Rapport du devoir de TP en Administartion des systemes

Abdelhalim Jean Abderrahmane (C08890) -DAII

Partie 1 : Gestion des utilisateurs

Exercice 1. En utilisant l'éditeur vi., modifier les fichiers nécessaires afin d'ajouter deux groupes : *gtr1* et un groupe correspondant à votre groupe de TP.

* Linux stocke les informations concernant les groupes d'utilisateurs dans le fichier /etc/group

-On ouvre le fichier avec l'editeur vi avec la commande:

```
$ sudo vi /etc/group
```

-Ensuite on tape la touche i pour entrer dans le mode d'insertion

-On utilise le fleche du bas pour arriver au dernier ligne puis on ajoute les lignes:

```
gtr1:x:1005:
mongroupTP:x:1006:
# le GID 1005 car le dernier groupe crée sur mon systeme avait le GID 1004
```

-Ensuite on tape la touche ESC pour quitter le mode d'insertion puis :wq pour enregistrer et quitter l'editeur vi

-On verifie que les groueps ont été crées avec la commande:

```
$ cat etc/group
```

qui montre :

```
gtr1:x:1005:
mongroupTP:x:1006:
jean@jean ~ $
```

Exercice 2. Appliquer la méthode naïve décrite ci-haut pour créer deux comptes pour les membres du binôme. Essayer d'accéder à la machine en fournissant le *login* de l'utilisateur après chaque étape (donc essayer 5 fois) et noter les messages rendus par le système. Préciser dans votre compte-rendu les modifications apportées aux fichiers de configuration.

* la structure /etc/passwd comprend 7 champs:

```
champs #1: le nom d'utilisateur
champs #2: le mot de passe ( x : qui fait reference au fichier /etc/shadow )
champs #3: l'identifiant de l'utilisateur
champs #4: l'identifiant du groupe de l'utilisateur
champs #5: informations complementaires ( nom complet,email,tel,...)
champs #6: le repertorie personnel de l'utilisateur
champs #7: le shell de l'utilisateur
```

* la structure /etc/shadow comprend 9 champs:

```
champs #1:
      le nom d'utilisateur
champs #2:
      le mot de passe crypté
champs #3:
      date de creation du mot de passe
champs #4:
      periode de changement de mot de passe ( 0 : pas d'obligation )
champs #5:
      date d'expiration de mot de passe
champs #6:
      periode apres l'expiration ou on est obligé de changer le mot de passe
champs #7-->#9:
      non utilisés
```

1) On ajoute les deux utilisateurs dans /etc/passwd et /etc/shadow :

on ouvre /etc/passwd avec la commande:

```
$ sudo vi etc/passwd
#puis on ajoute les lignes:
membre1:x:1002:1006:Premiere membre du groupe:/home/membre1/:/bin/bash
membre2:x:1003:1006:Second membre du groupe:/home/membre3/:/bin/bash
#1006 est l'identifiant du groupe mongroupTP
```

on ouvre /etc/shadow avec la commande:

```
$ sudo vi etc/shadow
#puis on ajoute les lignes:
membre1:::0:99999:7:::
membre2:::0:99999:7:::
#les mots de passe seront initialisé dans l'étape suivante
```

2) Pour le positionnement du mot de passe:

On change le mot de passe de chaque utilisateurs avec les commande

```
$ sudo passwd membre1
$ sudo passwd membre2
#on fait ceci pour pouvoir les crypter puis les stocker dans /etc/shadow
```

3) Création des répertoires personnels:

```
$ sudo mkdir /home/membre1/ /home/membre2/
```

4) Changement de propriétaire:

```
$ sudo chown membre1:mongroupTP /home/membre1/
$ sudo chown membre2:mongroupTP /home/membre2/
```

5) Copie des fichiers d'environnement (les fichiers contenus dans /etc/skel):

```
$ sudo cp /etc/skel/.* /home/membre1/
$ sudo cp /etc/skel/.* /home/membre2/
#on utilise .* car les fichiers qui nous interesse commencent par un point
```

Exercice 3. L'administrateur peut-il connaître le mot de passe d'un utilisateur ? Peut-il le changer ?

Non, l'administrateur ne peut pas savoir le mot de passe d'un utilisateur, tout les mots de passe sont cryptés, par contre il peut bien le modifier avec la commande:

```
$ sudo passwd nom_utilisateur
```

Exercice 4. Que fait la commande `id` ?

La commande id affiche des informations sur l'utilisateur courant ou bien l'utilisateur passé en argument, les information affichés sont le UID,le login les GID les noms des groupes

Exercice 5. Donner les étapes nécessaires fermer le compte d'un utilisateur.

