TP 4 - Admin Système

Piere Dubreuil & Lucie Mourer

Date : 13/03/20

Exercice 1. Gestion des utilisateurs et des groupes

1. Commencez par créer deux groupes groupe1 et groupe2

```
sudo addgroup groupe1 && addgroup groupe2
```

On fera attention à écrire les noms de groupe et les noms d'utilisateurs en minuscule

2. Créez ensuite 4 utilisateurs u1, u2, u3, u4 avec la commande useradd, en demandant la création de leur dossier personnel et avec bash pour shell

```
sudo useradd u1 -m -s /bin/bash
sudo useradd u2 -m -s /bin/bash
sudo useradd u3 -m -s /bin/bash
sudo useradd u4 -m -s /bin/bash
```

Le manuel de `useradd` permet de savoir que `-m` demande al création du dossier personnel et `-s /bin/bash` indique bash pour shell

3. Ajoutez les utilisateurs dans les groupes créés : - u1, u2, u4 dans groupe1 et - u2, u3, u4 dans groupe2

```
sudo gpasswd -M u1,u2,u4 groupe1
sudo gpasswd -M u2,u3,u4 groupe2

sudo usermod -a -G groupe1 u1
sudo usermod -a -G groupe1 u2
sudo usermod -a -G groupe1, groupe2 u4
sudo usermod -a -G groupe2 u2
sudo usermod -a -G groupe2 u3
```

`usermod u1 -a -G groupe1` permet de rajouter un utilisateur dans un groupe mais il faut faire un ligne par utilisateur

`gpasswd` permet d'être plus concis mais ne fait que définir les utilisateurs d'un groupe (il les ajoute s'ils n'existent pas mais il supprime tous les autres du groupe).

4. Donnez deux moyens d'afficher les membres de groupe2

```
getent group groupe2 | awk -F: '{print $4}'

group "groupe2:" /etc/group | awk -F: '{print $4}'
```

`awk -F : '{print $4}'` permet de split les paramètres avant les informations qui nous intéressent pour un meilleur affichage.
Pour la deuxième méthode nous ajoute un `:` afin de ne récupérer que les groupes et non les utilisateurs ayant le même groupe.

5. Faites de groupe1 le groupe propriétaire de /home/u1 et/home/u2 et de groupe2 le groupe propriétaire de /home/u3 et /home/u4

```
sudo chgrp -R groupe1 /home/u1
sudo chgrp -R groupe1 /home/u2
sudo chgrp -R groupe2 /home/u3
sudo chgrp -R groupe1 /home/u4
```

`chgrp` permet de modifier le groupe propriétaire de la cible sans changer le propriétaire du ficher (différence avec `chown`)

6. Remplacez le groupe primaire des utilisateurs : - groupe1 pour u1 et u2 - groupe2 pour u3 et u4

```
sudo usermod -g groupe1 u1
sudo usermod -g groupe1 u2
sudo usermod -g groupe2 u3
sudo usermod -g groupe2 u4
```

C'est le paramètre `-g` de `usermod` qui peremet de remplacer le groupe primaire des utilisateurs

7. Créez deux répertoires /home/groupe1 et /home/groupe2 pour le contenu commun aux groupes, et mettez en place les permissions permettant aux membres de chaque groupe d'écrire dans le dossier associé.

```
sudo mkdir /home/groupe1
sudo chgrp -R groupe1 /home/groupe1
sudo chmod -R g+w /home/groupe1

sudo mkdir /home/groupe2
sudo chgrp -R groupe2 /home/groupe2
sudo chmod -R g+w /home/groupe2
```

`g+w` indique que on on peut écrire dans le dossier**groupe2** tant que l'on appartient au*groupe2* (logique de group + write)

8. Comment faire pour que, dans ces dossiers, seul le propriétaire d'un fichier ait le droit de renommer ou supprimer ce fichier ?

```
sudo chmod +t /home/groupe1
sudo chmod +t /home/groupe2
```

