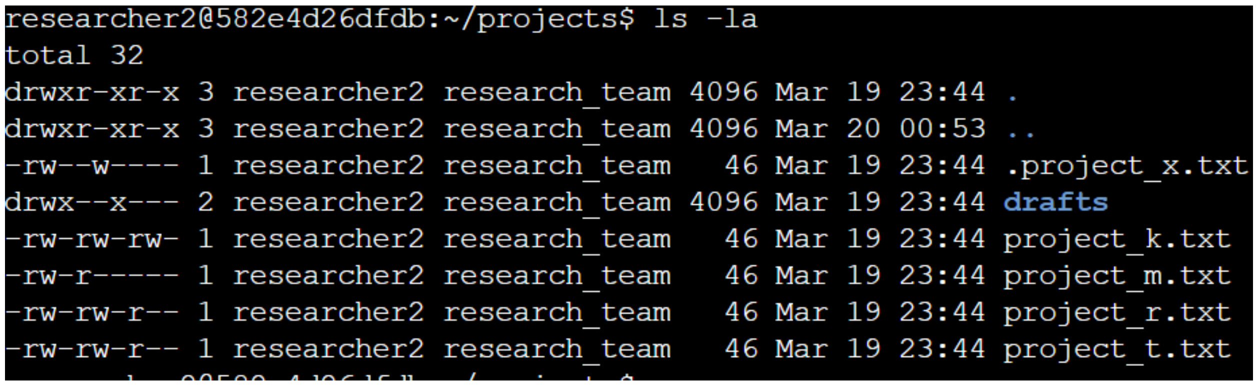
File permissions in Linux

Project description

The research team at my organization needs to update the file permissions for specific files and directories within the projects directory. The current permissions do not accurately reflect the appropriate level of authorization. Reviewing and adjusting these permissions will enhance system security. To accomplish this, I carried out the following tasks:

Check file and directory details

The following code illustrates how I utilized Linux commands to check the current permission settings for a specific directory in the file system.



```
researcher2@582e4d26dfdb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Mar 19 23:44 .
drwxr-xr-x 3 researcher2 research_team 4096 Mar 20 00:53 ..
-rw--w---- 1 researcher2 research_team   46 Mar 19 23:44 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Mar 19 23:44 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Mar 19 23:44 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Mar 19 23:44 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_t.txt
```

The first line of the screenshot shows the command I entered, while the subsequent lines present the output. The command lists all contents of the `projects` directory. I used the `ls` command with the `-la` option to generate a detailed listing, including hidden files. The output reveals one directory named `drafts`, a hidden file called `.project_x.txt`, and five other project files. The 10-character string in the first column represents the permissions assigned to each file or directory.

Describe the permissions string

The 10-character string can be broken down to determine access permissions for a file or directory. Each character represents specific information about the file type and the permissions assigned to different users. Here's what each section signifies:

- **1st character:** Indicates the file type. A d means it's a directory, while a hyphen (-) represents a regular file.

