File Permissions in Linux

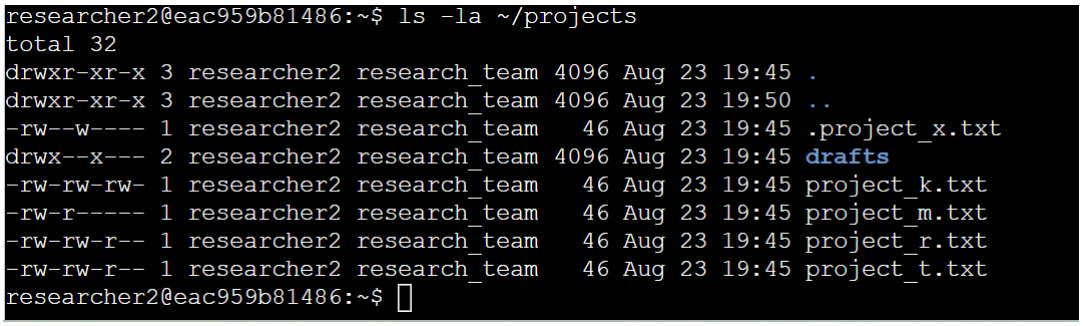
Project Description

I'm supporting the research team's Linux environment, where file permissions needed to be brought in line with the organization's security policy. Several files and a subdirectory inside `~/projects` had permission settings that were more open than they should be, including world-writable files and a directory accessible to users outside the owner. Using `ls -la` and `chmod`, I audited the existing permissions, identified where they didn't match the required access level, and corrected each one.

Check File and Directory Details

To view the permissions on everything inside the `projects` directory, hidden files included, I ran:

```
ls -la
```



The `-l` flag produces the long-format listing (permissions, owner, group, size, timestamp), and `-a` ensures nothing hidden is skipped, since files that start with a period don't show up by default. Running this against `~/projects` revealed one subdirectory (`drafts`), one hidden file (`.project_x.txt`), and four other regular files. Each entry's permissions are represented as a 10-character string in the leftmost column of the output.

Describe the Permissions String

That 10-character string breaks down into four parts:

- **Character 1** — file type: d for directory, - for a regular file.
- **Characters 2–4** — the owner's read/write/execute permissions.
- **Characters 5–7** — the group's read/write/execute permissions.
- **Characters 8–10** — permissions for everyone else on the system ("other").

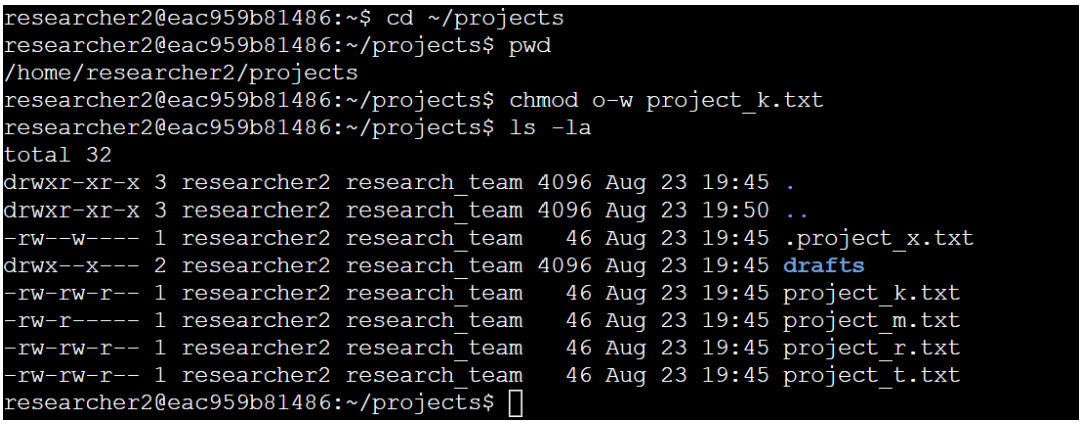
In each of those three-character blocks, a letter means the permission is granted; a hyphen (-) means it's absent.

Taking `project_r.txt` (`-rw-rw-r--`) as an example: the leading hyphen confirms it's a regular file. The owner and group sections (`rw-`, `rw-`) show both have read and write access but not execute. The final block (`r--`) shows other users can read the file but can't write to or execute it.

