Introduction aux systèmes d'exploitation (IS1)

TP 2 – Utilisateurs, groupes et permissions

Le but de ce TP est de vous familiariser avec le système des droits d'accès aux fichiers.

Groupes d'utilisateurs

Chaque fichier (et répertoire) est la propriété d'un utilisateur particulier. Par défaut, celui-ci appartient à l'utilisateur qui a créé le fichier. Les utilisateurs sont réunis en groupe. Un utilisateur pouvant faire partie de plusieurs groupes, pour chaque fichier est spécifié le *groupe propriétaire* (c'est-à-dire en tant que membre de quel groupe le propriétaire détient le fichier). Les droits d'un fichier sont définis en séparant les utilisateurs en trois : le propriétaire, les membres du groupe propriétaire, et les autres.

`id` : Vous connaissez déjà la commande `whoami` qui indique sous quel nom de login vous êtes connectés. La commande `id` vous indique en plus vos numéro d'utilisateur, nom et numéro de groupe principal (qui sera votre groupe propriétaire par défaut) ainsi que la liste des groupes auxquels vous appartenez.

Exercice 1 Déterminez votre ou vos groupes d'appartenance.

Droits des fichiers

Il y a trois types de droits :
- **Droit en lecture** :
  . Identifié par la lettre `r` (`read`) et le chiffre 4.
  . Donne l'accès à la lecture d'un fichier.
  . Permet de lister le contenu d'un répertoire.
- **Droit en écriture** :
  . Identifié par la lettre `w` (`write`) et le chiffre 2.
  . Donne l'accès à l'écriture dans un fichier.
  . Permet d'ajouter ou de supprimer un élément d'un répertoire.
- **Droit en exécution** :
  . Identifié par la lettre `x` (`execute`) et le chiffre 1.
  . Donne l'accès à l'exécution d'un fichier.
  . Permet de passer à travers un répertoire.

Exercice 2 Utilisez `ls -l` depuis votre répertoire personnel, et repérez les symboles décrivant les droits, le propriétaire et le groupe d'appartenance de chaque fichier/répertoire.
ATTENTION : A distinguer convenablement `l` et `1`.

`chmod` : La commande `chmod` *liste_droits fichier(s)* permet d'accorder ou de retirer au(x) fichier(s) (ou répertoire(s)) passés en arguments les permissions correspondant à *liste_droits*. Cette liste est de la forme $droit_1, \ldots, droit_n$ où chaque $droit_i$ peut être par exemple :

- `u+r` pour rajouter au propriétaire (`user`) le droit en lecture,
- `g-w` pour retirer aux membres du groupe (`group`) le droit en écriture,
- `o+x` pour donner aux autres utilisateurs (`other`) le droit en exécution,
- ou une combinaison de ces possibilités (ex : `ug-wx`).

Exercice 3 – *Un essai.*

1. A l'aide de la commande `echo "`*une phrase*`" > `*fic*, vous pouvez écrire le texte *une phrase* dans le fichier *fic*[1]. Créez un répertoire `test`, et un fichier `essai` dans ce répertoire, et écrivez-y la phrase de votre choix.

Correction.

```
mkdir test
echo "toto" > test/essai
```

2. Notez à l'aide de `ls -l` les permissions actuelles du répertoire `test` et du fichier `essai`.

Correction.

```
ls -ld test
ls -l test/essai
```

On constate que `test` est muni des permissions `rwxr-xr-x` et `essai` des permissions `rw-r--r-`.

3. En utilisant la commande `chmod`, retirez-vous le droit en lecture et en écriture sur le fichier `essai`. Vérifiez l'effet obtenu en essayant d'afficher le contenu du fichier sur la fenêtre du terminal, puis de remplacer ce contenu par une phrase différente.

Correction.

```
chmod u-rw test/essai
cat test/essai
echo "tata" > test/essai
```

Les deux dernières commandes renvoient une erreur.

4. Un fichier exécutable est simplement un fichier dont vous possédez le droit en exécution. Rétablissez le droit en écriture puis remplacez à l'aide de la commande `echo` le contenu du fichier `essai` par le texte `echo "Ceci est un essai"`. ATTENTION : Pas le texte `Ceci est un essai` seul. Tentez d'exécuter ce fichier en tapant `./essai` dans le terminal (depuis le répertoire qui le contient). Ajoutez-vous le droit en exécution, et réessayez.

Correction.

```
cd test
chmod u+w essai
echo "echo \"Ceci est un essai\"" > essai
./essai
chmod +x essai
./essai
```

La commande contenue dans le fichier `essai` n'est pas exécutée tant que le fichier n'est pas pourvu du droit en exécution. Cependant, lorsqu'il s'agit d'un script shell, le droit en exécution seul ne suffit pas.