Par cette commande on active le*sticky bit* qui donne au propriétaire du fichier seul le droit de le renommer ou de le supprimer. On retrouvera le *sticky bit* `t` lors de l'affichage des droits du dossier

9. Pouvez-vous vous connecter en tant que u1 ? Pourquoi ?

```
su u1
```

On ne peut pas se connecter en tant qu'u1 car on a pas configuré de mot de passe pour cette utilisateur lors de sa création

10. Activez le compte de l'utilisateur u1 et vérifiez que vous pouvez désormais vous connecter avec son compte.

```
sudo passwd u1
```

-> *réponse terminal*

```
Enter new UNIX password:
Retype new UNIX password:
passwd : le mot de passe a été mis à jour avec succès
```

11. Quels sont l'uid et le gid de u1?

```
id u1
```

-> *réponse terminal*

```
uid=1001(u1) gid=1004(groupe2) groups=1002(groupe2)
```

12. Quel utilisateur a pour uid 1003?

```
id 1003
```

-> *réponse terminal*

```
uid=1003(u1) gid=1001(groupe1) groups=1001(groupe1)
```

13. Quel est l'id du groupe groupe1?

```
getent group groupe1 | awk -F: '{printf "%s id = %d\n", $1, $3}'
```

14. Quel groupe a pour guid 1002 ? (⚠ Rien n'empêche d'avoir un groupe dont le nom serait 1002...)

```
getent group 1002
```

Si on ne passe qu'un nombre comme dernier paramètre, `gentent` va chercher seulement les guid correspondant et non les noms de groupe correspondants.

15. Retirez l'utilisateur u3 du groupe groupe2. Que se passe-t-il ? Expliquez.

```
sudo gpasswd groupe2 u3
```

-> *réponse terminal*

```
Retrait de l''utilisateur u3 du groupe groupe2
```

Avec la commande `id u3` on observe que l'utilisateur conserve le **gui** de groupe 2 et qu'il est toujours associé à **groupe2**.

Or , la commande `getent group groupe2` confirme bien que u3 ne fait plus parti du groupe groupe2.

16. Modifiez le compte de u4 de sorte que :

— il expire au 1er juin 2020 — il faut changer de mot de passe avant 90 jours — il faut attendre 5 jours pour modifier un mot de passe — l'utilisateur est averti 14 jours avant l'expiration de son mot de passe — le compte

sera bloqué 30 jours après expiration du mot de passe

```
sudo useradd u4 -e 2020-06-01
sudo chage u4 -M 90 -m 5 -w 14 -I 30
```

`chage` : - M: nombre jour maximum avant changement de mot de passe - m: nombre jour minimum avant changement de mot de passe - w: nombre jour de notification avant le changement du mot de passe - I: nombre de jour d'inactivité après l'expiration du mot de passe et avant le blocage du compte

17. Quel est l'interpréteur de commandes (Shell) de l'utilisateur root?

```
cat etc/passxd | grep root | awk -F: '{printf "shell = %d\n", $7}'
```

-> *réponse terminal*

```
shell = /bin/bash
```

L'interpréteur de commandes de l'utilisateur root est donc bash. On cherche root dans le fichier passwd (`cat etc/passxd | grep root`) et on récupère seulement l'information concernant l'interpréteur (`| awk -F: '{printf "shell = %d\n", $7}'`)

18. à quoi correspond l'utilisateur nobody ?

L'utilisateur `nobody` est un utilisateur dont qui n'a aucune permission sur les fichiers/dossiers du système. Il est utiliser pour exécuter des scripts qui peuvent avoir un impact sur le système pour en minimiser les dégats (ex: httpd, si le serveur est piraté les dégats seront minimes).

19. Par défaut, combien de temps la commande sudo conserve-t-elle votre mot de passe en mémoire ? Quelle commande permet de forcer sudo à oublier votre mot de passe ?

