Joshua Dunne

2239-CS-497-01-1763-1

Assignment 2

Abstract

We will be exploring the use of chmod and file access control lists to control access to files and directories. We will be demonstrating our knowledge of navigating the file heirarchy, creating users and groups, and setting permissions for files and directories. Too we will demonstrate basic redirection of output to files, and the use of File Access Control Lists to set permissions for files and directories.

Assignment

Using your Ubuntu virtual machine, complete the following tasks using only the bash shell. All activities must be done using the Linux command prompt in the bash shell.

Summary of results

- Create a new group called, read_sec_files.

```
jdizzle@jdizzle-vbuntu:~/CS497$ sudo groupadd read_sec_files
```

- Create another new group called, write_sec_files.

```
jdizzle@jdizzle-vbuntu:~/CS497$ sudo groupadd write_sec_files
```

- Create 2 new users with names of your choosing.

```
jdizzle@jdizzle-vbuntu:~/CS497$ sudo adduser --no-create-home --disabled-password cs-497-assignment-2-user-1
jdizzle@jdizzle-vbuntu:~/CS497$ sudo adduser --no-create-home --disabled-password cs-497-assignment-2-user-2
```

- Add one of your new users to read_sec_files. Add the other user to write_sec_files.

```
jdizzle@jdizzle-vbuntu:~/CS497$ sudo usermod -G read_sec_files cs-497-assignment-2-user-1
jdizzle@jdizzle-vbuntu:~/CS497$ sudo usermod -G write_sec_files cs-497-assignment-2-user-2
```

- Create a new folder in /var/tmp called CS497.

```
jdizzle@jdizzle-vbuntu:~/CS497$ mkdir /var/tmp/CS497
```

- Check the permissions for this folder. Show this.

```
jdizzle@jdizzle-vbuntu:~/CS497$ ls -ld /var/tmp/CS497/
drwxrwxr-x 2 jdizzle jdizzle 4096 Sep  7 22:30 /var/tmp/CS497/
```

- Create a new file called myFile.txt in the new CS497 folder.

```
jdizzle@jdizzle-vbuntu:~/CS497$ touch /var/tmp/CS497/myFile.txt
```

- Redirect the output of the following commands to this new file:

```

```

```
date; uname -a; whoami; ip address
```

```
jdizzle@jdizzle-vbuntu:~/CS497$ echo -e "$(date)" '\n' "$(uname -a)" '\n' "$(whoami)" '\n' "$(ip address)" > /var/tmp/CS497/myFile.t
```

- Move the CS497 folder to your user's Desktop folder.

```
jdizzle@jdizzle-vbuntu:~/CS497$ mv /var/tmp/CS497/ ~/Desktop/
jdizzle@jdizzle-vbuntu:~/CS497$ ls -R ~/Desktop/
/home/jdizzle/Desktop/:
CS497

/home/jdizzle/Desktop/CS497:
myFile.txt
```

- The /var/tmp/cs497 folder that you created earlier should no longer exist. Check. Show this.

```
jdizzle@jdizzle-vbuntu:~/CS497$ ls /var/tmp/CS*
ls: cannot access '/var/tmp/CS*': No such file or directory
```

- Check the permissions for the CS497 folder and myFile.txt file now located in the Desktop folder.

```
jdizzle@jdizzle-vbuntu:~/CS497$ ls -ld ~/Desktop/CS497/
drwxrwxr-x 2 jdizzle jdizzle 4096 Sep  7 22:39 /home/jdizzle/Desktop/CS497/
jdizzle@jdizzle-vbuntu:~/CS497$ ls -l ~/Desktop/CS497/myFile.txt
-rw-rw-r-- 1 jdizzle jdizzle 886 Sep  7 22:35 /home/jdizzle/Desktop/CS497/myFile.txt
```

- Your user account is the owner of this new folder. Show this.

```
jdizzle@jdizzle-vbuntu:~/CS497$ chown jdizzle ~/Desktop/CS497/
```

- Users in the read_sec_files group may read all contents of the CS497 folder. Show this

```
jdizzle@jdizzle-vbuntu:~/Desktop$ setfacl -R -m g:read_sec_files:rx CS497/

jdizzle@jdizzle-vbuntu:~/Desktop$ getfacl -R CS497/
# file: CS497/
# owner: jdizzle
# group: jdizzle

user::rwx
group::rwx
group:read_sec_files:r-x
mask::rwx
other::r-x

# file: CS497//myFile.txt
# owner: jdizzle
# group: jdizzle

user::rw-
group::rw-
group:read_sec_files:r-x
mask::rwx
other::r--
```

