GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

Assessable Task 1st Term - Solution

Computer Systems 23/24

DAM / DAW

Aarón Martín Bermejo
Francisco Lifante
Álvaro Maceda

Cicles Formatius

License

Nomenclature

Throughout this unit different symbols will be used to distinguish important elements within the content. These symbols are:

ÍNDICE

1. EXERCISE 1

Section a

Using the terminal, go to your home directory and from there create the following directory structure, as well as the files in /tmp. You must use only one command for each hierarchical level and use relative paths.

My Files
| -- Cook Recipes
        | -- Salty
        | -- Sweet
                | -- Doughnut.jpg
| -- Comics
        | -- Man of Steel.txt
        | -- Wonder Woman.txt
| -- Movies
        | -- Superheroes
        | -- Comedy

cd home/user

Level 1: mkdir ../../tmp/"My Files"

Level 2: mkdir ../../tmp/"My Files"/{"Cook Recipes",Comics,Movies}

Level 3, folders: mkdir ../../tmp/"My Files"/"Cook Recipes"/{Salty,Sweet} ../../tmp/"My Files"/Movies/{Superheroes,Comedy}

Level 3, files: touch ../../tmp/"My Files"/"Cook Recipes"/Sweet/Doughnut.jpg ../../tmp/"My Files"/Comics/{"Man of steel.txt"."Wonder Woman.txt"}

Although the statement says only one command per hierarchical level, I have considered as correct the solution with only one command as well as this solution with 2 commands, one for directories and one for files.

To make a single command you should do this:

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

Section b

From your personal folder, using absolute paths, perform the following 2 tasks.

Use a command to create a file called "Mac and cheese.txt" inside the /Cook Recipes/Salty directory and using pipes or redirection, enter "To make macaroni and cheese, you must first buy macaroni and cheese".
Echo “To make macaroni and cheese, you must first buy macaroni and cheese” > /tmp/”My Files”/”Cook Recipes”/Salty/”Mac and cheese.txt”

Delete the Sweet directory.
rm -r /tmp/”My Files”/Sweet

Now copy the file "Man of Steel.txt" into the Movies/Superheroes directory with the name "Man of Steel_copy.txt" and enter the text "Up up and away!".
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

Create a hard link to "Man of Steel.txt" inside the Comics directory and name it "Man of Steel_hard.txt". Create also, a symbolic link to "Wonder Woman.txt" and place it in Movies/Superheroes with the name "Wonder Woman_soft.txt".

Using absolute paths:
ln /tmp/”My Files”/Comics/”Man of Steel.txt” /tmp/”My Files”/Comics/”Man of Steel_hard.txt”

Modify the contents of the file "Man of Steel_copy.txt". Are the files "Man of Steel.txt" and "Man of Steel_hard.txt" modified? Why?

No, because Man of Steel_copy.txt is a copy, not a hard link or a symbolic link.

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

Both files, 'Man of Steel_hard.txt' and 'Man of Steel_copy.txt', will remain unchanged.
The former is a hard link, a dentry that points to the same inode of the linked file, and it will keep pointing to that inode regardless of what happens to the original file. The latter, while being a copy, is a separate file, and no changes to the original will affect it.

Delete the Comics Directory. What happens to the file Wonder Woman_soft.txt?
We will get a message pointing out that the original file no longer exists. The reason behind this is that a soft link does not point to an inode, it just stores the path to another file; if that file is removed, so is its path.

2.EXERCISE 2

Specify the command/s to achieve the next requirements:

Section a

Create the next users in a **non-interactive way**. Each one of them must have a home folder, their shell must be `/bin/sh` and their password must be their user followed by `_password` and the user name is **case-sensitive**:

- `gru`
- `kevin`
- `stuart`
- `nefario`
- `agnes`
- `supermegavillain`

Explain the command/s you use for both creating the user and achieving the requirements (home, shell, password) and why you chose that way. Think it thoroughly.

$ sudo useradd -m -d /home/gru -s /bin/sh -p $(echo 'gru_password' | openssl passwd -1 -stdin) gru

$ sudo useradd -m -d /home/kevin -s /bin/sh -p $(echo 'kevin_password' | openssl passwd -1 -stdin) kevin

$ sudo useradd -m -d /home/stuart -s /bin/sh -p $(echo 'stuart_password' | openssl passwd -1 -stdin) stuart

$ sudo useradd -m -d /home/nefario -s /bin/sh -p $(echo 'nefario_password' | openssl passwd -1 -stdin) nefario

$ sudo useradd -m -d /home/agnes -s /bin/sh -p $(echo 'agnes_password' | openssl passwd -1 -stdin) agnes

$ sudo useradd -m -d /home/supermegavillain -s /bin/sh -p $(echo 'supermegavillain_password' | openssl passwd -1 -stdin) supermegavillain

useradd: non interactive

-m with home

-d where the home should be

-s specify the shell. The shell by default maybe is sh but you cannot rely on that because that can differ on different distributions of linux and/or even in the same versions of the same distri.

