# File permissions in Linux

## Project description

The research team within our organization is in the process of enhancing the file permissions for specific files and directories located within the projects directory. The current permissions don't align with the desired level of authorization that needs to be granted. This effort is crucial to maintaining the security of their system. In order to accomplish this task, I undertook the following steps:

## Check file and directory details

researcher2@772cbccd429a:~$ ls -la

total 32

drwxr-xr-x 3 researcher2 research\_team 4096 Aug 12 00:13 .

drwxr-xr-x 1 root root 4096 Aug 11 23:18 ..

-rw------- 1 researcher2 research\_team 36 Aug 12 00:18 .bash\_history

-rw-r--r-- 1 researcher2 research\_team 220 Apr 18 2019 .bash\_logout

-rw-r--r-- 1 researcher2 research\_team 3574 Aug 11 23:18 .bashrc

-rw-r--r-- 1 researcher2 research\_team 3574 Aug 11 23:18 .profile

drwxr-xr-x 3 researcher2 research\_team 4096 Aug 11 23:18 projects

## Describe the permissions string

The permissions string in Linux consists of ten characters that represent the access permissions for a file or directory. These characters are divided into three groups: user, group, and others. Each group consists of three characters that indicate read (r), write (w), and execute (x) permissions, respectively.

Here's the breakdown of the permissions string:

User Permissions (First 3 Characters):

First character: Represents the owner's read permission (r) or hyphen (-) if not allowed.

Second character: Represents the owner's write permission (w) or hyphen (-) if not allowed.

Third character: Represents the owner's execute permission (x) or hyphen (-) if not allowed.

Group Permissions (Next 3 Characters):

Fourth character: Represents the group's read permission (r) or hyphen (-) if not allowed.

Fifth character: Represents the group's write permission (w) or hyphen (-) if not allowed.

Sixth character: Represents the group's execute permission (x) or hyphen (-) if not allowed.

Others Permissions (Last 3 Characters):

Seventh character: Represents others' read permission (r) or hyphen (-) if not allowed.

Eighth character: Represents others' write permission (w) or hyphen (-) if not allowed.

Ninth character: Represents others' execute permission (x) or hyphen (-) if not allowed.

## Change file permissions

Upon reviewing the existing file permissions, the organization concluded that it is essential to restrict write access for "others" on all their files. To align with this security requirement, I revisited the previously obtained file permissions. After careful consideration, I identified that the file "project\_k.txt" specifically needed to have its write access revoked for the "others" category.

researcher2@772cbccd429a:~/projects$ chmod o-w project\_k.txt

researcher2@772cbccd429a:~/projects$ ls -la

total 32

drwxr-xr-x 3 researcher2 research\_team 4096 Aug 11 23:18 .

drwxr-xr-x 3 researcher2 research\_team 4096 Aug 12 00:13 ..

-rw--w---- 1 researcher2 research\_team 46 Aug 11 23:18 .project\_x.txt

drwx--x--- 2 researcher2 research\_team 4096 Aug 11 23:18 drafts

-rw-rw-r-- 1 researcher2 research\_team 46 Aug 11 23:18 project\_k.txt

-rw-r----- 1 researcher2 research\_team 46 Aug 11 23:18 project\_m.txt

-rw-rw-r-- 1 researcher2 research\_team 46 Aug 11 23:18 project\_r.txt

-rw-rw-r-- 1 researcher2 research\_team 46 Aug 11 23:18 project\_t.txt

## Change file permissions on a hidden file

At my organization, the research team recently decided to archive "project\_x.txt." As part of this archival process, they intended to restrict write access to this project for all users except the owner and the group. To achieve this, I employed Linux commands to modify the permissions. The code snippet below illustrates the steps I took:

researcher2@772cbccd429a:~/projects$ chmod u-w,g-w,g+r .project\_x.txt

researcher2@772cbccd429a:~/projects$ ls -la

total 32

drwxr-xr-x 3 researcher2 research\_team 4096 Aug 11 23:18 .

drwxr-xr-x 3 researcher2 research\_team 4096 Aug 12 00:13 ..

-r--r----- 1 researcher2 research\_team 46 Aug 11 23:18 .project\_x.txt

drwx--x--- 2 researcher2 research\_team 4096 Aug 11 23:18 drafts

-rw-rw-r-- 1 researcher2 research\_team 46 Aug 11 23:18 project\_k.txt

-rw-r----- 1 researcher2 research\_team 46 Aug 11 23:18 project\_m.txt

-rw-rw-r-- 1 researcher2 research\_team 46 Aug 11 23:18 project\_r.txt

-rw-rw-r-- 1 researcher2 research\_team 46 Aug 11 23:18 project\_t.txt

## Change directory permissions

At my organization, the objective is to grant exclusive access to the "drafts" directory and its contents for the user "researcher2." As part of this access control, it is required that only "researcher2" possesses execute permissions for the directory. To accomplish this, I utilized Linux commands to modify the permissions accordingly. The provided code snippet outlines the sequence of commands I employed:

researcher2@772cbccd429a:~/projects$ chmod g-x drafts

researcher2@772cbccd429a:~/projects$ ls -la

total 32

drwxr-xr-x 3 researcher2 research\_team 4096 Aug 11 23:18 .

drwxr-xr-x 3 researcher2 research\_team 4096 Aug 12 00:13 ..

-r--r----- 1 researcher2 research\_team 46 Aug 11 23:18 .project\_x.txt

drwx------ 2 researcher2 research\_team 4096 Aug 11 23:18 drafts

-rw-rw-r-- 1 researcher2 research\_team 46 Aug 11 23:18 project\_k.txt

-rw-r----- 1 researcher2 research\_team 46 Aug 11 23:18 project\_m.txt

-rw-rw-r-- 1 researcher2 research\_team 46 Aug 11 23:18 project\_r.txt

-rw-rw-r-- 1 researcher2 research\_team 46 Aug 11 23:18 project\_t.txt

## Summary

I aligned various permissions to align with the desired level of authorization stipulated by my organization for files and directories within the "projects" directory. The initial phase of this process involved employing the "ls -la" command to assess the existing permissions of the directory. This evaluation guided my subsequent actions in the process. Subsequently, I utilized the "chmod" command on multiple occasions to systematically modify permissions for individual files and directories.