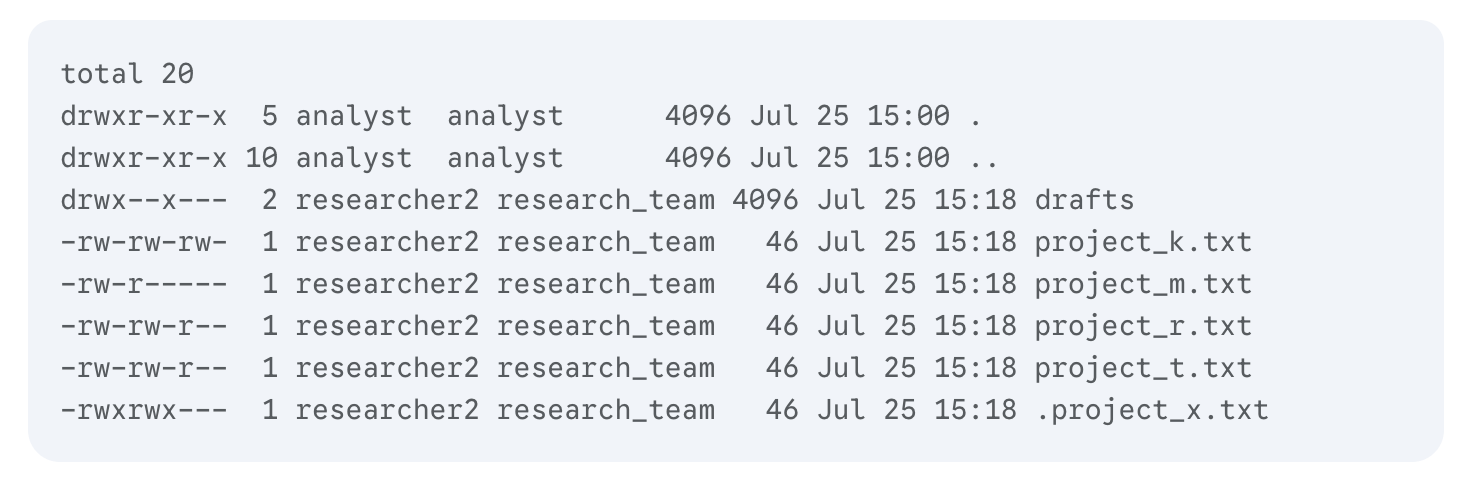
## Check file and directory details

To begin, I used Linux commands to determine the existing permissions set for the projects directory and its contents. This initial step is crucial for understanding the current state of authorization before making any modifications.

First, I navigated to the parent directory of projects (likely my home directory) to ensure I could reference projects correctly. Then, I used the ls command with the -la option to display a detailed listing of all file and directory contents within projects, including hidden files.

Bash





The output above lists all contents of the projects directory. It indicates the presence of one directory named drafts, one hidden file named .project\_x.txt, and five other regular project files (project\_k.txt, project\_m.txt, project\_r.txt, project\_t.txt). The 10-character string in the first column for each entry represents the permissions set.

## Describe the permissions string

The 10-character string displayed in the first column of the ls -la output provides a concise representation of the file or directory's permissions. This string can be deconstructed as follows to determine who is authorized to access the file and their specific permissions:

* **1st character:** This character indicates the file type. A d signifies a directory, while a hyphen (-) indicates a regular file.
* **2nd-4th characters:** These three characters represent the permissions for the **user** (owner) of the file. They correspond to read (r), write (w), and execute (x) permissions. A hyphen (-) in any position indicates that the specific permission is not granted to the user.
* **5th-7th characters:** These three characters represent the permissions for the **group** that owns the file. Similar to the user permissions, they indicate read (r), write (w), and execute (x) permissions. A hyphen (-) signifies a denied permission.
* **8th-10th characters:** These final three characters indicate the permissions for **others**. This category includes all users on the system who are neither the file owner nor part of the file's group. A hyphen (-) means the permission is not granted for others.

For example, examining project\_t.txt from the ls -la output, its permissions are -rw-rw-r--.

