# File permissions in Linux

## Project description

The research team at my organization needed updated file permissions for specific files and directories within the ***projects***directory. Several items had overly permissive access, putting sensitive research data at risk.

To ensure the system followed the **principle of least privilege,** I checked the current permissions, analyzed which users required access and applied appropriate modifications.

This project demonstrates using Linux commands to:

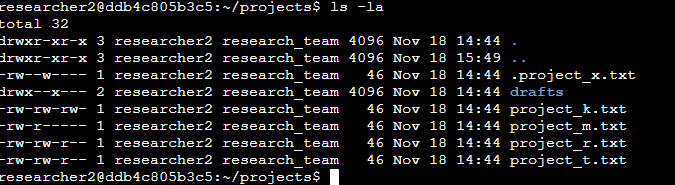
* Inspect permissions
* Analyze ownership
* Set proper access levels
* Verify secure configuration

## Check file and directory details

Linus command in use: ls -la

Purpose:

* Display hidden files
* Show long-format details of permissions, ownership, size, etc
* Identify insecure permissions such as 777, 757, or world-writable flags



The output shows several entries:

* ***projects*** and ***researcher2*** representing the current and parent directories
* A hidden file named ***.projects\_x.txt***
* A subdirectory called ***drafts***
* Multiple project files ending in \_k.txt, \_m.txt, *r.txt* and *\_*t.txt

The first column of each line contains the 10-character permission string, which details whether the item is a file or directory and how access is assigned to the owner, group and other users. Reviewing these details helped me identify locations where permissions needed tightening.

## Describe the permissions string

Each item in the directory listing begins with a 10-character permissions string. This sequence tells Linux how the file or folder can be accessed and by whom.

The first character indicates the type of object:

* D means it is a directory
* - means it is a regular file

The remaining nine characters are grouped into three sets of three, representing access levels for:

1. The file owner
2. the assigned group
3. and all other users

Within each group, the characters show whether read (r), write (w) or execute (x) privileges are granted. A dash (-) indicates that a particular permission is not allowed.

For example, a string like:

drwxr-x---

tells us that:

* the owner can read, write and execute
* members of the group can read and execute
* and all others have no access at all

Breaking down these permission strings helped me determine which items required stricter access controls.

## Change file permissions

After reviewing the initial permission settings, several files and directories were configured with access levels that exceeded what the research team needed. Overly permissive settings can increase the risk of accidental edits, unauthorized modifications, or data exposure, so my goal was to align each item with the principle of least privilege.

I updated the permissions as follows:

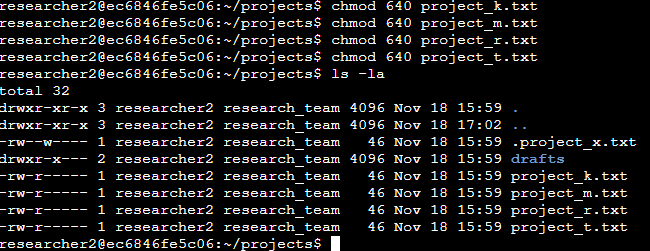
A screenshot of a computer

AI-generated content may be incorrect.

Using ***chmod 750 drafts/*** I applied more controlled access to the ***drafts*** directory so that only the file owner and designated group members could view or modify its contents, while all other users were completely blocked. This change prevents unauthorized users from browsing or executing anything inside the directory.

Several project files had write access granted more broadly than required. To prevent unintended edits and maintain data integrity, I narrowed the permissions so that:

* the owner retains the ability to read and write,
* the group may read but not modify,
* and others have no access.



This ensures sensitive content can be reviewed by the appropriate team members but not altered without proper authorization.

The hidden file ***.project\_x.txt*** contained notes not intended for broad distribution. To prevent unauthorized visibility or manipulation, I restricted it further:

A screenshot of a computer program

AI-generated content may be incorrect.

This allows only the file owner to read or modify the file – a best-practice setting for files containing configuration details or sensitive documentation.

To make permission enforcement meaningful, I also verified that files were assigned to the correct user and group. Where needed, I updated ownership to ensure that only authorized team members could inherit group-based access:

A screenshot of a computer

AI-generated content may be incorrect.

This ensures the group permissions actually apply to the intended audience.

These modifications eliminate unnecessary write access, reduce the chance of unauthorized access to research data and enforce a predictable permission model across the directory. By tightening these settings:

* sensitive files become harder to tamper with
* data integrity is better preserved
* internal access aligns with team roles
* and the attack surface for privilege misuse shrinks significantly

This demonstrates a practical application of least privilege, access control and system hardening – core concepts in real-world cybersecurity.

## Change file permissions on a hidden file

To protect archived project notes, the research team requested that the hidden file ***.project\_x.txt*** be set to read-only for both the file owner and the research group. No users should be able to modify the file, but it must remain visible to those who need to reference its contents.

The command I used to update the permissions was as follows:

A screenshot of a computer program

AI-generated content may be incorrect.

In Linux, any file beginning with (.) is treated as hidden, so ***.project\_x.txt*** does not appear in a standard directory listing unless the ***-a*** option is used. The updated permissions remove write access from both the owner (u-w) and the group (g-w) and ensure the group still retains read access (g+r).

This change locks down the file so it cannot be altered accidentally or intentionally, while still lllowing the appropriate team members to read its contents.

## Change directory permissions

The drafts directory contained early-stage research notes that only the primary researcher should be able to access. While other members of the research group could view the directory under the previous configuration, the team requested that only ***researcher2*** retain execution rights. Execution (x) is required to enter a directory, so removing this permission effectively blocks navigation into it.

To apply this restriction, I used ***chmod g-x drafts*** with the following results:

A screenshot of a computer program

AI-generated content may be incorrect.

This removes the execute permission from the group while leaving the owners access intact. After running the command I verified the updated permission with ***ls -la***. The output confirmed that the directory now prevents group members from entering or interacting with its contents. This ensures that only the designated researcher can access the draft material, further enforcing strict control over sensitive work-in-progress.

## Summary

Throughout this task, I reviewed and updated a variety of file and directory permissions within the projects directory to align with my organization’s access requirements. By examining the existing configuration with ***ls -la***, I identified several instances where permissions were broader than necessary. I then used Linux permission-modification commands such as ***chmod*** to remove excessive privileges, restrict write access and limit directory navigation to authorized users.

These changes strengthened the security posture of the environment by preventing unauthorized edits, protecting archived materials and ensuring that sensitive directories are only accessible to the appropriate individuals. The updated permission structure now accurately reflects the principle of least privilege and supports safer handling of research data across the team.