Module GNU/Linux CEFIPA : TP n°2 - Droits et processus

Corrigé

Nicolas Burrus

26 Avril 2005

Question 1

Ajouter un utilisateur nommé `tp2`. Vérifier qu'il est présent dans `/etc/passwd`. Quel est son UID ? Ajouter un groupe `tp2-group2`. Vérifier qu'il est présent dans `/etc/group`. Quel est son GID ?

Réponse :

`adduser tp2` ajoute un utilisateur nommé `tp2` et un groupe nommé `tp2` qui sera le groupe principal de l'utilisateur.
`groupadd tp2-group2` ajoute le groupe `tp2-group2`. `less /etc/group` devrait faire apparaître une ligne contenant `tp2-group2` à la fin. On peut lire son GID sur cette ligne.

Question 2

En utilisant `usermod`, faire en sorte que l'utilisateur `tp2` fasse partie du groupe `tp2` et du groupe `tp2-group2`. Vérifiez le résultat.

Réponse :

`usermod -g tp2 -G tp2-group2 tp2` spécifie que le groupe principal de l'utilisateur `tp2` est `tp2`, et qu'il appartient à un groupe additionnel nommé `tp2-group2`.
`groups tp2` permet de voir la liste des groupes auquel il appartient.

Question 3

Placez vous sous l'identité de l'utilisateur `tp2`. Pouvez-vous créer un fichier vide `/usr/test1` ? Pourquoi ? Pouvez-vous créer un fichier vide `/tmp/test1` ? Pourquoi ?
Réponse :

`ls -l / | grep usr` donne :
`drwxr-xr-x 12 root root 4096 Feb 10 2004 usr/`. Cela signifie que seul l'utilisateur propriétaire (root) a les droits en écriture (w) dans `/usr`. Il n'est donc pas possible d'écrire dedans sous l'identité de `tp2`.

`ls -l / | grep tmp` donne :
`drwxrwxrwx 13 root root 16384 May 2 22 :46 tmp/`. Ici tout le monde a les droits d'écriture, le fichier peut donc être crée par `tp2`.

Question 4

Faites en sorte que l'utilisateur `knoppix` ne puisse pas lire le fichier `/tmp/test1`. Faites en sorte que l'utilisateur `knoppix` ne puisse pas lister le contenu de `/home/tp2`.

Réponse :

Il suffit de s'assurer que le fichier ne soit pas accessible en lecture ni écriture pour les utilisateurs non propriétaire du fichier, soit :
`chmod 600 /tmp/test1` ou `chmod 640 /tmp/test1` pour laisser les droits aux utilisateurs appartenant au groupe `tp2`.

Question 5

Donnez les droits du fichier `/bin/su` sous forme numérique (ex : 0644). Qu'est ce que ça signifie ?

Réponse :

```
ls -l /bin/su
-rwsr-xr-x 1 root root 23416 Feb 1 23 :33 /bin/su
```
Si le 's' était un 'x', on aurait 755, correspondant à lecture et exécution pour tout le monde, plus écriture pour le propriétaire du fichier. Le 's' signifie que le programme s'exécutera sous l'identité du propriétaire du fichier et pas de l'utilisateur qui lance le programme. Un chiffre supplémentaire en préfixe permet de l'indiquer, la réponse finale est donc 4755.

Question 6

A quel utilisateur et à quel groupe appartiennent les fichiers dans `/sbin` ?

Réponse :

```
ls -l /sbin
```
On remarque que tous les fichiers appartiennent à l'utilisateur root et au groupe root.

Question 7

Vérifier que `init` est bien le processus parent de tous les autres. Quel est son PID ?

Réponse :

`pstree -p` permet de voir la structure arborescente des processus, et affiche les PID correspondants. Le PID de init est 1.

Question 8

En utilisant `top`, donner les PID et les commandes associées aux 2 processus qui utilisent le plus de mémoire (RES).

Réponse :

Taper '?' dans top pour affiche l'aide. 'M' permet de trier les processus selon leur occupation mémoire. 'F' puis 'q' aboutit au même résultat.

Question 9

`top` consomme des ressources. Faites en sorte que sa priorité soit de 18. Est-ce une forte priorité ? Quel est la priorité du processus associé à la commande `/usr/X11R6/bin/X` ?

Réponse :

`renice 18 ‘pidof top‘`. C’est une priorité très faible (le minimum est 20).

`ps axo nice,command | grep X` permet de voir la priorité de X. En général elle vaut 0.

Question 10

Consulter le man de `kill`. Placez le processus en arrière plan sans le terminer. Consulter le man de `ps`. Placez le également en arrière plan sans le terminer. Ré-affichez le man de `kill`. Ouvrez un autre terminal. Envoyez le signal SIGTERM au processus affichant le man de `kill`. Retournez dans le premier terminal, réactivez le man de `ps`, et quittez le normalement.

Réponse :

```
(terminal 1) $ man kill
[Ctrl-Z]
zsh: suspended  man kill
(terminal 1) $ man ps
[Ctrl-Z]
zsh: suspended  man ps
(terminal 1) $ jobs # liste les taches lancées par le shell
[1]  - suspended  man kill
[2]  + suspended  man ps
(terminal 1) $ fg %1 # reaffiche le man de kill

(terminal 2) $ ps aux | grep "man kill"
tp2       1368  0.0  0.1   2024  1212 pts/6    T    23:57   0:00 man kill
(terminal 2) $ kill -TERM 1368 # ou kill -15 1368 ou kill 1368

(terminal 1) $ jobs
[2]  + suspended  man ps
(terminal 1) $ fg %2
```

Question 11

Executez la commande `less /etc/mime.types`. Déplacez vous dans le fichier. Ouvrez un autre terminal. Envoyez le signal SIGSTOP au processus associé à `less`. Retournez dans le premier terminal. Pouvez vous continuer à naviguer dans le fichier ? Faites en sorte que le processus réponde à nouveau.

Réponse :

```
(terminal 1) $ less /etc/mime.types

(terminal 2) $ ps aux | grep "less /etc/mime.types"
tp2  1491  0.0  0.0  1844  676 pts/6  S+ 00:02  0:00 less /etc/mime.types
(terminal 2) $ kill -STOP 1491

(terminal 1) # less est figé

(terminal 2) $ kill -CONT 1491

(terminal 1) # less est reparti
```

Question 12

Que fait la commande `kill -9 -1`, exécutée sous l'identité de l'utilisateur `tp2` ?

Réponse :

Le -9 signifie que l'on souhaite envoyer le signal SIGKILL, causant la mort des processus qui le reçoivent, et le PID spécial -1 signifie que l'on veut l'envoyer à tous les processus. La commande tue donc tous les processus qu'elle peut tuer (un utilisateur ne peut pas tuer les processus d'un autre utilisateur).