Droits des utilisateurs

Le mécanisme permettant au système d'être multi-utilisateur induit un accès sélectif et des niveaux de protection sur les fichiers. Le principe de base s'énonce ainsi : le propriétaire d'un fichier définit son droit d'accès. Tout fichier ou répertoire sur Ubuntu appartient à un utilisateur et à un groupe.

1. Utilisateurs et attributs de fichiers

a. Principes

Sous Ubuntu Linux, voici l'identification des types d'utilisateurs :

- Le propriétaire du fichier noté `user`.
- Le groupe d'appartenance du propriétaire noté `group`.
- Les autres ou le reste du monde noté `other`.

Pour visualiser l'appartenance d'un fichier, la commande `ls` option `-l` (format long) montre pour chaque fichier un ensemble d'arguments associés.

Exemple :

```
touch essai.txt

ls -l essai.txt
```

La première commande crée un fichier texte vide. Les deux sont effectuées par l'administrateur (le créateur du fichier est le propriétaire) :

```
-rw-r—r--  1  root root 0   sept 24   16:21 essai.txt
```

Le premier bloc de dix caractères informe de la signification des droits :

- La première lettre donne l'indication du type de fichier. Les plus courants sont un tiret (`-`) pour l'attribut d'un fichier ordinaire et un (`d`) pour un répertoire.
- Les trois lettres suivantes indiquent le propriétaire avec dans l'ordre : `r` ou `-` (droit en lecture ou non), `w` ou `-` (droit en écriture ou non), `x` ou `-` (droit en exécution ou non).
- Les trois lettres suivantes donnent exactement la même signification mais pour le groupe.
- Les trois dernières lettres s'appliquent au reste du monde.

La suite de l'affichage donne respectivement le nombre de liens (cette notion est vue dans le chapitre Session de travail en mode console), le propriétaire du fichier, le groupe d'appartenance, la taille du fichier en octets, la date et l'heure de la dernière modification du fichier et enfin le nom du fichier.

Plus précisément :

- Dans le cas d'un fichier : le droit de lecture (`r`) autorise la visualisation du contenu, le droit d'écriture (`w`) autorise la modification du contenu et le droit d'exécution (`x`) autorise (pour les fichiers exécutables évidemment)... son exécution.
- Dans le cas d'un répertoire : le droit de lecture (`r`) autorise le listing des fichiers qu'il contient, le droit d'écriture (`w`) autorise la création, modification et suppression des fichiers et le droit d'exécution (`x`) autorise la possibilité d'y aller ou de le "traverser".

b. Changement des attributs de fichiers

Rappel du principe de base : seul le propriétaire peut changer son droit d'accès au fichier. Seul l'administrateur système (`root`) possède tous les pouvoirs. Cela rend le système fiable car l'infection et la propagation d'un virus ne peut se faire que dans l'espace de propriété de l'utilisateur, d'où le fait de ne pas effectuer des opérations courantes ou à risque comme la lecture de courriers sous l'identité du `root`.

Le changement d'attributs s'effectue avec la commande `chmod` (`change mode`). On distingue traditionnellement deux modes d'utilisation.

Méthode par le nombre octal (base huit)

Le nombre octal se donne par trois bits, positionnés à 0 ou 1, représentatifs de puissances de 2 :

```
chmod nombre_octal nom_fichier
```

Exemple :

```
chmod 644 essai.txt
```

Qui s'explique de la façon suivante : il ne faut pas lire 644 mais 6 (le premier pour le propriétaire), 4 (le deuxième pour le groupe) et 4 (le dernier pour le reste du monde). En binaire 6 se code sur 110, soit en représentant les puissance de 2 :

$1*2^2 + 1*2^1 + 0*2^0 = 110$ (ou 4 + 2 + 0 = 6)

Un bit positionné à 1 donne la permission, 0 donne une interdiction. 110 donne donc la permission en lecture , en écriture et non en exécution soit dans notre exemple :

```
rw-
```

Notez un chiffre couramment utilisé pour les scripts exécutables : 755 ce qui donne tous les droits pour le propriétaire, les droits en lecture et exécution pour le groupe et le reste du monde.

