File permissions in Linux

Project description

The research team at my large organization needs to ensure that all users are authorized with the proper permissions within the `projects` directory. This will help keep our system safe. I have been tasked with examining existing permissions on the file system, and modify as necessary. To complete this task, 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@7b9c91bbd432:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul  8 16:36 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul  8 17:11 ..
-rw--w---- 1 researcher2 research_team   46 Jul  8 16:36 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul  8 16:36 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Jul  8 16:36 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul  8 16:36 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  8 16:36 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  8 16:36 project_t.txt
researcher2@7b9c91bbd432:~/projects$
```

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, including hidden files. 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 four 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 deconstructed to determine who is authorized to access the file and their specific permissions. The characters and what they represent are 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 for `drafts` directory are `drwx--x---`. Since the first character is `d`, this indicates that `drafts` is a directory, not a file. The next three characters are `r` ,`w` , `x` , which indicates that the user has read, write, and execute permissions. The seventh character is `x` , which indicates that the research_team group also has execute permissions. No one else has read/write permissions aside from user, and other has no permissions for `drafts.`

.

Change file permissions

The organization determined that other shouldn't have write access to any of their files. To comply with this, I referred to the file permissions that I previously returned. I determined `project_k.txt` must have the write access removed for other.

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

```
researcher2@05232a13a07c:~/projects$ chmod o-w project_k.txt
researcher2@05232a13a07c:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul  8 20:20 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul  8 20:58 ..
-rw--w---- 1 researcher2 research_team   46 Jul  8 20:20 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul  8 20:20 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jul  8 20:20 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Jul  8 20:20 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  8 20:20 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul  8 20:20 project_t.txt
researcher2@05232a13a07c:~/projects$
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. The `chmod` command changes the permissions on files and directories. The first argument indicates what permissions should be changed, and the

second argument specifies the file or directory. In this example, I removed write permissions from other for the `project_k.txt` file. After this, I used `ls -la` to review the updates I made.

Change file permissions on a hidden file

The research team at my organization recently archived `project_x.txt`. They do not want anyone to have write access to this project, but the user and group should have read access.

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

```
researcher2@e781b86c9063:~/projects$ chmod u=r,g=r .project_x.txt
researcher2@e781b86c9063:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 03:55 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 04:35 ..
-r--r----- 1 researcher2 research_team   46 Jul 10 03:55 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Jul 10 03:55 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 03:55 project_k.txt
-rw------- 1 researcher2 research_team   46 Jul 10 03:55 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 03:55 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 03:55 project_t.txt
researcher2@e781b86c9063:~/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 know `.project_x.txt` is a hidden file because it starts with a period (`.`). In this example, I set permissions for both the user and group to read only with `u=r,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@e781b86c9063:~/projects$ chmod g-x drafts
researcher2@e781b86c9063:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 03:55 .
drwxr-xr-x 3 researcher2 research_team 4096 Jul 10 04:35 ..
-r--r----- 1 researcher2 research_team   46 Jul 10 03:55 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Jul 10 03:55 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 03:55 project_k.txt
-rw------- 1 researcher2 research_team   46 Jul 10 03:55 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 03:55 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Jul 10 03:55 project_t.txt
researcher2@e781b86c9063:~/projects$
```

The output here displays the permission listing for several files and directories. Line 1 indicates the current directory (projects), and line 2 indicates the parent directory (home). Line 3 indicates a regular file titled `.project_x.txt`. Line 4 is the directory (drafts) with restricted permissions. Here you can see that only researcher2 has execute permissions. It was 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.