-p sets up the password, but without being encrypted which won't work. You have to use openssl to encrypt it. Another choice would be to use chpasswd with echo, but this way is only one command and non interactive.

Section b

Only the next groups can exist, specify the command to create them:

- `masteroftheuniverse`
- `minions`
- `kids`
- `researchanddevelopment`

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

Given the next folder and files structure (which you don't need to create, because it already exists):

```
├── evilplans
│   └── steal_computer_systems_exam.txt
├── operation_birthday
│   └── grus_birthday_plan.txt
└── science
    ├── bananas
    └── evilserum
        └── recipe.txt
```

Accomplish the next requirements:

- `Gru` can access anything anywhere besides from the folder `operation_birthday`, which cannot see anything (not even listing the files).
- `Agnes` can only access to `operation_birthday` and she's the only able to write inside of it.
- `Nefario` can access to `operation_birthday`, but can't delete anything
- The minions `stuart` and `kevin` cannot access anything besides from the folder `bananas`.
- `Nefario` should only access to the science folder and should be the only one able to write in that folder.
- The `supermegavillain`, because of a silly `Gru`'s mistake, can access to the `evilplans`.

Specify the commands to define the permissions on the folders and file structure and the belongings to each of the groups.

Possible solution

The solution for these requirements is not unique as there are many possibilities to choose from. One way is to use groups to put many users inside each one and playing with the owner user and owner group permissions.

```
$ sudo adduser gru masteroftheuniverse
```

```
$ sudo adduser supermegavillain masteroftheuniverse

$ sudo adduser agnes kids

$ sudo adduser nefario kids

$ sudo adduser stuart minions

$ sudo adduser kevin minions

$ sudo adduser gru minions

$ sudo adduser nefario researchanddevelopment

$ sudo adduser gru researchanddevelopment

$ sudo adduser stuart researchanddevelopment

$ sudo adduser kevin researchanddevelopment
```

Gru is in every group except kids and the villain is (by mistake) in masteroftheuniverse.

```
$ sudo chmod -R g+rwx ./exercise
```

We setup permissions for the group so gru has permissions for everything everywhere, ex. Now, for operation birthday where gru is not allowed:

```
$ sudo chown -R agnes:kids ./exercise/operation_birthday

$ sudo chmow -R 750 ./exercise/operation_birthday
```

The owner is agnes and the group kids, so gru is not allowed and nefario is. Setup the permissions to 750 so the user owner has all permissions (agnes) and the group rw (agnes and nefario) so nefario can't delete.

```
$ sudo chown -R nefario:researchanddevelopment ./exercise/science

$ sudo chmod -R 750 ./exercise/science
```

This way, nefario has full access to everything in science, gru has access to science and the minions too so they reach the folder bananas.

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

$ sudo chmod g+rwx ./exercise/science/bananas
```

This way, the minions (stuart and kevin) have full access to everything in bananas, and gru too since it's in the group.

```
$ sudo chown :masteroftheuniverse ./exercise/evilplans
```

And this way the group masters of the universe (gru and the villain) own the evilplans and both have full access.

REMEMBER ACLs were not allowed, we have not studied them.

3. EXERCISE 3

First of all, there are multiple solutions to this exercise. We'll put down one of them or guidelines for it, but don't take it as a template.

To prepare the disk:

- Format the new disk. You can use gparted for instance to do it.
- Create an ext4 filesystem partition on the new disk, so there are no compatibility issues with the original one. You can also use gparted for that.
- Now you have to mount that partition. You can use mount or edit the fstab. As explained in the contents, one example to do so would be:
  - mount -t ext4 /dev/sdb1 /second_disk
  - And with an fstab edition:

```
# <file system>        <mount point>  <type>   <options>
                       <dump>         <pass>
# more stuff
/dev/sdb1              /second_disk   ext4     defaults,errors=remount-ro
                       0              0
```

If you use mount the mounting will be temporary, so if the computer restarts it won't be there. The best way to do it is with the edition of the fstab in this case. These things to prepare the disk would need a user with superuser privileges.

To run the copy, it would need a user with superprivileges to ensure the access to folders of other users. To copy could be done with the next command:

sudo cp -R --preserve /user/important_data /second_disk/backup

The --preserve argument is important, since that option would be the one that preserves the attributes: permissions, timestamps and ownership.

This command could be run each day so the copy is done daily or it could be configured in the crontab so that daily task is scheduled. To do so, the crontab should be edited:

sudo crontab -e

0 9 * * * cp -R --preserve /user/important_data /second_disk/backup

That way the command would be run each day at 9 AM.