2023 - 2024

Computer Systems
1st Term Assessable Activity

1º DAM

JOSE LUIS RUBIO FERNANDEZ

EXERCISE 1

Section a

Going to home directory:

```
$ cd ~
```

We are going to create the directory and files structure in /tmp directory for each hierarchical level using one command for level:

Step 1: Create My Files:

```
$ mkdir ../../tmp/"My Files"
```

Step 2: Create Cook Recipes, Comics, Movies:

```
$ mkdir ../../tmp/"My Files"/{"Cook Recipes",Comics,Movies}
```

Step 3: Create Salty, Sweet, Superheroes, Comedy, Man of Steel.txt, Wonder Woman.txt:

```
$ mkdir ../../tmp/"My Files"/"Cook Recipes"/{Salty,Sweet}
../../tmp/"My Files"/Movies/{Superheroes,Comedy} && touch
../../tmp/"My Files"/Comics/{"Man of Steel.txt","Wonder Woman.txt"}
```

Step 4: Create Doughnut.jpg:

```
$ touch ../../tmp/"My Files"/"Cook Recipes"/Sweet/Doughnut.jpg
```

Section b

b.1:

```
$ echo "To make macaroni and cheese, you must first buy macaroni and
cheese" > /tmp/My\ Files/Cook\ Recipes/Salty/"Mac and cheese.txt"
```

b.2:

```
$ rm -r /tmp/My\ Files/Cook\ Recipes/Sweet/
```

b.3:

```
$ cp /tmp/My\ Files/Comics/Man\ of\ Steel.txt /tmp/My\
Files/Movies/Superheroes/"Man of Steel_copy.txt"
```

```
$ echo "Up up and away!" > /tmp/My\ Files/Movies/Superheroes/Man\ of\
Steel_copy.txt
```

Section c

In this section, I am going to use absolute paths.

c.1:

```
$ ln /tmp/My\ Files/Comics/Man\ of\ Steel.txt /tmp/My\
Files/Comics/Man\ of\ Steel_hard.txt
```

```
$ ln -s /tmp/My\ Files/Comics/Wonder\ Woman.txt /tmp/My\
Files/Movies/Superheroes/"Wonder Woman_soft.txt"
```

c.2:

I modify the content of Man of Steel_copy.txt adding “Hello hello and bye bye”:

```
$ echo "Hello hello and bye bye" >> /tmp/My\
Files/Movies/Superheroes/Man\ of\ Steel_copy.txt
```

Are the files Man of Steel.txt and Man of Steel_hard.txt modified? Why?

In the provided context, 'Man of Steel_copy.txt' doesn't modify 'Man of Steel.txt' or 'Man of Steel_hard.txt' because it functions as an independent copy. When creating a copy, a new file is generated with its distinct set of data, pointing to a different inode and data block compared to the original file. Consequently, any modifications made to 'Man of Steel_copy.txt' won't impact the other mentioned files.

c.3:

If you delete the file Man of Steel.txt, what will happen to Man of Steel_hard.txt and Man of Steel_copy.txt?

If you delete the file "Man of Steel.txt," it will only affect that specific file, not "Man of Steel_hard.txt" or "Man of Steel_copy.txt." Each file is an independent entity in the file system, and deleting one file does not automatically affect other files, even if they were hard linked or copied.

c.4:

I delete the Comics directory:

```
$ rm -r /tmp/My\ Files/Comics/
```

What happens to the file Wonder Woman_soft.txt?

By deleting the Comics directory, we have deleted Wonder Woman.txt. So, the symbolic link "Wonder Woman_soft.txt" will no longer work correctly. The symbolic link is a reference to the original file, and if that original file is deleted, the link becomes invalid or "broken." Attempting to access or use "Wonder Woman_soft.txt" after the deletion of "Wonder Woman.txt" may result in errors or indicate that the file it points to no longer exists.

EXERCISE 2

Section a

The first thing we must do is create all de users using “useradd” command:

```
$ sudo useradd -m -s /bin/sh gru
$ sudo useradd -m -s /bin/sh kevin
$ sudo useradd -m -s /bin/sh stuart
$ sudo useradd -m -s /bin/sh nefario
$ sudo useradd -m -s /bin/sh agnes
$ sudo useradd -m -s /bin/sh supermegavillain
```

The command “useradd” creates a user in a non-interactive way.

“-m” option creates the home directory for the user.

“-s /bin/sh” sets the user's shell to /bin/sh.

In the "useradd" command, we can assign the password using the -p option, but I preferred to use the "passwd" command since it is safer to change the password that way.

I’m going to assign the password to each user with the command “passwd”:

```
$ sudo passwd gru
Nueva contraseña: gru_password
Vuelva a escribir la nueva contraseña: gru_password
$ sudo passwd kevin
Nueva contraseña: kevin_password
Vuelva a escribir la nueva contraseña: kevin_password
$ sudo passwd stuart
Nueva contraseña: stuart_password
Vuelva a escribir la nueva contraseña: stuart_password
$ sudo passwd nefario
Nueva contraseña: nefario_password
```

```
Vuelva a escribir la nueva contraseña: nefario_password
$ sudo passwd agnes
Nueva contraseña: agnes_password
Vuelva a escribir la nueva contraseña: agnes_password
$ sudo passwd supermegavillain
Nueva contraseña: supermegavillain_password
Vuelva a escribir la nueva contraseña: supermegavillain_password
```

Section b

Now I create all groups with “groupadd” command:

```
$ sudo groupadd masteroftheuniverse
$ sudo groupadd minions
$ sudo groupadd kids
$ sudo groupadd researchanddevelopment
```

The next step is to assign to each user their primary group. We achieve this with “usermod” command:

```
$ sudo usermod -g masteroftheuniverse gru
$ sudo usermod -g kids agnes
$ sudo usermod -g researchanddevelopment nefario
$ sudo usermod -g minions stuart
$ sudo usermod -g minions kevin
$ sudo usermod -g masteroftheuniverse supermegavillain
```

“-g” option sets the primary group for each user.