Pour l'epecher de se connecter :

On ouvre le fichier /etc/shadow et on remplace le champs de mot de passe par ! Ou *.
Puis on ouvre le fichier /etc/passwd et on remplace le shell par /usr/sbin/nologin

Pour supprimer le compte :

On ouvre les fichiers /etc/shadow et /etc/passwd puis on supprime les lignes concernant l'utilisateur puis on supprime son repertorie personnel, on peut aussi faire tout ça en une seule commande : `$ sudo userdel nom_utilisateur`

- `groupadd` : pour créer un nouveau groupes
- `groupdel` : pour détruire un groupe.
- `useradd` : pour créer un nouvel utilisateur
- `userdel` : pour effacer un utilisateur

Exercice 6. Consulter les pages de manuels de ces commandes et utiliser les pour créer un nouveau compte d'un utilisateur puis le détruire.

La creation du groupe et de l'utilisateur:

```
jean@jean /home $ sudo groupadd groupeEx6
jean@jean /home $ sudo useradd utilisateurEx6 -g groupeEx6
```

La verification du creation du groupe et de l'utilisateur:

```
jean@jean ~ $ tail -n 1 /etc/group
groupeEx6:x:1007:
jean@jean ~ $ tail -n 1 /etc/passwd
utilisateurEx6:x:1004:1007::/home/utilisateurEx6:
```

La suppression du groupe et de l'utilisateur

```
jean@jean /home $ sudo userdel utilisateurEx6
jean@jean /home $ sudo groupdel groupeEx6
```

Exercice 7. Connectez-vous au système comme un un des utilisateur créé précédément. et créer un fichier vide nommé file dans le répertoire de connexion.

a) Quels sont les droits des autres sur ce fichier ?

b) Modifier les droits sur files de sorte à permette au groupe de le lire et le modifier. Les autres utilisateurs n'ont aucun droit sur ce fichier. (Donner les deux versions des commandes : masque octal et notation symbolique).

c) Appliquer la commande les séries des commandes suivantes et noter la différence :

Série 1) chmod 200 file ; chmod u=r file

Série 2) chmod 200 file ; chmod u+r file

Série 3) chmod 220 file , chmod u=r file

ci) Donner au propriétaire le droits de lire et écrire le fichier, au groupe le droit de lire t l'exécuter et aux autres le droit de l'exécuter.

cii) Quel est le rôle du droit d'exécution sur un répertoire ? Donner un scénario d'exemple qui illustre droit.

```
$ sudo login membre1 # pour se connecter en tantque l'utilisateur membre1
$ touch file # pour crée le fichier
```

a) Les droits des autres sur ce fichier sont uniquement le droit de lecture (r), on peut le verifier avec la commande : `$ ls -l file`

b)

```
$ chmod 660 file #avec la notation octal
$ chmod g=rw,o= file #avec la notation symbolique
```

c)

Serie 1 : La notation octal donne seulement le droit de modifier a l'utilisateur alors que la notation symbolique lui donne uniquement le droit de lecture

Serie 2 : La notation octal donne seulement le droit de modifier à l'utilisateur alors que la notation symbolique lui ajoute le droit de lecture

Serie 3 : La notation octal donne le droit de modification à l'utilisateur et au groupe, alors que la notation symbolique retire le droit de motification de l'utilisateur et lui donne le droit de lecture a sa place

ci)

```
$ chmod a=wx,g=rx,o=x file
```

a) Le droit d'execution sur un repertoire indique si ou pas un utilisateur possede le droit de traverser le repertoire (utilisation du commande cd)

```
membre1@jean ~ $ mkdir test
membre1@jean ~ $ chmod a-x test/
membre1@jean ~ $ cd test/
-bash: cd: test/: Permission denied
membre1@jean ~ $
```

Exercice 8 Utiliser la commande `umask` pour donner le droit de lecture seulement aux membres de votre groupe (linux) et aucun droit aux autres.

```
$ umask g+r,o-rwx
```

Exercice 9. Consulter les pages de manuels de la commande `newgrp` et donner un exemple de son utilisation. Comment vérifier que le groupe actif de l'utilisateur a bien changer ?

Cette commande permet à l'utilisateur de se connecter à partir d'un groupes secondaires.

```
#Pour tester cette commande j'ai ajouté l'utilisateur membre1 à un groupe da2i
```

Pour utiliser la commande :

```
$ newgrp da2i
```

Pour verifier que le groupe actif a bien changer avec la commande `id` comme suit:

```
membre1@jean ~ $ id
uid=1002(membre1) gid=1006(mongroupTP) groups=1006(mongroupTP),1002(da2i)
membre1@jean ~ $ newgrp da2i
membre1@jean ~ $ id
uid=1002(membre1) gid=1002(da2i) groups=1006(mongroupTP),1002(da2i)
membre1@jean ~ $
```

Partie 2 : Configuration d'interface réseau

Exercice 1

1. Donner les adresses MAC des interfaces Ethernets disponibles sur votre machine.
2. Les cartes Ethernets disponibles sur la machine sont-elles fabriquées par le même constructeur ?

