SYSTEME ET RESEAU
Devoir 2

Licence Informatique à distance

B.Herrmann et G.Laville

Exercice 1 : Mise en place d'un serveur de fichiers

Dans cet exercice, on souhaite mettre en place un service de partage de fichiers par le biais de comptes utilisateurs distants. Une partie des fichiers seront accessibles par tous les utilisateurs, tandis que d'autres fichiers seront d'accès restreints.

Pour cela, nous aurons :

- des comptes : étudiants et enseignants
- des groupes : enseignants, un par type d'étudiant

Nous proposons de créer les différents utilisateurs et groupes suivants pour gérer les accès :

- trois comptes utilisateurs étudiants : `alice`, `bob` et `charlie`
- un compte utilisateur enseignement : `eve`
- `alice` et `charlie` appartiennent au groupe `licence3`
- `alice` et `bob` appartiennent au groupe `reinscription`
- `eve` appartient au groupe `enseignement`
- tous les utilisateurs ont pour groupe principal `accesSSH`
- chaque groupe a accès au répertoire lui correspondant dans `/home` : par exemple, le groupe `enseignement` a accès à `/home/enseignement`.
- chaque utilisateur a accès à un répertoire personnel dans `home` correspondant à son nom de connexion

Question 1 : Détailler la création des utilisateurs, des groupes et de leurs répertoires ainsi que les commandes utilisées, avec la méthode manuelle (pas d'utilisation de commandes comme `useradd` ou `groupadd`).

Remarque : tous les numéros d'utilisateurs et de groupe indiqués dans cette correction sont donnés à titre indicatif, et ne doivent par encore être présents sur votre système.

Création des groupes :

Listing 1– Fichier /etc/group

```
accesSSH: :9:
licence: :10:alice,charlie
reinscription: :11:alice,bob
enseignement: :12:eve
```

Création des utilisateurs :

Listing 2– Fichier /etc/passwd

```
alice: :20:9:Alice A.:/home/alice:/bin/bash
bob: :21:9:Bob B.:/home/bob:/bin/bash
charlie: :22:9:Charlie C.:/home/charlie:/bin/bash
eve: :23:9:Eve E.:/home/eve:/bin/bash
```

Création des répertoires personnels et attribution des permissions :

Si l'on souhaite avoir les fichiers par défaut dans les nouveaux répertoires utilisateurs (.bashrc, .bash_profile, ...) :

```
# Si on souhaite avoir les fichiers par defaut
# (.bash_profile, .bash_rc, ...)
cp -a /etc/skel /home/alice
cp -a /etc/skel /home/bob
cp -a /etc/skel /home/charlie
cp -a /etc/skel /home/eve
```

Sinon pour créer des répertoires utilisateurs vides :

```
# Sinon, pour des nouveaux repertoires vides
mkdir -p /home/alice
mkdir -p /home/bob
mkdir -p /home/charlie
mkdir -p /home/eve
```

Permissions :

```
chown -R alice:accesSSH /home/alice
chown -R bob:accesSSH /home/bob
chown -R charlie:accesSSH /home/charlie
chown -R eve:accesSSH /home/eve
chmod 700 /home/alice
chmod 700 /home/bob
chmod 700 /home/charlie
chmod 700 /home/eve
```

Création des répertoires pour les groupes (à faire pour chacun des groupes, ici "accessSSH") :

```
mkdir -p /home/accesSSH
chgrp accessSSH /home/accessSSH
chmod 770 /home/accessSSH
\end{file}
```

Question 2 : Écrire un script, lancé à la connexion d'un utilisateur, qui propose à l'utilisateur de saisir un nom de répertoire dans `home` vers lequel se déplacer ("licence3" par exemple). Si une réponse vide est saisie, ce script laisse l'utilisateur dans son répertoire personnel.

Listing 3– /etc/profile.d/choose_directory.sh

```
#!/bin/bash

read -p "Please enter a starting directory (leave empty for default): "

if [ -z "$REPLY" ]; then
  # Nothing to do
else
  cd /home/$REPLY
fi
```

Question 3 : Sur ce système, il est possible de positionner trois `umask` différents pour les utilisateurs : 044, 066 ou 077. Expliquer quel sera la conséquence de chacune de ces valeurs en terme de permissions des nouveaux fichiers, et donc de partage des données.