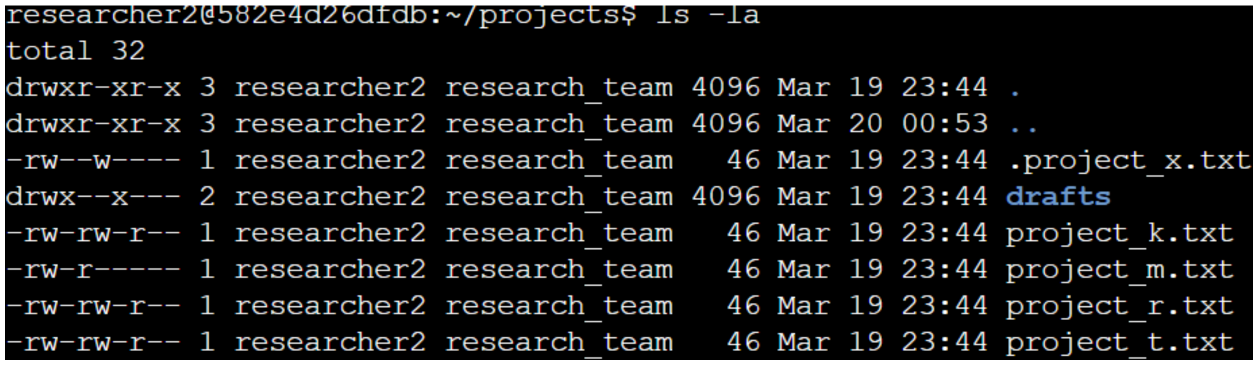
- **2nd-4th characters:** Represent the permissions for the **user** (owner) of the file. These include read (`r`), write (`w`), and execute (`x`). A hyphen (`-`) in place of a character means that permission is not granted.
- **5th-7th characters:** Represent the permissions for the **group** associated with the file, following the same `r`, `w`, and `x` format. A hyphen (`-`) means the corresponding permission is not given.
- **8th-10th characters:** Represent the permissions for **others** (all users who are neither the owner nor part of the group). These follow the same structure, with a hyphen (`-`) indicating a missing permission.

For instance, the file permissions for the file “project_k.txt” are -rw-rw-rw-. Since the first character is a hyphen that indicates that “project_k.txt” is a file. Additionally, the 2nd, the 5th and the 7th characters are r which reveals that all groups (user, group, other) all have read permissions. As for the 3rd, the 6th and the 9th character they are also all w which means that all groups have write permissions. It seems that no one has execute permissions in file “project_k.txt”.

Change file permissions

The organization decided that "others" should not have write access to any files. To ensure compliance, I reviewed the previously retrieved file permissions and identified that **project_k.txt** had write access enabled for others.

The following code illustrates how I used Linux commands to remove this write permission:

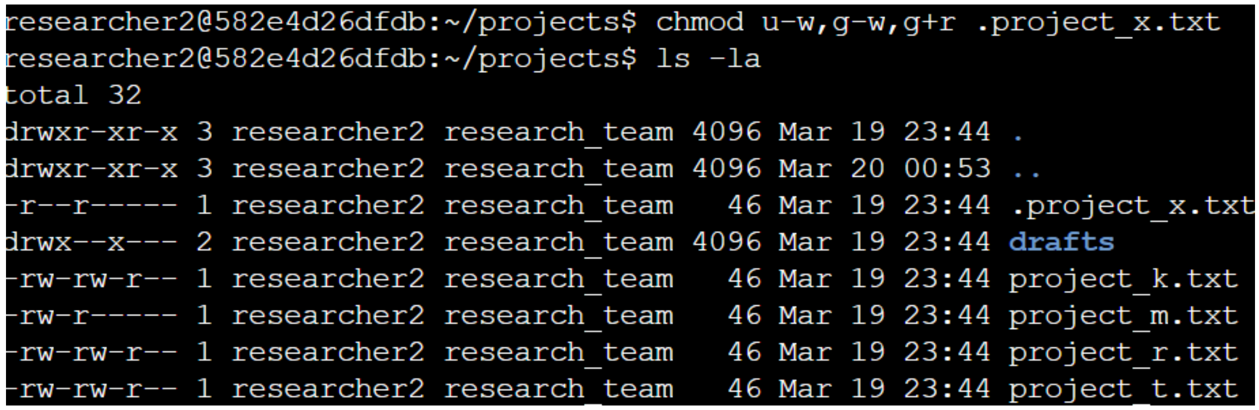


```
researcher2@582e4d26dfdb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Mar 19 23:44 .
drwxr-xr-x 3 researcher2 research_team 4096 Mar 20 00:53 ..
-rw--w---- 1 researcher2 research_team   46 Mar 19 23:44 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Mar 19 23:44 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Mar 19 23:44 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_t.txt
```

The first two lines of the screenshot show the commands I entered, while the remaining lines display the output from the second command. The `chmod` command modifies the permissions on files and directories. The first argument specifies the permissions to be changed, and the second argument identifies the file or directory. In this example, I removed write permissions for "other" on the `project_k.txt` file. Afterward, I used `ls -la` to verify the changes I made.