Now we must assign to each directory the owner and the group, we can do this with “chown user:group directory” command. Other way to do this is with “chown user directory” to change the owner and “chgrp group directory” to change the group.

-R operate on files and directories recursively.

1. To /evilplans gru can access as the owner and supermegavillain with the group masteroftheuniverse assigned to him by error.

```
$ sudo chown -R gru:masteroftheuniverse ./evilplans
```

2. To /operation_birthday agnes can access as the owner and nefario that is in the researchanddevelopment group.

```
$ sudo chown -R agnes:researchanddevelopment ./operation_birthday
```

3. To /science gru can access as the owner and nefario that is in the researchanddevelopment group.

```
$ sudo chown -R gru:researchanddevelopment ./science
```

4. To /science/bananas gru can access as the owner and kevin and stuart in the minions group.

```
$ sudo chown gru:minions ./science/bananas
```

5. To /science/evilserum only can access gru as the owner.

```
$ sudo chown -R gru ./science/evilserum
```

We have to assign the permissions to the directories, I use the “chmod” command and the octal form to assign the permissions:

1. To /evilplans the owner gru and the group masteroftheuniverse (where is supermegavillain) have all the permissions.

```
$ sudo chmod -R 770 ./evilplans
```

2. To /operation_birthday the owner agnes has total access (r,w,x) but the group researchanddevelopment (where is nefario) only can read(r) and execute(x), but not write(w).

```
$ sudo chmod -R 750 ./operation_birthday
```

3. To /science the owner gru only can read(r) and execute(x), but not write(w) and the group researchanddevelopment (where is nefario) can access with all the permissions. Besides, the others groups can access to the directory asigning access permissions (x), so the minions can access to bananas directory inside the science.

```
$ sudo chmod -R 571 ./science
```

4. To /science/bananas the owner gru and the group minions (where are kevin and stuart) have total access.

```
$ sudo chmod -R 770 ./science/bananas
```

5. To /science/evilserum only have total access the owner gru.

```
$ sudo chmod -R 700 ./science/evilserum
```

EXERCISE 3

1. Preparing the System:

a. Attach a New Hard Disk:

- We attach a new SATA hard disk to our virtual machine.

- I use the “lsblk” command to see the new disk attached:

```
$ lsblk

sda      8:0    0    25G  0 disk
```

b. Create a GPT Partition Table and Assign the Whole Disk to a Unique Partition:

We are going to use “fdisk” command to create the partition table and assign the partition.

```
$ sudo fdisk /dev/sda
```

I Type `g` to create a new GPT partition table:

```
Orden (m para obtener ayuda): g

Se ha creado una nueva etiqueta de disco GPT (GUID: D26E51E8-9B4A-D248-B961-9C8FED1EF512).
```

I Type `n` to create a new partition and accept the default values to use the whole disk.:

```
Orden (m para obtener ayuda): n

Número de partición (1-128, valor predeterminado 1): 1

Primer sector (2048-52428766, valor predeterminado 2048):

Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-52428766, valor predeterminado 52428766):

Crea una nueva partición 1 de tipo 'Linux filesystem' y de tamaño 25 GiB.
```

Now I type `w` to write the changes and exit:

```
Orden (m para obtener ayuda): w

Se ha modificado la tabla de particiones.

Llamando a ioctl() para volver a leer la tabla de particiones.
```

```
Se están sincronizando los discos.
```

If we use “lsblk” again, we can see the new partition:

```
$ lsblk

sda      8:0    0    25G  0 disk

└─sda1   8:1    0    25G  0 part
```

c. Make File System with mkfs.ext4:

```
$ sudo mkfs.ext4 /dev/sda1
```

d. Mount the Disk:

I’m going to create a mount point (/mnt/backup_disk) and mount the new disk:

```
$ sudo mkdir /mnt/backup_disk

$ sudo mount /dev/sda1 /mnt/backup_disk
```

e. Modify /etc/fstab:

We need to modify the fstab file and add the new mount point. This is necessary so that the new disk is mounted automatically each time we start the virtual machine.

```
$ sudo sh -c 'echo "/dev/sda1  /mnt/backup_disk   ext4  defaults  0 2"
>> /etc/fstab'
```

‘sh -c’ option is to ensure that we execute the whole command with sudo privileges.

The format that we have to modify fstab is:

<device> <mount_point> <filesystem_type> <mount_options> <dump> <pass>

2. Daily Backup:

We can copy all the data from /user/important_data to /mnt/backup_disk with the next command:

```
$ sudo cp -a /user/important_data /mnt/backup_disk
```

“cp” command to copy /user/important_data to /mnt/backup_disk.

With “-a” option we preserve the attributes: permissions, timestamps, and ownership.

<u>Schedule the Daily Backup with Cron:</u>

Now we have to edit the crontab for the user who will run the backup (for example, root):

```
$ sudo crontab -e
```

I add the following line to run the command daily at 9:00 AM:

```
0 9 * * * cp -a /user/important_data /mnt/backup_disk
```

Save and exit.

Now, the system is prepared with a daily backup process that preserves attributes and mounts the backup disk automatically. The cron job will execute the command to copy the data every day at 9:00 AM.