File permissions in Linux

Project description

As part of the research team at our organization our job is to ensure users have the current authorization and appropriate level permissions for certain files and directories.This helps keep the system secure. Some permissions do not reflect the level that should be given. To complete this task, I performed the following tasks:

Check file and directory details

The following screenshot demonstrates the Linux command that I used to list and show the set permissions for a specific directory in the file system.

```
researcher2@85ea0c1da42e:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Apr  1 19:21 .
drwxr-xr-x 3 researcher2 research_team 4096 Apr  1 19:34 ..
-rw--w---- 1 researcher2 research_team   46 Apr  1 19:21 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Apr  1 19:21 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Apr  1 19:21 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Apr  1 19:21 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_t.txt
```

The first line of the screenshot displays the command that I entered and the other lines displays the output. The command `ls` with the combination `-la` displays a detailed list of all contents of the `projects` directory. Also, shows all the hidden files. The output indicates that there is one directory named `drafts`, one hidden file named `.project_x.txt` which we can identify by the dot (`.`) that comes before the name of the file and four other project files. The list also provides a 10-character string which represents the permission set for each file or directory.

Describe the permissions string

The 10-character string in Linux file permissions shows the type of file and who can access it and what they can do with it. Here is what each character means:

> *1st character*: This character shows the file type. If it's a **d**, it's a directory. If it's a hyphen **(-)**, it's a regular file.
> *2nd - 4th characters*: These characters indicate (r) for reading, (w) for writing and (x) for execute. They represent the permissions for the **user**, if there is a (-) hyphen instead of a letter it means that the permission is not granted to the user.
> *5th - 7th characters*: Similar to the user permissions, these characters indicate the same, (r) for reading, (w) for writing, (x) for execute and (-) hyphen to not grant the permission. But they are related with the **group** permission instead of the user.
> *8th - 10th characters:* These characters show the permission for **other**. Meaning everyone who isn't the user or group. Again, you will see **r,w,x or -** for the granted or not granted permission for **other**.

For example, the file permissions for **project_k.txt** are -rw-rw-r--. The first character is a hyphen (**-**) indicating that **project_k.txt** is a file, not a directory. The second, fifth and eighth characters show a **r** which means that the user, group and other have the read permission. The third and sixth characters are a **w** indicating the permission for write only for user and group. In the file **project_k.txt** no one has the permission to execute.

Change file permissions

The organization decided not allow others to have write access to any of their files. Based on the previous file permissions returned I determine that **project_k.txt** needs to have the write permission removed for **other**. As shown in the following screenshot howl used Linux commands to achieve this:

```
researcher2@85ea0c1da42e:~/projects$ chmod o-w project_k.txt
researcher2@85ea0c1da42e:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Apr  1 19:21 .
drwxr-xr-x 3 researcher2 research_team 4096 Apr  1 19:34 ..
-rw--w---- 1 researcher2 research_team   46 Apr  1 19:21 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Apr  1 19:21 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Apr  1 19:21 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_t.txt
```

I entered a first command in the first line and entered a second command as shown in the second line. The other lines show the output for the second command I entered. The command **chmod** is used to make permission changes on files or directories.The first argument indicates

the permissions that need to be changed and the second argument indicates where the change needs to be performed. As asked by the organization, I removed the write permission for **other** on the file **project_k.txt** then I used **ls -l** to show the updates. After, (not shown on the screenshot) I also performed the command **ls -a** to make sure no hidden files or directories had **other** write permission.

Change file permissions on a hidden file

The research team at my organization has archived **.project_x.txt**, which is why it's a hidden file. They do not want this file to have write permissions for anyone, but the user and group should be able to read the file. In the following screenshot demonstrate how I use Linux commands to change the permissions:

```
researcher2@85ea0c1da42e:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@85ea0c1da42e:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Apr  1 19:21 .
drwxr-xr-x 3 researcher2 research_team 4096 Apr  1 19:34 ..
-r--r----- 1 researcher2 research_team   46 Apr  1 19:21 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Apr  1 19:21 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Apr  1 19:21 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_t.txt
```

The first two lines display the commands that I entered, the other lines show the output of the second command. I removed write permissions for the user **u-w**, removed write permissions for the group **g-w** and added read permissions to the group **g+r**. Same as before, I runned the **ls -la** command to show updates.

Change directory permissions

The organization wants only the researcher2 user to have access to the drafts directory and its contents in the projects directory, meaning that no one other than researcher2 should have execute (**x**) permissions. The following screenshot displays how I use a Linux command to modify the permissions accordingly.

```
researcher2@85ea0c1da42e:~/projects$ chmod g-x drafts
researcher2@85ea0c1da42e:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Apr  1 19:21 .
drwxr-xr-x 3 researcher2 research_team 4096 Apr  1 19:34 ..
-r--r----- 1 researcher2 research_team   46 Apr  1 19:21 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Apr  1 19:21 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Apr  1 19:21 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Apr  1 19:21 project_t.txt
researcher2@85ea0c1da42e:~/projects$
```

The first two lines display the commands that I entered, the other lines show the output of the second command. Previously, it was determined that the group had execute permissions, so I used the **chmod** command to remove them. Researcher2 already had execute permissions, no modification was applied. The second command was used the same as before, to show updates.

Summary

As requested by my organization, I have applied multiple changes to match the level of authorization needed for the files and directories in the projects directory. Before making a decision, I needed to check the permissions using **ls -la** to modify the permissions wanted. Then, I used the command **chmod** to apply the necessary permissions on the files and directories.