File permissions in Linux

Project description

Within our organization, the team dedicated to research needs to investigate and update their existing file permissions. To keep their system safe and secure I will need to ensure that permissions match authorization and modify permissions to authorize users and remove unauthorized access. To complete this, I performed 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@382ca232d7ef:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec 24 02:44 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec 24 03:51 ..
-rw--w---- 1 researcher2 research_team   46 Dec 24 02:44 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Dec 24 02:44 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Dec 24 02:44 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec 24 02:44 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_t.txt
```

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

The 10-character string can be deciphered to determine who is authorized to access the file and their specific permissions. Below are the characters and their corresponding representations are outlined as follows:

- **1st character**: This character is either a `d` or hyphen (`-`) and indicates 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 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.
- **5th-7th characters:** These 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.
- **8th-10th characters:** These characters indicate the read (`r`), write (`w`), and execute (`x`) permissions for 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 `project_r.txt` are `-rw-rw-r--`. Since the first character is a hyphen (-), this indicates that the `project_r.txt` is a regular 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 written permissions. No one has execute permissions for `project_r.txt`.

Change file permissions

The organization concluded that others should not possess write access to any of their files. To adhere to this directive, I consulted the file permissions I had previously retrieved. Upon examination, I found that the write access for others must be revoked specifically for the file project_k.txt.

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

```
researcher2@382ca232d7ef:~/projects$ chmod o-w project_k.txt
researcher2@382ca232d7ef:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Dec 24 02:44 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec 24 02:44 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_t.txt
```

The initial two lines in the screenshot showcase the commands I inputted, while the subsequent lines exhibit the output of the second command. The `chmod` command is responsible for altering permissions on files and directories. The first argument signifies the permissions targeted for modification, and the second parameter designates the file or directory affected. In this instance, I eliminated write permissions from others for the file `project_k.txt`. Subsequently, I employed `ls -la` to inspect the changes I implemented.

Change file permissions on a hidden file

The research team in my organization has recently archived `project_x.txt`. The aim is to prevent any write access to this project, while maintaining read access for both the user and the group.

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

```
researcher2@382ca232d7ef:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@382ca232d7ef:~/projects$ ls -l
total 20
drwx--x--- 2 researcher2 research_team 4096 Dec 24 02:44 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_k.txt
-rw------- 1 researcher2 research_team   46 Dec 24 02:44 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_t.txt
```

+

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@382ca232d7ef:~/projects$ chmod g-x drafts
researcher2@382ca232d7ef:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec 24 02:44 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec 24 03:51 ..
-r--r----- 1 researcher2 research_team   46 Dec 24 02:44 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Dec 24 02:44 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_k.txt
-rw------- 1 researcher2 research_team   46 Dec 24 02:44 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 24 02:44 project_t.txt
researcher2@382ca232d7ef:~/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 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

I changed multiple permissions to match the level of authorization my organization wanted for files and directories in the `projects` directory. The first step in this was using `ls -la` to check the permissions for the directory. This informed my decisions in the following steps. I then used the `chmod` command multiple times to change the permissions on files and directories. I now have practical experience using basic Linux Bash shell commands. I've successfully examined file and directory permissions, changed permissions on files, and adapted permissions on directories.