Change file permissions on a hidden file

The research team at my organization recently archived `project_x.txt` and decided that no one should have write access to this project. However, the user and group should retain read access. The following code demonstrates how I used Linux commands to modify the permissions:

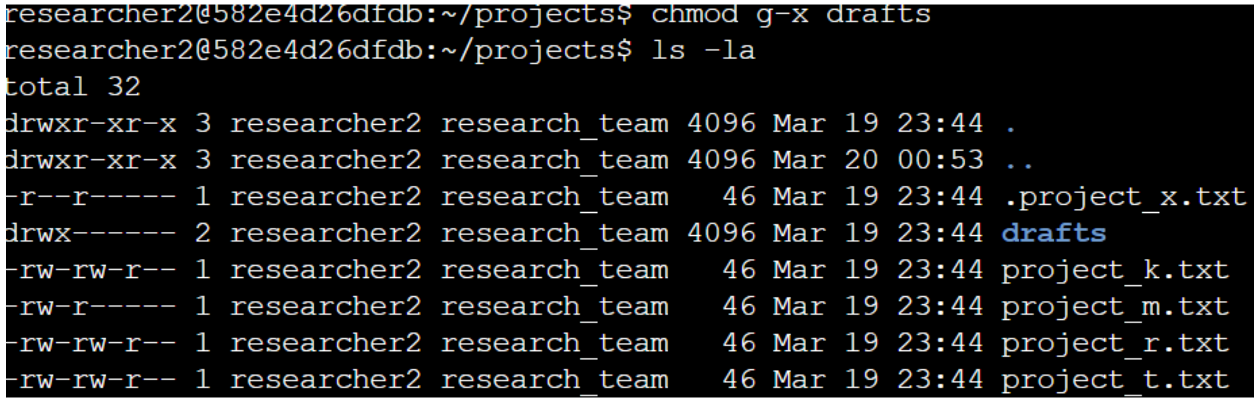


```
researcher2@582e4d26dfdb:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@582e4d26dfdb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Mar 19 23:44 .
drwxr-xr-x 3 researcher2 research_team 4096 Mar 20 00:53 ..
-r--r----- 1 researcher2 research_team   46 Mar 19 23:44 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Mar 19 23:44 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Mar 19 23:44 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_t.txt
```

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

Change directory permissions

My organization requires that only the `researcher2` user has 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 modify the permissions:



```
researcher2@582e4d26dfdb:~/projects$ chmod g-x drafts
researcher2@582e4d26dfdb:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Mar 19 23:44 .
drwxr-xr-x 3 researcher2 research_team 4096 Mar 20 00:53 ..
-r--r----- 1 researcher2 research_team   46 Mar 19 23:44 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Mar 19 23:44 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Mar 19 23:44 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 19 23:44 project_t.txt
```

The output here shows the permission listing for several files and directories. Line 1 indicates the current directory (`projects`), and line 2 shows the parent directory (`home`). Line 3 displays a regular file named `.project_x.txt`. Line 4 represents the `drafts` directory, which has restricted permissions. As shown, only `researcher2` has execute permissions. Since it was previously determined that the group had execute permissions, I used the `chmod` command to remove them. The `researcher2` user already had execute permissions, so there was no need to add them again.

Summary

I changed multiple permissions to align with the level of authorization my organization wanted for files and directories in the `projects` directory. The first step was using `ls -la` to check the current permissions for the directory, which helped guide my decisions in the following steps. Afterward, I used the `chmod` command several times to modify the permissions on files and directories as needed.