Méthode par la définition symbolique

On aura cette fois la commande :

```
chmod [qui][opération][type_permission] nom_fichier
```

Avec :

qui : u pour user

opération : - pour une interdiction, + pour une permission

type_permission : `rwx` lecture, écriture et exécution

Exemple :

```
chmod o-r essai.txt
```

Cet exemple ôte la permission en lecture pour le reste du monde.

c. Changement de propriétaire ou de groupe

Pour "donner" un fichier à un autre utilisateur, il faut bien sûr en être propriétaire et avoir les droits (sauf le `root`) : on ne peut donner que ce qui nous appartient ! La commande :

```
chown [options] nouveau_propriétaire nom_fichier
```

s'utilise pour changer le propriétaire d'un fichier (`change owner`), alors que :

```
chgrp [options] nouveau_groupe nom_fichier
```

s'utilise pour changer le groupe d'un fichier (`change group`).

Exemples (on donne à max le fichier de léon) :

```
chown max essai.txt

chgrp max essai.txt

chown max.max essai.txt
```

La dernière commande exploite une fonctionnalité de la commande `chown` qui permet d'effectuer un raccourci en changeant en même temps le propriétaire et le groupe.

d. Droits supplémentaires

Parallèlement aux droits standards, des droits étendus existent et répondent à des besoins spécifiques :

- Pour un répertoire, restreindre la suppression au seul propriétaire : droit **sticky bit**.
- Pour un fichier binaire, exécution sous l'identité du propriétaire : droit **SUID** (`setuid`) ou sous l'identité du groupe : droit **SGID** (`setgid`).

Ce dernier point permet d'exécuter un programme (normalement un binaire et non un fichier de commandes) à partir d'un utilisateur quelconque nécessitant les droits d'un autre.

Pour affecter ou retirer ces droits, on utilise la commande `chmod` avec cette fois-ci sur quatre bits, le premier indiquant la nature du droit ou par une lettre spécifique en notation symbolique :

Droit	En octal	En symbolique
Sticky bit	1000	o+t
SUID	4000	u+s
SGID	2000	g+s

Le champ d'application du droit par la notation symbolique se fait clairement : `u` pour le SUID et `g` pour le SGID. Dans le cas du `sticky bit`, l'application porte sur le "reste du monde" soit `o`, partant du principe qu'un fichier sous Ubuntu appartient à un groupe du même nom que le propriétaire.

Exemples de commandes :

```
chmod o+t /home/donald/programmes/

chmod 4755 liste.sh

ls -l /usr/bin/chage
```

La première commande positionne le droit `sticky bit` pour le répertoire `programmes` situé dans le répertoire de l'utilisateur `donald`. La deuxième commande met les droits en exécution pour le fichier comprenant des commandes shell et positionne en même temps le droit SUID. Enfin la dernière commande montre que le droit SGID est mis pour la commande `chage` (modifie les informations de validité d'un mot de passe) pour le groupe `shadow` (note : le fichier `/etc/shadow` a pour propriétaire le `root` et comme groupe `shadow`).

Si le fichier/répertoire n'a pas les droits en exécution, le `s` et `t` apparaissent en majuscules.

2. Entraînement

L'utilisation et la compréhension des droits utilisateurs nécessitent un minimum de pratique. Afin de vous aider dans la maîtrise des commandes, voici une liste de **questions/exercices** (les solutions se trouvent en **Annexe 2**) :

Cadre de travail

Session ouverte en `root` sur une version Ubuntu serveur.

Questions

- Vérifiez l'endroit où vous êtes, votre identité de compte avec notamment l'UID et le GID par la commande `id`.
- Quelles sont les permissions en octal de votre répertoire personnel ? et celles du répertoire parent ?
- À quel groupe appartenez-vous ?
- Créez un fichier par la commande `touch liste.sh`.
- À qui appartient-il ? À quel groupe ?
- Quels sont les droits de ce fichier en `octal` ?
- Ajoutez le droit en écriture pour le groupe par la méthode octale.
- Enlevez le droit en écriture pour le groupe par la méthode symbolique.
- Le fichier `liste.sh` contiendra des commandes shell. Aussi, donnez-lui les permissions que l'on donne généralement à un programme, c'est-à-dire 755 et vérifiez-le.