Par defaut la commande `sudo` conserve le mot de passe 15 minutes en mémoire.

Pour forcer `sudo` à oublier le mot de passe (en quittant un sesion par exemple) on utilise la commande :

```
sudo -K
```

Où `-K` est un paramètre qui force de `remove` le mot de passe (`-k` va le remettre à 0).

Exercice 2. Gestion des permissions

1. Dans votre $HOME, créez un dossier test, et dans ce dossier un fichier

"*fichier*" contenant quelques lignes de texte. Quels sont les droits sur test et fichier ?

```
mkdir ~/test
nano ~/test/fichier
```

retour dans le terminal

```
ll ~/
```

---> Sortie

```
drwxr-xr-x pir pir 19 Mar 13 16:32 test
```

User : read, write and execute Groupe : read and execute All : read and execute

```
ll ~/test/
```

---> Réponse terminal

```
-rw-r--r-- pir pir 19 Mar 13 16:32 fichier
```

User : read and write Groupe : read All : read

Comme il s'agit d'un fichier on ne retrouve pas le `d` de directory

2. Retirez tous les droits sur ce fichier (même pour vous), puis essayez de le modifier et de l'a**ff**icher en tant que root. Conclusion ?

```
sudo chmod 000 ~/test/fichier
```

Nous pouvons encore lire et écrire dans le fichier en tant que root.

3. Redonnez vous les droits en écriture et exécution sur fichier puis exécutez la commande echo "echo Hello" > fichier. On a vu lors des TP précédents que cette commande remplace le contenu d'un fichier s'il existe déjà. Que peut-on dire au sujet des droits ?

```
sudo chmod u+wx ~/test/fichier
echo "echo Hello" > ~/test/fichier
```

On peut remplacer le contenu du fichier puisqu'on a le droit d'écriture.

4. Essayez d’exécuter le fichier. Est-ce que cela fonctionne ? Et avec sudo ? Expliquez.

```
cd test
./fichier

sudo ./fichier
```

On peut bien exécuter le fichier en tant qu'utilisateur ainsi qu'avec sudo. C'est normal on a ajouté le droit d'execution

5. Placez-vous dans le répertoire test, et retirez-vous le droit en lecture pour ce répertoire. Listez le contenu du répertoire, puis exécutez ou affichez le contenu du fichier fichier. Qu’en déduisez-vous ? Rétablissez le droit en lecture sur test

```
cd test
sudo chmod u-r .
ls
```

---> Réponse terminal

```
Impossible d''ouvrir le répertoire, permission non accordée
```

```
./fichier
```

---> Réponse terminal

```
Hello
```

On en deduit que même en enlevant les droits de lecture d'un dossier, on peut toujours garder les droits d'execution sur les fichiers à l'intérieur (à condition de connaitre leur existance au préalable évidement)

6. Créez dans test un fichier nouveau ainsi qu’un répertoire sstest. Retirez au fichier nouveau et au répertoire test le droit en écriture. Tentez de modifier le fichier nouveau. Rétablissez ensuite le droit en écriture au

répertoire test. Tentez de modifier le fichier nouveau, puis de le supprimer. Que pouvez- vous déduire de toutes ces manipulations ?

```
touch nouveau
mkdir sstest
sudo chmod u-w nouveau
sudo chmod u-w sstest
nano nouveau
sudo chmod u+r .
nano nouveau
rm nouveau
```

Il n'est pas possible de modifier le fichier*nouveau* (dans les deux cas).
Lors de la demande de suppression, on nous de mande de confirmer notre action : `rm: remove write-protected regular empty file 'nouveau'?`
Les droits d'un dossier ne s'appliquent donc pas aux fichiers contenu dans ce répertoire.

