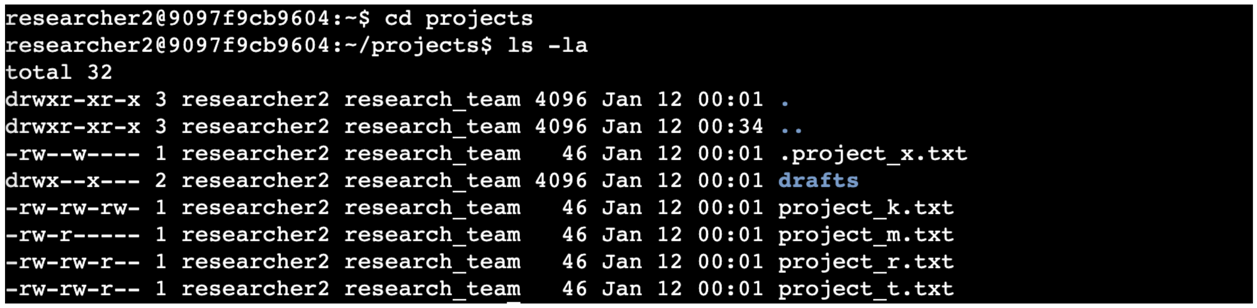
File Permissions in Linux

Project description

In this scenario, I've been tasked with modifying the permissions for files and directories in the ***project*** directory using the CLI in Linux Bash.

Check file and directory details

The following code demonstrates my ability to use Linux commands to outline the current permissions for a particular directory in the file system:



```
researcher2@9097f9cb9604:~$ cd projects
researcher2@9097f9cb9604:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jan 12 00:01 .
drwxr-xr-x 3 researcher2 research_team 4096 Jan 12 00:34 ..
-rw--w---- 1 researcher2 research_team   46 Jan 12 00:01 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jan 12 00:01 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jan 12 00:01 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jan 12 00:01 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 12 00:01 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jan 12 00:01 project_t.txt
```

This ***ls -la*** command lists detailed information about files and directories in the current directory for the user ***researcher2***. The output includes file permissions, ownership, group associations, file sizes, and modification timestamps. For example, the ***project_k.txt*** file has ***rw-rw-r--*** permissions, meaning it can be read and written by the owner and group, and read-only for others. Hidden files (like ***.project_x.txt***) and directories (like . and ..) are also shown, with varying permissions set for access control.

Describe the permissions string

The permissions string (ex., -rw-rw-r--) is a 10-character representation that specifies the type of file and its access permissions for the owner, group, and others. Here's a breakdown:

File Type (1st character):

- **-** indicates a regular file.
- **d** indicates a directory.

Owner Permissions (2nd-4th characters):

- **r** (read): The owner can read the file.
- **w** (write): The owner can modify the file.
- **x** (execute): The owner can execute the file if it's a program or script.
- **-** means the permission is not granted.

Group Permissions (5th-7th characters):

- Same as the owner permissions but apply to users in the file's group.

Others Permissions (8th-10th characters):

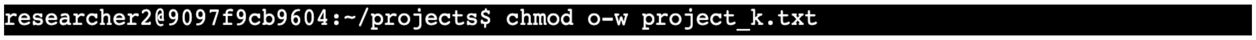
- Same as the owner and group permissions but apply to all other users.

Example:
For -rw-rw-r--:

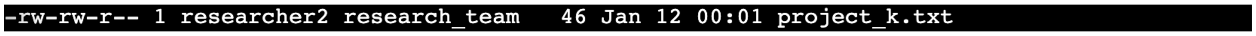
- -: Regular file.
- rw-: The owner can read and write but cannot execute.
- rw-: Group members can read and write but cannot execute.
- r--: Others can only read.

Change file permissions

It was then determined that the ***other*** group shouldn't be allowed write access. The following code handles that:



```
researcher2@9097f9cb9604:~/projects$ chmod o-w project_k.txt
```

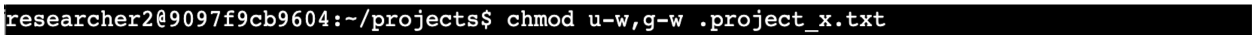


```
-rw-rw-r-- 1 researcher2 research_team   46 Jan 12 00:01 project_k.txt
```

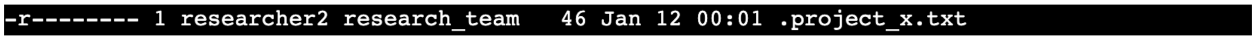
The command ***chmod o-w project_k.txt*** modifies the permissions of the file ***project_k.txt*** by removing the write permission for "others" (users who are not the owner or part of the file's group). The ***chmod*** command is used to change file permissions, where o stands for "others" and ***-w*** specifies the removal of the write permission. After running this command, users outside the file's owner and group will still be able to read the file (if read permission is set) but will no longer be able to modify or write to it, enhancing the file's security.

Change file permissions on a hidden file

It was determined that no one should have write access but the group should have read access to ***.project_x.txt***. The following code handles that (the following code is incorrect. Should be ***chmod u-w, g=r .project_x.txt***):



```
researcher2@9097f9cb9604:~/projects$ chmod u-w,g-w .project_x.txt
```

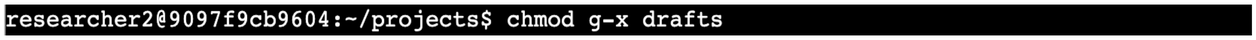


```
-r-------- 1 researcher2 research_team   46 Jan 12 00:01 .project_x.txt
```

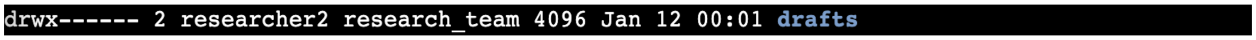
The command ***chmod u-w, g=r .project_k.txt*** modifies the permissions of the file ***.project_k.txt***. It removes the write permission for the file owner (u-w) and sets the group permissions to read-only (g=r). The chmod command allows for simultaneous changes to multiple permission categories, separated by commas. After executing this command, the file owner will no longer be able to modify the file, while members of the group will have read access only. Permissions for others will remain unchanged. This command is often used to enforce stricter access control over a file.

Change directory permissions

It was determined that researcher2 alone should have access to the drafts directory. This code handles that:



```
researcher2@9097f9cb9604:~/projects$ chmod g-x drafts
```



```
drwx------ 2 researcher2 research_team 4096 Jan 12 00:01 drafts
```

The command ***chmod g-x drafts*** removes the execute permission for the group on the drafts directory. The chmod command is used to modify file or directory permissions, where ***g*** specifies the group and ***-x*** removes the execute permission. By removing this permission, group members will no longer be able to navigate into the drafts directory, even if they have read or write permissions for it, only allowing researcher2 to have access.

Summary

I adjusted several permissions to align with my organization's desired authorization levels for files and directories within the projects directory. To begin, I used the ***ls -la*** command to review the existing permissions, which guided my decisions. Based on this information, I utilized the ***chmod*** command multiple times to modify the permissions for the relevant files and directories.