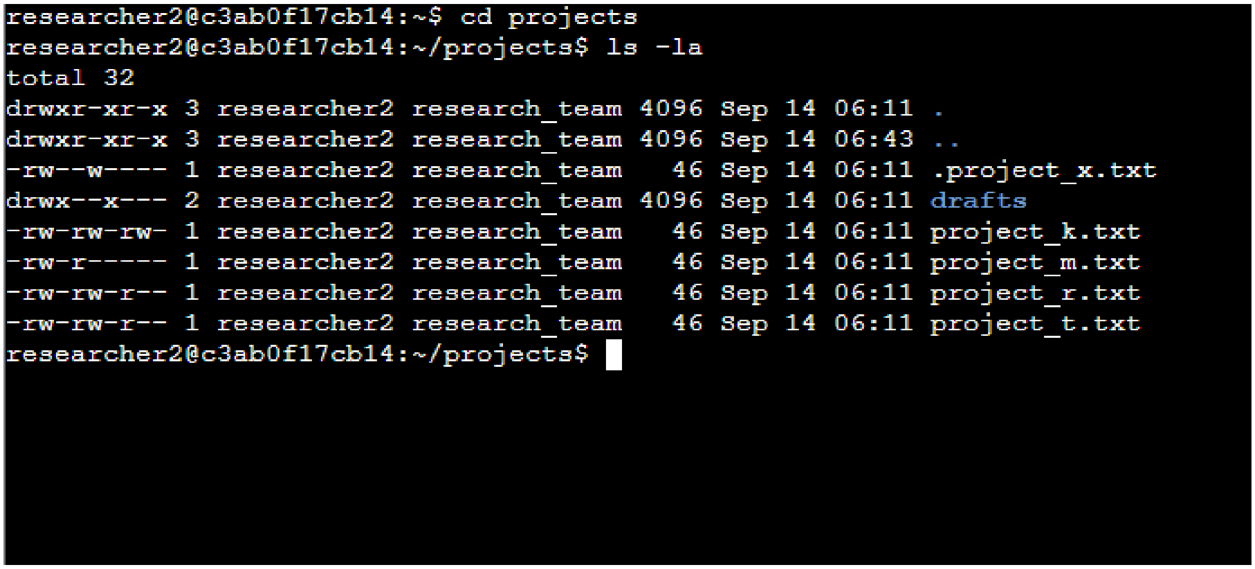
File permissions in Linux

Project description

My company's research team needs to change the file permissions for a few specific files and folders in the `projects` directory. The level of authorization that should be granted is not yet reflected in the permissions. Their system will remain secure if these permissions are checked and updated. I did the following things to finish this task:

Check file and directory details

The code that follows shows how I used Linux commands to find out the current permissions that are in place for a particular directory in the file system.



```
researcher2@c3ab0f17cb14:~$ cd projects
researcher2@c3ab0f17cb14:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 14 06:11 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 14 06:43 ..
-rw--w---- 1 researcher2 research_team   46 Sep 14 06:11 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 14 06:11 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Sep 14 06:11 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 14 06:11 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:11 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:11 project_t.txt
researcher2@c3ab0f17cb14:~/projects$ 
```

The command I typed is shown on the first line of the screenshot, and the output is shown on the following lines. The code contains a list of every item in the `projects` directory. I displayed a thorough listing of the file contents that included hidden files using the `ls` command with the `-la` option. According to the results of my command, there is one directory called `drafts`, one hidden file with the name `.project_x.txt`, and five other project files. The permissions assigned to each file or directory are represented by the 10-character string in the first column.

Describe the permissions string

You can decipher the 10 characters to find out who has permission to access the file and what those permissions are. The following are the characters and what they stand for:

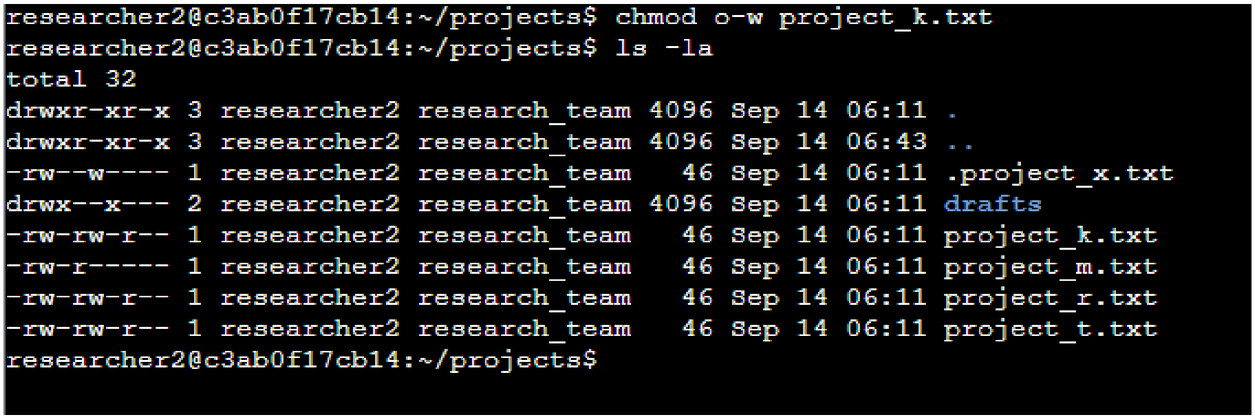
- **1st character**: The file type is indicated by this character, which is either a `d` or a hyphen (`-`). A directory if it starts with a `d`. If there is a hyphen (`-`), the file is a regular one.
- **2nd-4th characters**: These letters stand for the user's permissions to read (`r`), write (`w`), and execute (`x`). When one of these characters is substituted with a hyphen (`-`), it means that the user does not have access to that feature.
- **5th-7th characters:** The read (`r`), write (`w`), and execute (`x`) permissions for the group are denoted by these characters. If one of these characters is a hyphen (`-`) instead, it means that the group does not have this permission.
- **8th-10th characters:** The read (`r`), write (`w`), and execute (`x`) permissions for other are denoted by these characters. All users on the system except the user and the group are included in this owner type. A hyphen (`-`) in place of one of these characters denotes that this permission is not given for the other.

For instance, `project_t.txt`'s file permissions are `-rw-rw-r--`. The fact that `project_t.txt`'s initial character is a hyphen (`-`) denotes that it is a file, not a directory. Indicating that user, group, and other all have read permissions, the second, fifth, and eighth characters are all `r`’s. Only the user and group have write rights, as shown by the third and sixth characters, which are `w`. No one has the ability to execute `project_t.txt`.

Change file permissions

The company decided that no one else should be able to write to any of their files. I referenced back to the file permissions that I had previously returned in order to comply with this. I came to the conclusion that other users should not have write access to `project_k.txt`.

The code below shows how I accomplished this using Linux commands:



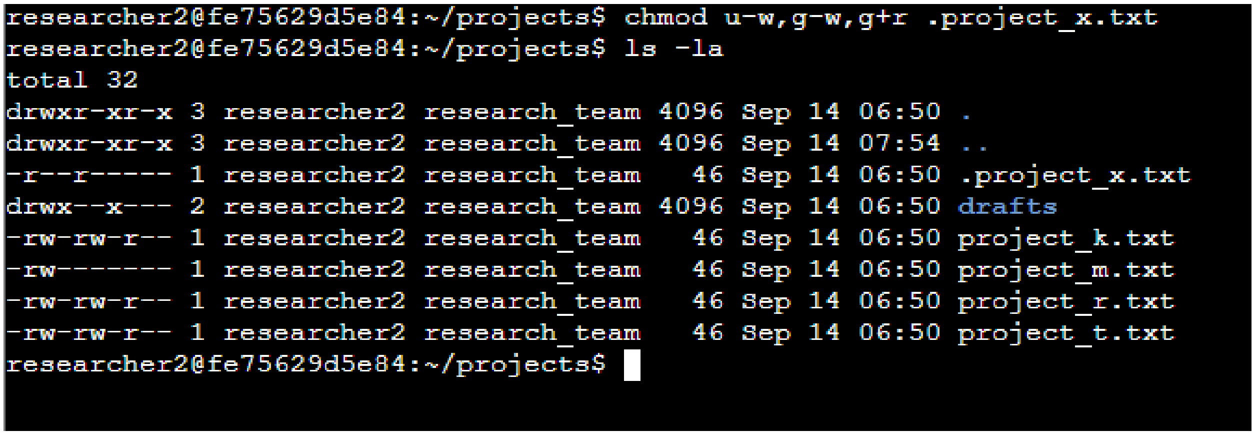
```
researcher2@c3ab0f17cb14:~/projects$ chmod o-w project_k.txt
researcher2@c3ab0f17cb14:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 14 06:11 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 14 06:43 ..
-rw--w---- 1 researcher2 research_team   46 Sep 14 06:11 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 14 06:11 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:11 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 14 06:11 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:11 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:11 project_t.txt
researcher2@c3ab0f17cb14:~/projects$
```

The screenshot's first two lines show the commands I typed in, and the following lines show the second command's results. The `chmod` command modifies a file's or directory's permissions. The second argument provides the file or directory, while the first argument specifies which permissions should be altered. In this case, I disabled other's ability to write to the `project_k.txt` file. I then ran `ls -la` to look over the adjustments I had made.