1) Ma machine possede une seule carte Ethernet dont l'addresse MAC est : (`00:8c:fa:6b:3a:ef`)

2) Je possede une seule carte, mais si on veut savoir si deux cartes reseaux ont le meme constructeur on compare les 24 premiers bits (3 premiers octet)

exemple en comparant l'adress MAC de ma carte Ethernet avec celle de Wi-Fi qui possede l'adresse 64:5a:04:75:ab:4f on remarque clairement qu'il ne sont pas fabriquées par le meme constructeur

L'exercice 2 necessite avoir un hub, un cable ethernet,... et je ne possede pas de ces materiels

Partie 3 : Introduction à la programmation en shell bash

Exercice 1 : Gestion de sauvegarde

Ecrire un programme shell nommé `sauvegardeTxt` qui permet de *copier* tous les fichiers dans le compte utilisateur (e.g. compte étudiant) et qui se terminent par le suffixe `.txt` dans un répertoire nommé `~/.BACKUP`. Si le répertoire `BACKUP` n'existe pas alors la commande doit le créer.

Modifier le programme précédent afin de ne pas écraser les fichiers existants dans le répertoire `BACKUP`.

Version 1:

sauvegardeTxt_V1.sh

```
homeDirectory='/home/membre1/';
backupFolder="$homeDirectory.BACKUP/";
if [ ! -d $backupFolder ]; # si le repertoir n'existe pas
    then mkdir $backupFolder
else
    rm -f $backupFolder*; #on supprime s'il existent des fichiers
fi
for fichier in `find $homeDirectory -type f -iwholename '*.txt'`
    do
        cp $fichier $backupFolder/;
done
echo "L'opération a réusssi";
```

Version 2:

sauvegardeTxt_V2.sh

```
homeDirectory='/home/membre1/';
backupFolder="$homeDirectory.BACKUP/";
if [ ! -d $backupFolder ]; # si le repertoir n'existe pas
    then mkdir $backupFolder
fi
for fichier in `find $homeDirectory -type f -iwholename '*.txt'`
    do
        cp $fichier $backupFolder/ 2>/dev/null;
        #on remplace le fichier s'il existe
        #2>/dev/null pour ne pas afficher les erreurs
done
echo "L'opération a réusssi";
```

Exercice 2 : Poubelle

Développer une commande nommée `poubelle` qui permet de transférer les fichiers à effacer dans un répertoire nommé *trash*. La syntaxe de cette commande est la suivante :

1. `poubelle f1 f2 f3 …. fn` a pour effet de transférer les fichiers `f1` à `fn` dans le répertoire *trash*.
2. `poubelle -f` a pour effet d'effacer le contenu du répertoire *trash*.

L'appel de la commande sans arguments a pour effet d'afficher un message d'aide décrivant la syntaxe correcte de la commande.

poubelle.sh

```
if [[ $# == 0 ]]; then #s'il y'a pas d'arguments
    echo "<---- AIDE ---->
Syntax d'utilisation : poubelle f1 f2 .. fn
exemple:
    poubelle index.php test.java
poubelle -f: permet de vider le dossier trash
-----------------------------------------";
elif [[ $# == 1 && $1 = '-f' ]]; then # s'il y'a un seul argument -f
    rm -rf trash/*;
else
    for i in $* ; do
        mv "$i" trash/ 2>/dev/null;
    done
fi
```

Pour créer une commande on peut soit utiliser la commande alias pour créer un alias temporaire, soit on l'enregistre dans ~/.bashrc pour etre permanent.

Donc aprés avoir crée un script qui fait le travaille il nous reste d'attacher ce script à un alias:

```
$ alias poubelle='bash ./poubelle.sh'
```

```
jean@jean ~/TP $ ls trash/
jean@jean ~/TP $ ls
a.php  b.java  c.py  poubelle.sh  trash
jean@jean ~/TP $ poubelle a.php b.java c.py
jean@jean ~/TP $ ls trash/
a.php  b.java  c.py
jean@jean ~/TP $ poubelle
<---- AIDE ---->
Syntax d'utilisation : poubelle f1 f2 .. fn
exemple:
        poubelle index.php test.java
poubelle -f: permet de vider le dossier trash
-----------------------------------------
jean@jean ~/TP $ poubelle -f
jean@jean ~/TP $ ls trash/
jean@jean ~/TP $
```

Exercice 3 : Arrêt de processus

Développer une commande nommée `pkill` qui a le fonctionnement suivant : la commande demande à l'utilisateur d'entrer le nom du processus à tuer. Tous les processus qui porte le nom désigné seront arrêter.

pkill.sh

```
echo "Entrer le nom du processus : ";
read pname;
nbrInstancePS=`pgrep $pname | wc -l`;
if [[ `pgrep $pname | wc -l ` == '0' ]]; then
    # on verifie que le processus existe
    echo "Le processus '$pname' n'est pas en cours d'execution";
else
    kill `pidof $pname` 2>/dev/null;
    # 2>/dev/null pour ne pas afficher les erreurs
    echo "Tous les processus '$pname' ont été terminé";
fi
```

Test:

```
jean@jean ~/TP $ pidof sublime_text
7749
jean@jean ~/TP $ alias pkill='bash ./pkill.sh'
jean@jean ~/TP $ pkill
Entrer le nom du processus :
sublime_text
Tous les processus 'sublime_text' ont été terminé
jean@jean ~/TP $ pidof sublime_text
jean@jean ~/TP $
```