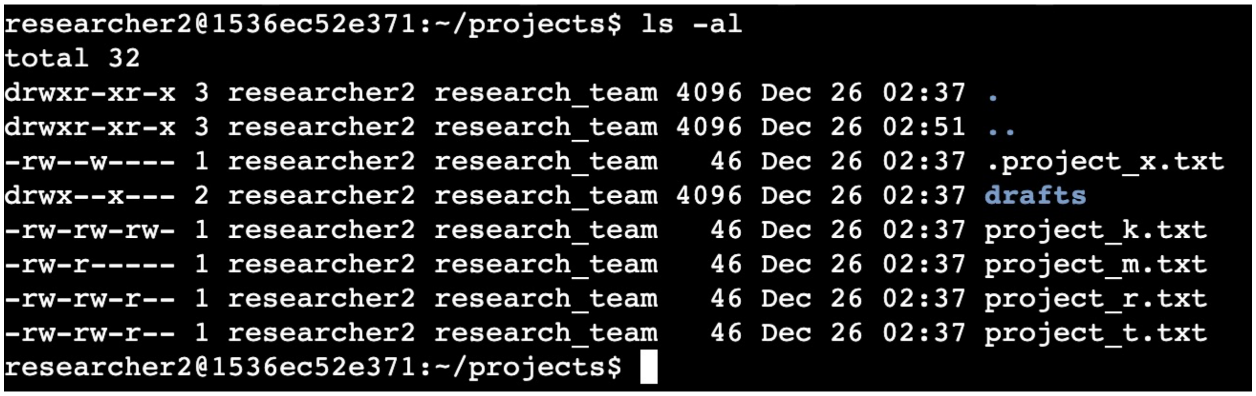
File permissions in Linux

Project description

Linux/bash commands are direct instructions that help structure and organize the OS. In this particular example, Linux commands are used to manage files, specifically, file permissions so that every file can only be read from, written to, and executed by the correct people.

Check file and directory details

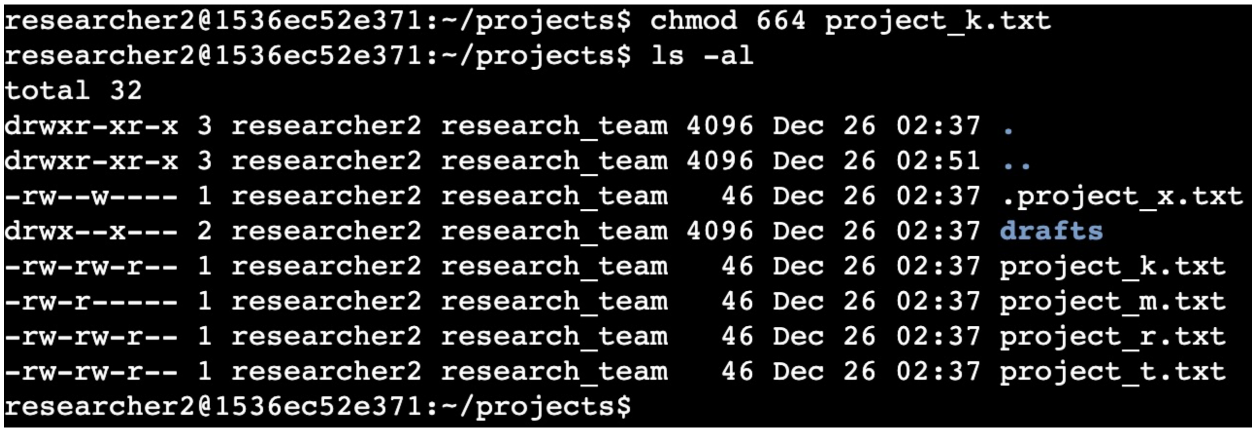


```
researcher2@1536ec52e371:~/projects$ ls -al
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec 26 02:37 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec 26 02:51 ..
-rw--w---- 1 researcher2 research_team   46 Dec 26 02:37 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Dec 26 02:37 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Dec 26 02:37 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec 26 02:37 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_t.txt
researcher2@1536ec52e371:~/projects$ 
```

Describe the permissions string

Looking at “drafts” in the picture above, we can see that the first character starting from the left is “d”. This means that this item is a directory! Next, observing the following 9 characters in blocks of three, we get: “rwx”, “--x”, “---”. Each three characters corresponds to the permissions that different users would have when trying to read, write, execute the item in question. The first set represents the permissions of the main user, in this case, “researcher2”. This user has the permission to read, write, and execute this directory. The next set represents the permissions of the group, “research_team”. This user has the permission to only execute. Finally, “other” users don’t have any permissions with this folder, thus leading to their permission characters to be “---”