Change file permissions on a hidden file

`project_x.txt` was recently archived by my organization's research team. The user and group should only be able to have read acces to this project; they do not want anyone to have write access.

The code below shows how I changed the permissions using Linux commands:



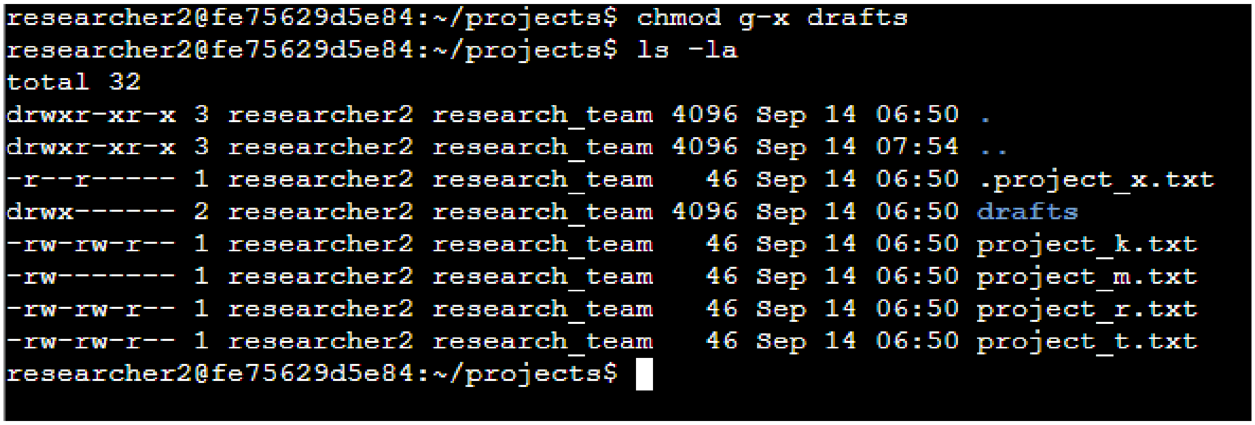
```
researcher2@fe75629d5e84:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@fe75629d5e84:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 14 06:50 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 14 07:54 ..
-r--r----- 1 researcher2 research_team   46 Sep 14 06:50 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 14 06:50 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:50 project_k.txt
-rw------- 1 researcher2 research_team   46 Sep 14 06:50 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:50 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:50 project_t.txt
researcher2@fe75629d5e84:~/projects$
```

The screenshot's first two lines show the commands I typed in, and the following lines show the second command's results. I know .Due to its period (`.`) start , `.project_x.txt` is a hidden file. In this illustration, I gave the group read capabilities while removing write permissions from the user and group. I used `u-w` to take the user's write permissions away. I then added read permissions to the group with `g+r` and deleted write permissions with `g-w`.

Change directory permissions

Only the `researcher2` user should have access to the `drafts` directory and its contents, according to my organization. This indicates that only `researcher2` should be granted execute permissions.

The code below shows how I changed the permissions using Linux commands:



```
researcher2@fe75629d5e84:~/projects$ chmod g-x drafts
researcher2@fe75629d5e84:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 14 06:50 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 14 07:54 ..
-r--r----- 1 researcher2 research_team   46 Sep 14 06:50 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Sep 14 06:50 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:50 project_k.txt
-rw------- 1 researcher2 research_team   46 Sep 14 06:50 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:50 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 14 06:50 project_t.txt
researcher2@fe75629d5e84:~/projects$ 
```

The screenshot's first two lines show the commands I typed in, and the following lines show the second command's results. I used the `chmod` command to remove the execute rights after previously determining that the group possessed them. It was not necessary to add execute rights because the `researcher2` user already had them.

Summary

In order to provide files and directories in the `projects` directory the degree of authorization that my organization desired, I altered a number of permissions. Using `ls -la` to examine the directory's permissions was the initial step in this process. My choices in the subsequent steps were influenced by this. Then, I repeatedly used the `chmod` command to alter the permissions of files and directories.