Change File Permissions

Per policy, no file should be writable by "other." Reviewing the permission strings from the previous step, `project_k.txt` was the only file where other still had write access (`-rw-rw-rw-`). I corrected it with:

```
chmod o-w project_k.txt
ls -la
```

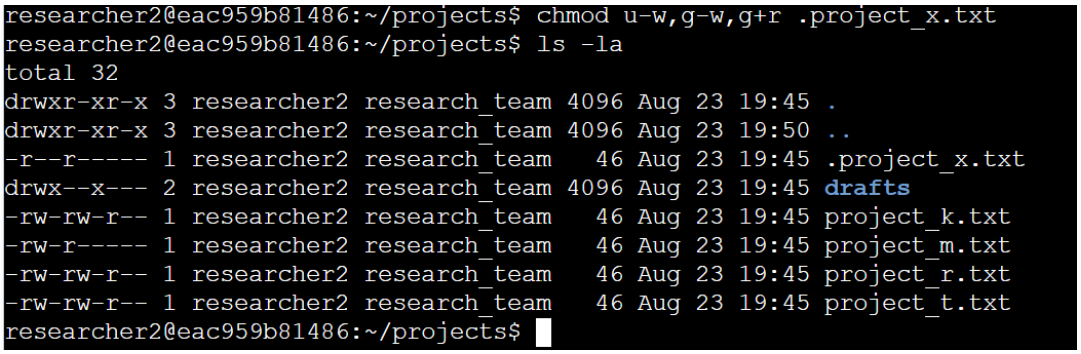


`chmod` modifies access permissions; `o-w` targets the "other" category specifically and removes the write bit, leaving user and group permissions untouched. Re-running `ls -la` confirmed the change — `project_k.txt` now reads `-rw-rw-r--`.

Change File Permissions on a Hidden File

`.project_x.txt` has been archived by the research team, and the requirement was that no one retain write access, while the owner and group should still be able to read it. Starting from `-rw--w----`, the owner had write but the group had write with no read at all — neither matched what was needed. I ran:

```
chmod u-w,g-w,g+r .project_x.txt
ls -la
```

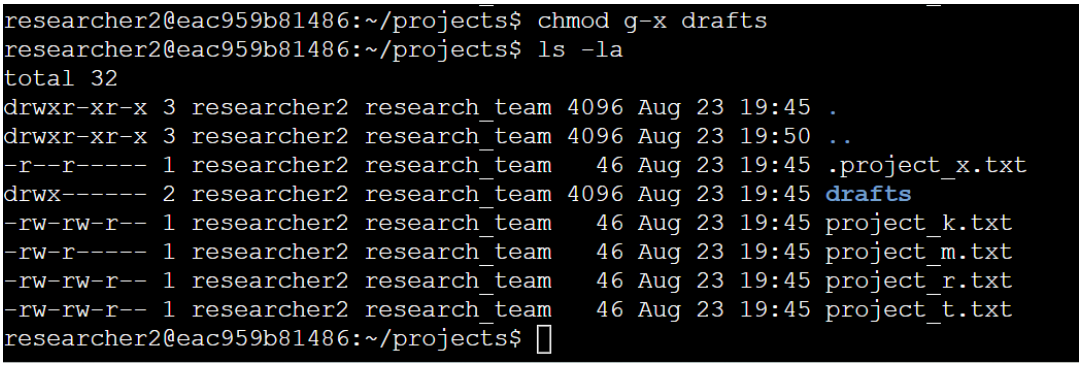


This removes write from the owner (`u-w`), removes write from the group (`g-w`), and grants read to the group (`g+r`) in a single command. The result, `-r--r-----`, gives the owner and group read-only access while eliminating write permissions entirely, satisfying the requirement. I recognized `.project_x.txt` as a hidden file because its name begins with a period.

Change Directory Permissions

Access to `drafts` was meant to be restricted to the owner, `researcher2`, alone. The directory's starting permissions, `drwx--x---`, still gave the group execute access, meaning group members could enter it. Since other already had no access, group's execute bit was the only thing left to remove:

```
chmod g-x drafts
ls -la
```



The result, `drwx------`, confirms that only the owner retains any access to the directory — group and other are both fully locked out.

Summary

Working through `~/projects`, I used `ls -la` to establish a baseline of the existing permissions, then compared each file and directory against the organization's access policy. Three gaps stood out: `project_k.txt` allowed outside users to write to it, the archived `.project_x.txt` didn't match the read/write policy set for it, and the `drafts` directory was accessible to group members who shouldn't have had entry. I used targeted `chmod` commands to close each gap, removing only the specific permissions that violated policy rather than resetting files wholesale, and confirmed each fix with a follow-up `ls -la`. The result is a permission set that reflects least-privilege access across the directory.