La valeur de l'umask est soustraite à une valeur par défaut, qui est 777 pour les répertoires, et 666 pour les autres fichiers, de manière à pouvoir appliquer par défaut les droits d'exécution dans le premier cas, sans risque de sécurité dans les autres cas.

Cette opération donne donc les résultats suivants en terme de permissions pour chacune des valeurs d'umask proposées dans l'énoncé :

- *umask 044 : Permissions à 755 (rwxr-xr-x) pour les répertoires, à 644 (rw-r–r–) pour les autres fichiers. Cela signifie que le propriétaire a tous les droits possibles par défaut sur ses fichiers, tandit que tous les autres utilisateurs ne peuvent que consulter le contenu de ses fichiers et répertoires.*
- *umask 066 : Permissions à 711 (rwx–x–x) pour les répertoires, à 600 (rw——) pour les autres fichiers. Par rapport aux permissions précédentes, le groupe et les autres n'a cette fois plus la permission de lecture sur les répertoires et les fichiers : cela signifie qu'ils ne peuvent accéder qu'à des chemins connus de fichiers sur lesquels les droits en lecture ont été positionnés par ailleurs.*
- *umask 077 : Permissions à 700 (rwx——) pour les répertoires, à 600 (rw——) pour les autres fichiers. Le groupe et les autres n'ont plus aucune permissions sur les fichiers créés par l'utilisateur par défaut, tout partage de données est impossible.*

Exercice 2 : Informations utilisateurs

Question 1 : Donner une commande permettant de connaître le répertoire personnel de l'utilisateur `root`, ainsi que son affichage sur votre système.

```
$ grep root /etc/passwd | cut -d: -f6
/root
```

Question 2 : Donner une commande permettant de connaître les groupes auxquels appartient votre utilisateur courant, ainsi que son affichage sur votre système.

```
$ groups
glaville adm cdrom sudo dip plugdev lpadmin sambashare
```

Question 3 : Quels sont les utilisateurs définis dans /etc/passwd qui ont un identifiant inférieur à 1000 ? Pourquoi ?

Ces utilisateurs sont root, bin, sys, pool... (la liste peut varier suivant le système et les services installés). Il s'agit d'utilisateurs dits "système", qui n'ont pas le droit de se connecter (aucun mot de passe), mais permettent de faire tourner certains services en tant que simple utilisateur plutôt que root.

Exercice 3 : Utilisation de filtres

Soit le fichier texte `etudiants.txt` qui contient la liste des étudiants de L3. Chaque ligne de ce fichier est de la forme :

`nom:prénom:provenance:année naissance`

Les champs sont séparés par des deux points.

On utilise le fichier `etudiants.txt` de contenu :
Dupond :Ferdinand :L2 :1994
Merquer :Amélie :BTS :1993
Dupont :Floriane :L2 :1994
Dupond :Agnès :IUT :1992
Gonzales :Armand :L2 :1994
Térieur :Alain :L2 :1993
Urand :Bérénice :IUT :1994
Bondel :Hubert :L2 :1993
Laison :Fabrice :BTS :1990
Airon :Wiliam :L2 :1995
Garries :Gabriel :IUT :1993
Crison :Patrick :L2 :1993

Répondre aux questions suivantes en utilisant à chaque fois une ligne de commande shell :

Question 1 : Afficher le nombre d'étudiants en L3.

```
$ cut -d: -f3 etudiants.txt | grep L3 | wc -l
```

Question 2 : Afficher tous les étudiants en provenance de L2.

```
$ grep L2 etudiants.txt
```

Question 3 : Afficher les étudiants, en triant le fichier selon leur année de naissance.

```
$ sort -d: -k 4 etudiants.txt
```

Question 4 : Afficher tous les étudiants ne provenant pas d'IUT.

```
$ grep -v "IUT" etudiants.txt
```

Question 5 : Afficher uniquement l'année de naissance du premier étudiant "Dupond" apparaissant dans le fichier.

```
$ grep -m1 "Dupond" etudiants.txt | cut -d: -f4
```

Question 6 : Afficher le prénom du premier étudiant né en 1993 apparaissant dans le fichier.

```
$ grep -m1 "1993" etudiants.txt | cut -d: -f2
```