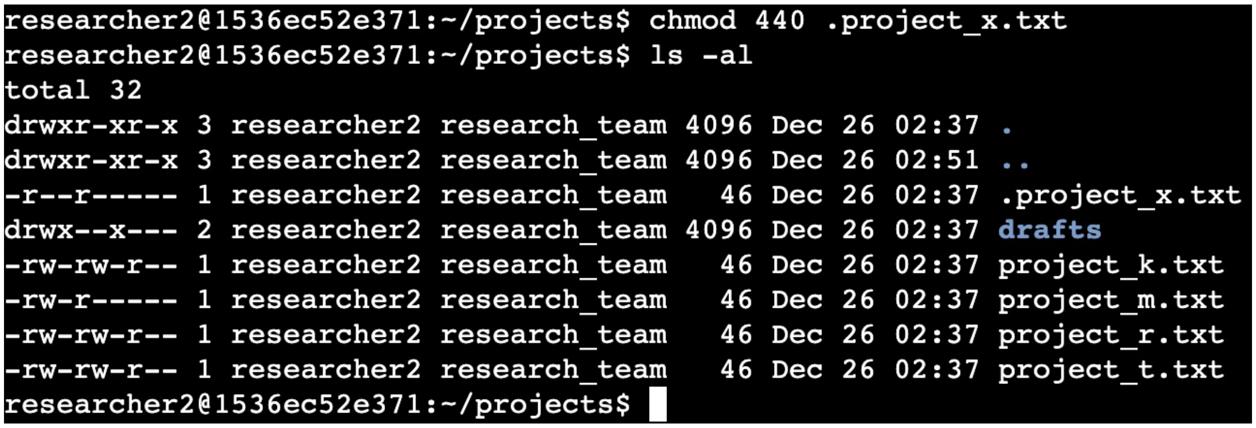
Change file permissions



```
researcher2@1536ec52e371:~/projects$ chmod 664 project_k.txt
researcher2@1536ec52e371:~/projects$ ls -al
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec 26 02:37 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec 26 02:51 ..
-rw--w---- 1 researcher2 research_team   46 Dec 26 02:37 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Dec 26 02:37 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec 26 02:37 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_t.txt
researcher2@1536ec52e371:~/projects$
```

Observing the permissions in the first image, we can see that “other” had write permissions for the file “project_k.txt”. Therefore, using the chmod command, I changed the permissions so that “other” only has read permissions for that file. I did so using the binary code approach to the command “chmod” where read is the highest bit (4 in base_10), write is the second highest bit (2 in base_10) and execute is the lowest bit (1 in base_10). Thus to maintain read and write permissions for the group and main user, I used the code 664, modifying “other” to “read-only”

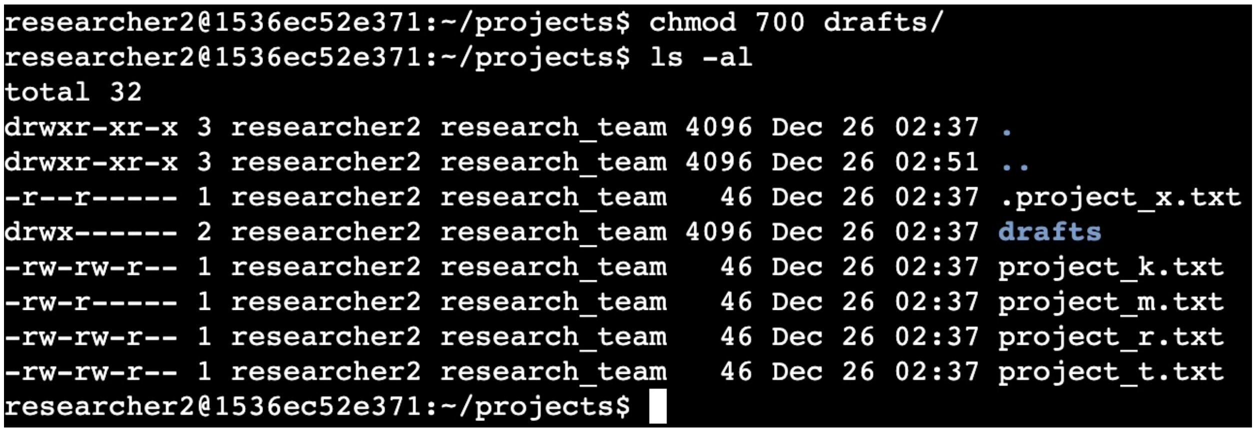
Change file permissions on a hidden file



```
researcher2@1536ec52e371:~/projects$ chmod 440 .project_x.txt
researcher2@1536ec52e371:~/projects$ ls -al
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec 26 02:37 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec 26 02:51 ..
-r--r----- 1 researcher2 research_team   46 Dec 26 02:37 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Dec 26 02:37 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec 26 02:37 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_t.txt
researcher2@1536ec52e371:~/projects$
```

Just like the previous section, “.project_x.txt” had incorrect permissions as both the user and the group were able to write to the file. Instead, we wanted both user and group to be able to read from the file, but nothing else. Thus, I used the same command “chmod” and the binary code for “read-only” (4) for both sets, leaving “other” with no permissions.

Change directory permissions



```
researcher2@1536ec52e371:~/projects$ chmod 700 drafts/
researcher2@1536ec52e371:~/projects$ ls -al
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Dec 26 02:37 .
drwxr-xr-x 3 researcher2 research_team 4096 Dec 26 02:51 ..
-r--r----- 1 researcher2 research_team   46 Dec 26 02:37 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Dec 26 02:37 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Dec 26 02:37 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Dec 26 02:37 project_t.txt
researcher2@1536ec52e371:~/projects$ 
```

Now, “drafts” had the incorrect permissions as it allowed the group to execute it. To modify the permissions so that only the user has full permissions over the directory, “chmod 700” was used to correct the permissions.

Summary

Through this lab, we went through how file permissions work in the Linux/bash environment and the commands needed to properly navigate and administrate permissions. Different sets of characters were were used to represent the permissions each “user” would have and whether the item is a directory or a file. Then, I learned how to use commands to change the permissions of items, using binary code to easily specify permissions to spec.