NSI – 1ere	COURS Linux : gérer les droits d'accès aux fichiers	LFV

Droits d'accès en lecture, écriture ou exécution (r 4, w 2, x 1)

Les systèmes d'exploitation Linux sont multi-utilisateurs (comme Windows ou OSX). Plusieurs utilisateurs ont donc accès au système de fichiers. Or une règle de base en informatique est que l'utilisateur est a-priori dangereux (soit par malveillance soit par ignorance). On comprend alors qu'il est nécessaire d'interdire à un utilisateur quelconque d'accéder à certains fichiers :

- soit parce que ces fichiers sont nécessaires au fonctionnement du système d'exploitation,
- soit parce que ces fichiers appartiennent à d'autres utilisateurs.

Ici la notion «d'accéder à certains fichiers» mérite d'être précisée car on peut distinguer trois accès différents à un fichier :

- accéder en lecture à un fichier (uniquement le lire, r pour read),
- accéder en écriture à un fichier (le modifier ou l'effacer, w pour write),
- accéder en exécution à un fichier (lorsqu'il s'agit d'un programme, l'exécuter, x pour execute).

Il y a donc trois types de droits (lecture, écriture, exécution, respectivement identifiés par les lettres r, w et x et les valeurs 4, 2 et 1) dont la signification est résumée dans le tableau ci-dessous :

	Fichier	**Répertoire**
LECTURE r 4	Regarder le contenu	Lister le contenu
ECRITURE w 2	Modifier le contenu	Ajouter ou supprimer un élément
EXECUTION x 1	Exécuter	Passer à travers

Remarque : si un utilisateur peut lire un fichier, il peut le recopier et donc écrire ou exécuter *une copie* du fichier lu. En revanche si le fichier est dans un répertoire il lui faudra en plus les droits en lecture et exécution sur le répertoire.

Simplifier les partages : notion de groupe propriétaire

On comprend aussi qu'il est nécessaire d'autoriser les utilisateurs à partager certains fichiers : soit parce que c'est plus simple que de s'envoyer un mail avec le fichier en question, soit parce que certains fichiers (en particulier les exécutables) doivent pouvoir être exécutés par plusieurs utilisateurs sans pour autant avoir à les recopier (imaginez un peu recopier tous les programmes pour tous les utilisateurs !).

Pour gérer ces problèmes de partage, les différents utilisateurs peuvent être répartis dans différents groupes d'utilisateurs lors de la création de leur compte sur le système d'exploitation. On peut ensuite attribuer des droits d'accès identiques à tous les membres d'un même groupe. Cela permet ainsi de facilement gérer les problèmes de partage.

Ainsi chaque fichier ou répertoire possède :

- un unique utilisateur propriétaire qui a des droits rwx bien déterminés,
- un unique groupe propriétaire qui a aussi des droits rwx (souvent égaux ou plus faibles que ceux de l'utilisateur propriétaire) : tous les utilisateurs membres de ce groupe auront les droits correspondants sur le fichier ou répertoire.
- des droits rwx pour tous les autres utilisateurs (souvent égaux ou plus faibles que ceux des membres du groupe propriétaire !)

Un exemple pour comprendre

Voici comment pourraient être organisés les groupes d'utilisateurs d'un établissement scolaire (nous n'avons pas mis les 400 élèves de seconde, première et terminale ni tous les enseignants et autres personnels pour des raisons évidentes) :

groupe	seconde	premiere	terminale	delegue	eleve	enseignant	direction
ayouba	X			X	X		
briceb	X				X		
charlottec	X				X		
divined		X		X	X		
eloie		X			X		
flavief		X			X		
gildasg			X	X	X		
hamzah			X		X		
inesi			X		X		
teresat	X	X	X		X	X	
sylvains	X	X	X		X	X	
fabricef	X	X	X	X	X	X	X
chantalc	X	X	X	X	X	X	X

Pour créer un répertoire (appelons le «`dossier_delegue`») accessible uniquement aux trois délégués et aux deux proviseurs, il suffirait d'indiquer que le ***groupe*** propriétaire est le groupe «`delegue`» et d'attribuer à ce groupe tous les droits (c'est-à-dire les droits en écriture, lecture et exécution).
L'***utilisateur*** propriétaire de ce répertoire pourrait être le super-utilisateur `root` qui pourrait avoir tous les droits sur le dossier.
Enfin pour tous les ***autres*** utilisateurs, on pourrait n'accorder aucun droit : ni en lecture, ni en écriture, ni en exécution.

Finalement les droits de ce répertoire seraient paramétrés ainsi :

r w x	r w x	- - -
utilisateur	groupe	autres

Avec la commande `ls` voici ce qui s'afficherait dans le terminal (ce répertoire s'appelle «`dossier_delegue`»):

```
$ ls –l –d dossier_delegue
drwxrwx--- 2 root delegue 4096 nov.  22 14:06 dossier_delegue
```

On voit que informations associées au répertoire sont dans l'ordre :

- `d` : il s'agit d'un répertoire (`d` pour «directory», un tiret `-` indique un fichier et `l` un lien)
- `rwxrwx---` : les droits pour l'utilisateur propriétaire, le groupe propriétaires et les autres utilisateurs
- `2` : le répertoire contient 2 liens
- `root` : l'utilisateur propriétaire est root
- `delegue` : le groupe propriétaire est delegue
- `4096` : la taille du répertoire
- `nov. 22 14:06` : date et heure de création
- `dossier_delegue` : nom du répertoire

Qui est root ? Comment exécuter des commandes en tant que super-utilisateur ?

