# File permissions in Linux

## Project description:

The research team at my organization needs to adjust the file permissions for specific files and directories within the `projects` directory, as the current permissions do not align with the appropriate levels of authorization. Updating these permissions is essential to maintaining system security. To address this, I carried out the following tasks:

## Check file and directory details:

The following code demonstrates how I used Linux commands to determine the existing permissions set for a specific directory in the file system.

```
researcher2@b2b856d75dbb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Oct 24 12:57 .
drwxr-xr-x 3 researcher2 research_team 4096 Oct 24 13:06 ..
-rw--w---- 1 researcher2 research_team   46 Oct 24 12:57 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Oct 24 12:57 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Oct 24 12:57 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Oct 24 12:57 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_t.txt
researcher2@b2b856d75dbb:~/projects$
```

The first line of the screenshot displays the command I entered, and the other lines display the output. The code lists all contents of the `projects` directory. I used the `ls` command with the `-la` option to display a detailed listing of the file contents that also returned hidden files. The output of my command indicates that there is one directory named `drafts`, one hidden file named `.project_x.txt`, and five other project files. The 10-character string in the first column represents the permissions set on each file or directory.

## Describe the permissions string:

A 10-character string begins each entry and indicates how the permissions on the file are set. For instance, a directory with full permissions for all owner types would be `drwxrwxrwx`:

- **The 1st character indicates**: the file type. The d indicates it's a directory. When this character is a hyphen (`-`), it's a regular file.
- **The 2nd-4th characters**: indicate the read (`r`), write (`w`), and execute (`x`) permissions for the user. When one of these characters is a hyphen (`-`) instead, it indicates that this permission is not granted to the user.
- **The 5th-7th characters**: indicate the read (`r`), write (`w`), and execute (`x`) permissions for the group. When one of these characters is a hyphen (`-`) instead, it indicates that this permission is not granted for the group.
- **The 8th-10th characters**: indicate the read (`r`), write (`w`), and execute (`x`) permissions for the owner type of other. This owner type consists of all other users on the system

apart from the user and the group. When one of these characters is a hyphen (`-`) instead, that indicates that this permission is not granted for other.

For example, the file permissions for `project_t.txt` are `-rw-rw-r--`. Since the first character is a hyphen (`-`), this indicates that `project_t.txt` is a file, not a directory. The second, fifth, and eighth characters are all `r`, which indicates that user, group, and other all have read permissions. The third and sixth characters are `w`, which indicates that only the user and group have write permissions. No one has execute permissions for `project_t.txt`.

## Change file permissions:

The organization determined that other shouldn't have write access to any of their files. To comply with this, I referred to the file permissions that I previously returned. I determined `project_k.txt` must have the write access removed for other.

The following code demonstrates how I used Linux commands to do this:

```
researcher2@b2b856d75dbb:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Oct 24 12:57 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Oct 24 12:57 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Oct 24 12:57 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_t.txt
researcher2@b2b856d75dbb:~/projects$ chmod o-w project_k.txt
researcher2@b2b856d75dbb:~/projects$ chmod g-r project_m.txt
researcher2@b2b856d75dbb:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Oct 24 12:57 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_k.txt
-rw------- 1 researcher2 research_team   46 Oct 24 12:57 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_t.txt
researcher2@b2b856d75dbb:~/projects$
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. The `chmod` command changes the permissions on files and directories. The first argument indicates what permissions should be changed, and the second argument specifies the file or directory. In this example, I removed write permissions from other for the `project_k.txt` file. After this, I used `ls -la` to review the updates I made.

## Change file permissions on a hidden file:

The research team at my organization recently archived `project_x.txt`. They do not want anyone to have write access to this project, but the user and group should have read access.

The following code demonstrates how I used Linux commands to change the permissions:

```
researcher2@b2b856d75dbb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Oct 24 12:57 .
drwxr-xr-x 3 researcher2 research_team 4096 Oct 24 13:06 ..
-rw--w---- 1 researcher2 research_team   46 Oct 24 12:57 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Oct 24 12:57 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_k.txt
-rw------- 1 researcher2 research_team   46 Oct 24 12:57 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_t.txt
researcher2@b2b856d75dbb:~/projects$ chmod ug-w,g+r .project_x.txt
researcher2@b2b856d75dbb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Oct 24 12:57 .
drwxr-xr-x 3 researcher2 research_team 4096 Oct 24 13:06 ..
-r--r----- 1 researcher2 research_team   46 Oct 24 12:57 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Oct 24 12:57 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_k.txt
-rw------- 1 researcher2 research_team   46 Oct 24 12:57 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_t.txt
researcher2@b2b856d75dbb:~/projects$
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. I know `.project_x.txt` is a hidden file because it starts with a period (`.`). In this example, I removed write permissions from the user and group, and added read permissions to the group. I removed write permissions from the user with `u-w`. Then, I removed write permissions from the group with `g-w`, and added read permissions to the group with `g+r`

## Change directory permissions:

My organization only wants the `researcher2` user to have access to the `drafts` directory and its contents. This means that no one other than `researcher2` should have execute permissions.

The following code demonstrates how I used Linux commands to change the permissions:

```
researcher2@b2b856d75dbb:~/projects$ ls
drafts  project_k.txt  project_m.txt  project_r.txt  project_t.txt
researcher2@b2b856d75dbb:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Oct 24 12:57 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_k.txt
-rw------- 1 researcher2 research_team   46 Oct 24 12:57 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_t.txt
researcher2@b2b856d75dbb:~/projects$ chmod g-x drafts
researcher2@b2b856d75dbb:~/projects$ ls -l
total 20
drwx------ 2 researcher2 research_team 4096 Oct 24 12:57 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_k.txt
-rw------- 1 researcher2 research_team   46 Oct 24 12:57 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 24 12:57 project_t.txt
researcher2@b2b856d75dbb:~/projects$
```

The output here displays the permission listing for several files and directories. Line 1 indicates the current directory (projects), and line 2 indicates the parent directory (home). Line 3 indicates a regular file titled `.project_x.txt`. Line 4 is the directory (drafts) with restricted permissions. Here you can see that only researcher2 has execute permissions. It was previously determined that the group had execute permissions, so I used the `chmod` command to remove them. The `researcher2` user already had execute permissions, so they did not need to be added.

## Summary:

In this project, I gained hands-on experience using fundamental Linux Bash shell commands to manage and configure file and directory permissions. Key aspects of the project included:

- **Examining Permissions**: I utilized commands such as `ls -l` and `ls -la` to review and analyze file, hidden files and directory permissions, understanding the distinctions between user, group, and others.
- **Changing File Permissions**: Using commands like `chmod`, I effectively modified file permissions to control access and editing rights, ensuring appropriate levels of security and functionality based on user roles.
- **Managing Directory Permissions**: I practiced altering directory permissions, focusing on allowing or restricting access to directory contents using both symbolic and numeric methods of permission management.