7. Positionnez vous dans votre répertoire personnel, puis retirez le droit en exécution du répertoire test. Tentez de créer, supprimer, ou modifier un fichier dans le répertoire test, de vous y déplacer, d'en lister le contenu, etc...Qu'en déduisez vous quant au sens du droit en exécution pour les répertoires ?

```
cd ~
sudo chmod u-x test
touch test/fichier2
rm test/fichier
cd test
ls test
ll test
```

Une fois la permission d'éxecution retirée, on ne peut plus rien faire sur les fichers présents dans **test** On peut lister le contenu du dossier mais ne ne peut pas récupérer les informations de ce dossier (cela sous entend que l'ont pourrait rentrer dans le dossier). Si on retire les droits d'éxécution sur un dossier en se plaçant dans le répertoire personnel, on ne peux plus acceder aux fichiers présents dans le dossier sauf en lecture.

8. Rétablissez le droit en exécution du répertoire test. Positionnez vous dans ce répertoire et retirez lui à nouveau le droit d'exécution. Essayez de créer, supprimer et modifier un fichier dans le répertoire test, de vous déplacer dans ssrep, de lister son contenu. Qu'en concluez-vous quant à l'influence des droits que l'on possède sur le répertoire courant ? Peut-on retourner dans le répertoire parent avec "cd .." ? Pouvez-vous donner une explication ?

```
cd test
sudo chmod u-x .
touch fichier2
rm fichier
cd sstest
ls
```

Une fois la permission d'éxecution retirée, on ne peut plus rien faire sur les fichers présents dans **test** et dossier à part le répertoire `..` qui est le répertoire parent. Une fois dans le repertoire courant, si on retire le droit d'éxécution, on ne peut plus que revenir dans le repertoire parent.

9. Rétablissez le droit en exécution du répertoire test. Attribuez au fichier fichier les droits suffisants pour qu'une autre personne de votre groupe puisse y accéder en lecture, mais pas en écriture.

```
cd ~
sudo chmod u+x test
sudo chmod g+r-w test/fichier
```

L'utilisateur à bien le droit d'éxecution sur le dossier**test**. Le groupe lui en revanche avec `g+r-w` peut seulent lire mais pas écrire dans le fichier **fichier**

10. Définissez un umask très restrictif qui interdit à quiconque à part vous l'accès en lecture ou en écriture, ainsi que la traversée de vos répertoires. Testez sur un nouveau fichier et un nouveau répertoire.

```
umask 077
```

A completer

11. Définissez un umask très permissif qui autorise tout le monde à lire vos fichiers et traverser vos répertoires, mais n'autorise que vous à écrire. Testez sur un nouveau fichier et un nouveau répertoire.

```
umask 022
```

12. Définissez un umask équilibré qui vous autorise un accès complet et autorise un accès en lecture aux membres de votre groupe. Testez sur un nouveau fichier et un nouveau répertoire.

```
umask 037
```

13. Transcrivez les commandes suivantes de la notation classique à la notation octale ou vice-versa (vous pourrez vous aider de la commande stat pour valider vos réponses) :

- chmod u=rx,g=wx,o=r fic ---> `chmod 534 fic`
- chmod uo+w,g-rx fic en sachant que les droits initiaux de fic sont r--r-x--- ---> `chmod 602 fic`
- chmod 653 fic en sachant que les droits initiaux de fic sont 711 `chmod uo-x,g+r fic`
- chmod u+x,g=w,o-r fic en sachant que les droits initiaux de fic sont r--r-x---

14. Affichez les droits sur le programme passwd. Que remarquez-vous ? En affichant les droits du fichier /etc/passwd, pouvez-vous justifier les permissions sur le programmepasswd?

```
ll -l /usr/bin/passwd
```

---> Réponse terminal

```
-rwxr-xr-x 1 root root  59640 mars 22 2019 /usr/bin/passwd*
```

Pour les plus rapides :

15. Access Control Lists (ACL) : suivez le tutoriel de cette page : https://doc.ubuntu-fr.org/acl.

16. Quotas disques : suivez le tutoriel de cette page : https://doc.ubuntu-fr.org/quota.