Parmi les utilisateurs, certains ont les droits d'administration sur la machine et pas d'autres. Par exemple, sur un serveur web linux, il peut n'y avoir un seul compte utilisateur ayant les droits d'administration : typiquement celui de la personne qui gère le serveur. Et sur les machines virtuelles du lycée, il s'agit de l'utilisateur nsi (mot de passe : pass2NSI).

Lorsqu'ils ont besoin d'exécuter des commandes qui requièrent les droits d'administration, les utilisateurs autorisés doivent prendre le rôle d'un utilisateur très spécial qui est le «super-utilisateur root». Avec ce rôle de super-utilisateur, on peut tout faire : installer des logiciels, consulter ou effacer les fichiers de tous les utilisateurs etc. Compte-tenu du fait du rôle très sensible du super-utilisateur, les utilisateurs qui ont les droits d'administration doivent explicitement indiquer au système d'exploitation lorsqu'ils souhaitent prendre ce rôle de super-utilisateur. Cela se fait grâce à `sudo` (*su*peruser *do*) de deux façons :

- pour exécuter une seule commande en super-utilisateur root :
  ```
  $ sudo commande [arguments]
  ```

- pour ouvrir une session de terminal en super-utilisateur root et saisir ensuite plusieurs commandes en tant que super-utilisateur root :
  ```
  $ sudo su –
  ```

Modifier les droits rwx (421) d'un fichier ou répertoire : chmod (change mode)

On modifie les droits d'un fichier (ou d'un répertoire) avec la commande `chmod` et deux arguments : le premier correspond aux nouveaux droits et le deuxième au nom de fichier.
Seuls le propriétaire d'un fichier (ou répertoire) et le super-utilisateur `root` peuvent modifier les droits d'accès (et donc les utilisateurs ayant les droits d'administration). Le changement de droits peut se faire de deux façons :

Changement absolu :
Par exemple, pour le répertoire `dossier_delegue` ci-dessus, on a :
- une valeur de 4+2+1 (`r+w+x`) pour l'utilisateur propriétaire,
- une valeur de 4+2+1 (`r+w+x`) pour le groupe propriétaire,
- une valeur de 0+0+0 (`-+-+-`) pour les autres utilisateurs.

Si on souhaite attribuer correctement les droits sur ce répertoire, il suffira d'utiliser la ligne de commande :
```
$chmod 770 dossier_delegue
```

Changement relatif :
Il s'agit de spécifier les droits à ajouter ou à supprimer relativement aux droits existants, pour telle ou telle catégorie d'utilisateurs en précisant :
- le(s) utilisateur(s) concerné(s) : **u, g, o** ou **a** pour utilisateur propriétaire (user), groupe propriétaire (group), autres utilisateurs (others : ceux ni utilisateur ni groupe propriétaire), tous les utilisateurs (all).
  En l'absence de spécification, c'est l'ensemble des utilisateurs (a) qui est concerné.
- s'il s'agit d'un ajout **(+)**, d'un retrait **(-)** ou d'une affectation **(=).**
- les droits concernés (**r**, **w** ou **x**).

Par exemple, si les droits de `dossier_delegue` étaient initialement `rw-rw-rw-`, pour les ramener à `rwxrwx---` on pourrait faire :
```
$chmod ug+x,o-rw dossier_delegue
```
ou alors :
```
$chmod u=rwx,g=rwx,o=- dossier_delegue
```
Remarque : pour l'affectation, on ne peut pas à proprement parler de changement «relatif».

Changer les droits d'un répertoire et de tout son contenu :
Dans le cas d'un répertoire on peut vouloir changer les droits du répertoire et de tout son contenu en une seule instruction. Pour cela, il suffit d'utiliser l'option `–R` :
```
$chmod –R 770 dossier_delegue
```

Changer l'utilisateur ou le groupe propriétaire (`chown` : change owner)

Seul le super-utilisateur root peut effectuer cette manipulation (et donc tous les utilisateurs ayant les droits d'administration). On utilise la commande chown d'une des trois façons suivante :

```
$chown toto nom_fichier                #proprietaire = toto
$chown toto:enfants nom_fichier        #proprietaire = toto et groupe propriétaire = enfants
$chown :enfants nom_fichier            #groupe propriétaire = enfants
```

Les inclassables à connaître malgré tout : rappels, *, `man`, > et >>

A. Dans les noms de fichiers ou de répertoires, on rappelle que :
   - ~ ou ~/ désigne le répertoire utilisateur (ne pas confondre avec /home commun à tous)
   - . ou ./ désigne le répertoire courant
   - `..` ou ../ ou ./../ désigne le répertoire parent du répertoire courant

B. * est un "joker" pour les noms de fichiers :

```
$ls
tri_insertion.py    cours_tris.pdf      tp_tris.pdf  tri_selection.py    essai.py
$ls *.py
tri_insertion.py    tri_selection.py    essai.py
$mkdir fichiers_python
$mv *.py fichiers_python #le répertoire fichiers_python contient tous les fichiers .py
```

   D'autres utilisations sont envisageables : `rm *` pour effacer tout le contenu d'un dossier par exemple …

C. man est une instruction permettant d'afficher l'aide du manuel Linux :

```
$man mv
MV(1)                          User Commands                          MV(1)
NAME
   mv - move (rename) files
SYNOPSIS
   mv [OPTION]... [-T] SOURCE DEST
   mv [OPTION]... SOURCE... DIRECTORY
   mv [OPTION]... -t DIRECTORY SOURCE...
   […]
```

D. on peut utiliser > et >> pour rediriger la sortie produite par une commande vers un fichier texte (pour sauvegarder la sortie produite par la commande) :

```
$ls –l > liste.txt        #sauvegarde du résultat dans liste.txt (écrasé si besoin)
$ls –l >> liste.txt       #rajoute le résultat de ls -l à la fin de liste.txt
```