# File permissions in Linux

## Project description

My organization's research team required an update to file permissions for specific files and directories within the project’s directory. The current permissions did not align with the necessary authorization levels. This effort was crucial to enhance system security. To achieve this, I undertook the following actions:

## Check file and directory details

## Below is the code exemplifying my utilization of Linux commands to ascertain the current permissions configured for a particular directory within the file system

## Screenshot of the command line with the command for this task and its output

## The screenshot's initial line showcases the command I input, while the subsequent lines exhibit the resulting output. The provided code enumerates all elements within the project’s directory. To achieve this, I employed the ls command, incorporating the -la option to furnish an exhaustive directory listing, encompassing concealed files. The outcome of my command discerns the presence of one directory labeled "drafts," one concealed file denominated ".project\_x.txt," and five additional project files. Within the first column of each line, you will observe a 10-character string signifying the permissions allocated to every file or directory.

## Describe the permissions string

The 10-character string can be analyzed to discern authorized access and specific permissions for the file. Here's a breakdown of the characters and their representations:

1. 1st character: Either "d" (directory) or "-" (regular file).
2. 2nd-4th characters: User's permissions (read "r," write "w," execute "x," or "-").
3. 5th-7th characters: Group's permissions (read "r," write "w," execute "x," or "-").
4. 8th-10th characters: Other users' permissions (read "r," write "w," execute "x," or "-").

For instance, consider the file permissions for "project\_t.txt," which are "-rw-rw-r--." The first character "-" indicates it's a file, not a directory. The second, fifth, and eighth characters "rw-" signify that the user, group, and other have read permissions but lack execute permissions. The third and sixth characters "rw-" indicate that only the user and group have write permissions while others do not.

## Change file permissions

The organization's security policy required revoking write access for "other" on all files. To achieve this, I referred to the previously retrieved file permissions. Upon examination, I identified that "project\_k.txt" needed to have its write access for "other" removed.

Below is the code illustrating the Linux commands I employed to implement this change:



In the screenshot, you'll find the initial two lines showcasing the commands I input, while the subsequent lines exhibit the output of the second command. The "chmod" command plays a pivotal role in altering permissions for both files and directories. The first argument dictates which permissions are to be modified, and the second argument specifies the targeted file or directory.

In this particular case, I opted to withdraw write permissions from the "other" category for the "project\_k.txt" file. Following this adjustment, I employed the "ls -la" command to thoroughly inspect the revisions I implemented.

Top of Form

Top of Form

## Change file permissions on a hidden file

The research team in our organization recently archived "project\_x.txt." Their requirement was to revoke write access for all users except the user and group, who should retain read access.

Here's a demonstration of the Linux commands I employed to modify the permissions:



The initial two lines in the screenshot showcase the commands I input, while the subsequent lines exhibit the output from the second command. I discerned that ".project\_x.txt" is concealed because of its leading period (.) in the filename. In this instance, I rescinded write permissions for both the user and group and supplemented read permissions for the group. Specifically, I eliminated write permissions for the user using "u-w." Subsequently, I withdrew write permissions for the group employing "g-w" and introduced read permissions for the group through "g+r."

## Change directory permissions

My organization's exclusive requirement is to grant access to the "researcher2" user for the "drafts" directory and its contents. This implies that execute permissions should be restricted solely to "researcher2."

The subsequent code exemplifies the Linux commands I employed to modify the permissions:



## In the provided screenshot, the initial two lines showcase the commands I inputted, while the subsequent lines exhibit the output of the second command. Having previously ascertained that the group possessed execute permissions, I utilized the "chmod" command to eliminate those permissions. The "researcher2" user, having already been granted execute permissions, necessitated no further adjustments.

## Summary

I adjusted various permissions to align with my organization's desired authorization level for files and directories within the projects directory. Initially, I employed "ls -la" to inspect the directory's permissions, which served as a foundation for subsequent actions. Subsequently, I employed the "chmod" command repeatedly to modify permissions on files and directories.