- Users in the write_sec_files group may read and write all contents of the CS497 folder.

```
jdizzle@jdizzle-vbuntu:~/Desktop$ setfacl -R -m g:write_sec_files:rwx CS497/

jdizzle@jdizzle-vbuntu:~/Desktop$ getfacl -R CS497/
# file: CS497/
# owner: jdizzle
# group: jdizzle

user::rwx
group::rwx
group:read_sec_files:r-x
group:write_sec_files:rwx
mask::rwx
other::r-x

# file: CS497//myFile.txt
# owner: jdizzle
# group: jdizzle

user::rw-
group::rw-
group:read_sec_files:r-x
group:write_sec_files:rwx
mask::rwx
other::r--
```

- Users not in either group have no access to the files in this folder

```
jdizzle@jdizzle-vbuntu:~/Desktop$ getfacl -R CS497/
# file: CS497/
# owner: jdizzle
# group: jdizzle

user::rwx
group::rwx
group:read_sec_files:r-x
group:write_sec_files:rwx
mask::rwx
other::---

# file: CS497//myFile.txt
# owner: jdizzle
# group: jdizzle

user::rw-
group::rw-
group:read_sec_files:r-x
group:write_sec_files:rwx
mask::rwx
other::---
```

- Create a third user account. Using su, switch to this new user and attempt to access the CS497 folder contents.

```
jdizzle@jdizzle-vbuntu:~/Desktop$ sudo adduser --no-create-home cs-497-assignment-2-user-3
jdizzle@jdizzle-vbuntu:~/Desktop$ su cs-497-assignment-2-user-3
Password:
cs-497-assignment-2-user-3@jdizzle-vbuntu:/home/jdizzle/Desktop$ cd CS497/
bash: cd: CS497/: Permission denied
cs-497-assignment-2-user-3@jdizzle-vbuntu:/home/jdizzle/Desktop$ touch CS497/some_stuff
touch: cannot touch 'CS497/some_stuff': Permission denied
```

- Last using redirection, add the following to your myFile.txt file

```

```

```
Use wc to display the word count info for your file.
Use ls -la to display the permission info for the contents of the CS497 directory.
```

```
jdizzle@jdizzle-vbuntu:~/Desktop$ wc CS497/myFile.txt >> CS497/myFile.txt
jdizzle@jdizzle-vbuntu:~/Desktop$ ls -la CS497/ >> CS497/myFile.txt
```

Conclusion

- chmod
  - What does chmod 644 filename do?
    - chmod 644 changes the files specified to the command to set of permissions specified in octal notation. The effect is different for files and directories. We need both read and execute to traverse a directory.
  - What does the 644 represent?
    - The 644 represents the permissions that are being set for the file or directory. The first number is the owner permissions, the second is the group permissions, and the third is the other permissions. We supply numbers for each in this order. The numbers represent the permissions in octal notation. We add numbers together to form each. 4 specifies read permission. 2 Specifies write permissions. 1 specifies execute permissions. In this case, as 6 = 2 + 4, the owner has read and write permissions. As 4 = 4, the group has read permissions. As 4 = 4, the other has read permissions.
  - What would other numbers do?
    - Lets take another example. Say 755. The first number is the owner permissions, the second is the group permissions, and the third is the other permissions. We supply numbers for each in this order. The numbers represent the permissions in octal notation. We add numbers together to form each. 7 specifies read, write, and execute permissions. 5 specifies read and execute permissions. 1 specifies execute permissions. In this case, as 7 = 4 + 2 + 1, the owner has read, write, and execute permissions. As 5 = 4 + 1, the group has read and execute permissions. As 5 = 4 + 1, the other has read and execute permissions.
    - We can also specify a fourth term. 4 is setuid. 2 is setgid. 1 is sticky bit. 0 is none. For example 4755 would set the setuid bit for the owner, and read, write, and execute permissions for the owner, read and execute permissions for the group, and read and execute permissions for the other.
  - How can this be used to make files only readable by one user?
    - Supposing the user is the owner of the files/directories. We can specify permissions such as 700. This retains all permissions for the owner, and extends no permissions to anyone in the files/directories group or others. Only the user would be able to read the files or traverse the directories.
  - How can this be used to make files executable?
    - By adding the one as explained above. We can make the file executable. We need interpreted files to be both readable and executable. We need compiled files to be only executable. If the file is given permissions such as 550, it will be both readable an executable by both users and the group, but not readable or executable by others.
  - Describe the difference between owner and group permissions
    - The owner is a single user, while the group can be a collection of users. If we change the permissions for the user, it will only apply to how they can access it. If we change the permissions for the group, it will apply to how all users in the group can access it.
  - Clearly describe file access control lists. What is the problem solved by FACL
    - FACLs are a more modern approach than was originally usable in UNIX. With them, we can solve more complex permissions issues, such as allowing two different groups to have access to files/directories, as shown above. We can also allow users to have access to files/directories that they are not the owner of. This is useful for collaboration, and solves the necessity of having to change the owner of the file/directory to allow access to other users.