Exercice 4 : Gestion des flux d'entrée/sortie

Question 1 : Ecrire une commande qui simultanément liste les noms des étudiants du fichier `etudiants.txt` de l'exercice 3 à l'écran et les écrit dans un fichier `nomsEtudiants.txt`.

```
$ cut -d: -f1 etudiants.txt | tee nomsEtudiants.txt
```

Question 2 : Reprendre la commande précédente pour qu'elle affiche en plus le nombre d'étudiants traités.

```
$ cut -d: -f1 etudiants.txt | tee nomsEtudiants.txt | wc -l
```

Question 3 : Ecrire une commande qui transforme les noms des étudiants du fichier `etudiants.txt` en majuscule et les affiche à l'écran.

```
$ cut -d: -f1 etudiants.txt | tr 'a-z' 'A-Z'
```

Question 4 : Ecrire un script qui lit sur l'entrée standard les informations concernant un étudiant et ajoute cette ligne à la fin du fichier `etudiants.txt`.

Listing 4– read_informations.sh

```
1 echo "entrer le nom de l'etudiant"
2 read nom
3 echo "entrer le prenom de l'etudiant"
4 read prenom
5 echo entrer "le niveau de l'etudiant"
6 read niveau
7 echo "entrer l'année de naissance de l'etudiant"
8 read naissance
9 echo "$nom:$prenom:$niveau:$naissance" >> etudiants.rtf
```

Exercice 5 : Ecriture d'un fichier de commandes : création d'un exécutable

Ecrire un fichier de commandes qui réalise les tâches suivantes :

- prendre un répertoire en paramètre,
- vérifier qu'il existe et qu'il s'agit bien d'un répertoire,
- se déplacer dans ce répertoire qui devient le répertoire courant,
- vérifier s'il existe déjà un répertoire `bin` à l'intérieur du répertoire courant, le créer sinon,
- vérifier s'il existe déjà un répertoire `src` à l'intérieur du répertoire courant, le créer sinon,
- afficher à l'utilisateur les fichiers contenus dans le répertoire `bin` du répertoire courant qui ne sont pas exécutables,
- créer un répertoire `compilation` dans le répertoire courant,
- compiler chaque fichier source `C` (`<nom du fichier>.c`) du répertoire `src` contenu dans le répertoire courant. Chaque fichier objet obtenu prendra le nom `<nom du fichier>.o` et sera stocké dans le répertoire `compilation` du répertoire courant,
- générer un fichier exécutable nommé `exec-<nom du répertoire>` dans le répertoire `bin` du répertoire courant à partir des fichiers objets générés précédemment dans le répertoire `compilation`.

Indications :
Compilation d'un fichier source C :
`gcc <nom du fichier>.c -o <nom du fichier>.o`

Génération d'un binaire ou exécutable à partir de fichiers objets :
`gcc <nom du fichier1>.o <nom du fichier2>.o -o <nom de l'executable>`

Listing 5– compilation.sh

```
#!/bin/bash

# Prendre un répertoire en paramètre
if [ $# -ne 1 ]; then
    echo "Usage: $0 <repertoire>"
    exit 1
fi

rep=$1

# Vérifier qu'il existe et qu'il s'agit bien d'un répertoire
if [ ! -d "$rep" ]; then
    echo "$rep n'est pas un repertoire, interruption..."
    exit 2
fi

# Se déplacer dans ce répertoire
cd "$rep"

# Créer bin/ si nécessaire
if [ ! -e "bin" ]; then
    echo "Creation de bin/..."
    mkdir bin
fi

# Créer src/ si nécessaire
if [ ! -e "src" ]; then
    echo "Creation de src/..."
    mkdir src
fi

# Afficher les fichiers de bin/ non exécutables
for f in bin/*; do
    if [ ! -x "$f" ]; then
        echo "$f n'est pas executable"
    fi
done

# Créer un répertoire compilation
mkdir compilation

# Compiler chaque fichier source C de src/
for source in src/*.c; do
    object=$(basename $source | cut -f2 -d.)
    gcc "$source" -o "compilation/$object"
done

# Générer un fichier exécutable nommé dans bin/
gcc compilation/*.o -o bin/exec-$(basename($rep))
```