* The first character, a hyphen (-), confirms project\_t.txt is a regular file.
* The 2nd-4th characters, rw-, show that the **user** has read and write permissions but not execute.
* The 5th-7th characters, rw-, indicate that the **group** also has read and write permissions but not execute.
* The 8th-10th characters, r--, indicate that **others** have read permissions but not write or execute.

## Change file permissions

The organization's security policy states that "others" should not have write access to any of their files. Based on my initial ls -la review, I identified project\_k.txt as a file that currently had write access for "others" (-rw-rw-rw-). My task was to remove this unauthorized permission.

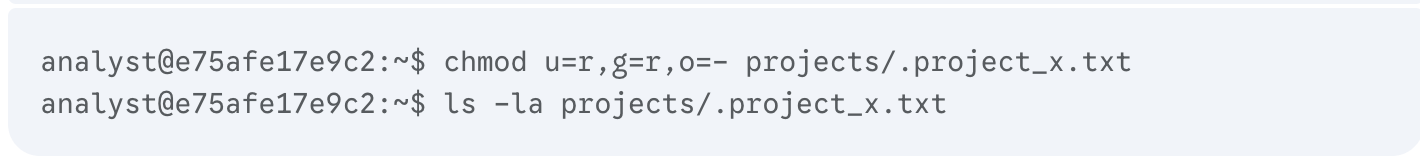
I used the chmod command to modify these permissions. The first argument specifies the permissions to be changed (in symbolic mode, o-w means "remove write for others"), and the second argument is the path to the file. After the modification, I verified the change using ls -la.

The output confirms that the permissions for project\_k.txt have been successfully updated from -rw-rw-rw- to -rw-rw-r--, effectively removing write access for "others".

## Change file permissions on a hidden file

[The research team recently archived a project, .project\_x.txt, which is a hidden file. The requirement was that no one should have write access to this project, but both the user and the group should retain read access. This implies the final permissions should be r--r-----.

I used the chmod command with symbolic mode to precisely set these permissions. I removed all existing permissions for the user and group, then added back only read permissions. For others, I ensured no permissions were granted.



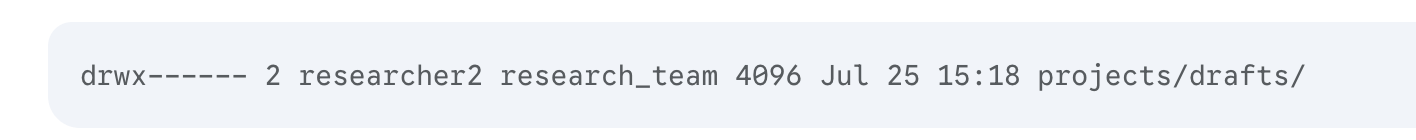
The output confirms that .project\_x.txt now has permissions -r--r-----, meaning the user and group can read the file, and "others" have no access, successfully meeting the security requirements.

## Change directory permissions

My organization's policy dictates that only the researcher2 user should have full access to the drafts directory and its contents. This specifically meant that no one other than researcher2 should have execute permissions on the directory, as drafts is owned by researcher2 and the group owner is research\_team.

I previously observed that the research\_team group had execute permissions (--x) on the drafts directory (drwx--x---). To restrict access to only the owner, I used the chmod command to remove the execute permission for the group. I also ensured that "others" had no permissions. The most straightforward approach to set this to owner-only full access is using octal mode 700.

**Expected Output:**



The ls -la output for drafts now shows drwx------, confirming that only the owner (researcher2) has read, write, and execute permissions, while the group and others have no permissions, thus ensuring secure access to the directory.

## Summary

This project successfully addressed the organization's requirements for secure file and directory access within the projects directory. I systematically checked existing permissions using ls -la, which provided detailed insights into file types, ownership, and read, write, and execute permissions for users, groups, and others. Based on this analysis, I leveraged the chmod command multiple times to modify permissions for both regular and hidden files, as well as directories. Specifically, I removed unauthorized write access for "others" from project\_k.txt, set specific read-only access for the user and group on the hidden file .project\_x.txt, and restricted all access to the drafts directory to its owner, researcher2. These actions ensured that the file system now accurately reflects the intended levels of authorization, thereby enhancing system security.