5. Rétablissez enfin le droit en lecture et tentez à nouveau d'exécuter le fichier. Que se passe-t-il ?

1. La syntaxe précise de cette commande sera expliquée en détail dans un prochain TP.

Correction.

```
chmod u+r essai
./essai
```

La commande s'exécute enfin.

Droits des répertoires

Exercice 4 – *Un test.*

1. Placez-vous dans le répertoire `test`, et retirez-vous le droit en lecture pour ce répertoire. Listez le contenu du répertoire avec `ls`, puis exécutez ou affichez le contenu du fichier `essai`. Qu'en déduisez-vous ? Rétablissez le droit en lecture sur `test`.

   Correction. Le droit en lecture pour un répertoire est nécessaire pour pouvoir lister son contenu. Il n'est pas nécessaire pour accéder au contenu des fichiers qu'il contient (mais leur nom doit être connu, et ceux-ci doivent bien sûr être eux-mêmes accessibles en lecture pour cela).

2. 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 ?

   Correction. Le droit en écriture sur un répertoire est nécessaire pour lui ajouter un fichier ou supprimer l'un de ses fichiers. Il n'est pas nécessaire pour modifier leur contenu.

3. 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 ?

   Correction. Il est nécessaire d'avoir le droit en exécution d'un répertoire pour que celui-ci puisse être choisi comme répertoire courant. De plus, sans ce droit, toute action que l'on souhaite effectuer sur un fichier atteint en passant à travers ce répertoire échouera.

4. 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 `sstest`, de lister son contenu. Qu'en concluez-vous quant à l'influence des droits que l'on possède sur le répertoire courant ?

   Correction. Le système vérifie que l'utilisateur possède le droit d'exécution sur le répertoire courant avant de lui permettre l'accès à un fichier référencé relativement à celui-ci.

Partager ses fichiers et répertoires

Dans certains cas, par exemple dans le cadre d'un projet à plusieurs ou de la fabrication d'une page web, il peut être intéressant de donner accès à certains de ses fichiers ou répertoires à d'autres utilisateurs. Les exercices suivants explorent deux cas d'utilisation possibles.

Exercice 5 – *Donner accès à ses fichiers.*

1. Attribuez au fichier `essai` les droits suffisants pour qu'une autre personne de votre groupe UNIX puisse y accéder en lecture (mais pas en écriture).

Correction.

```
chmod g+r,g-w essai
```

2. Les fichiers des utilisateurs, et en particulier vos fichiers, sont visibles depuis tous les ordinateurs de la salle. Demandez à votre voisin(e) de tenter de lire votre fichier `essai` depuis sa machine. Pensez à lui donner un chemin complet (qui convienne).
3. Essayez d'accéder aux répertoires personnels d'autres étudiants et de consulter leurs fichiers.

Exercice 6 (Difficile) – *Ouvrir un répertoire d'accueil.*

A 2 avec votre voisin(e), créez un fichier `voisin` dans votre répertoire `test` et fixez les droits nécessaires pour que :
- seul(e) votre voisin(e) puisse écrire dans ce fichier,
- toute personne du groupe puisse lire ce fichier,
- vous seul(e) puissiez effacer ce fichier et créer de nouveaux fichiers dans `test`.

Pour ce faire, vous avez tous les deux des commandes à taper et votre voisin a besoin de droits supplémentaires provisoires au cours des manipulations.

La configuration du script ne permet pas toujours l'utilisation de la commande `ls` sur les répertoires parents des répertoires personnels. Il faut donc connaître le chemin pour accéder au répertoire de votre voisin.

Correction. Il faut d'abord donner les droits au voisin d'écrire dans votre répertoire `test` en lui donnant les droits `rwxrwx---` par exemple. À lui de créer le fichier `essai` avec les permissions `rw-r-----`. Une fois le fichier créé, il faut ensuite se réserver le droit en écriture sur `test` en redonnant les permissions `rwxr-x---` à ce répertoire (avec `chmod g-w test`).

Droits par défaut des nouveaux fichiers

Lorsque de nouveaux fichiers ou répertoires sont créés, des droits par défaut leur sont attribués. Ces droits sont calculés à partir d'un ensemble de droits par défaut en utilisant un *masque des droits par défaut des fichiers utilisateurs* (appelé *umask*).

Les droits par défaut pour un fichier sont : droit en lecture et exécution pour tout le monde (propriétaire, groupe et autres utilisateurs), et droit en écriture pour le propriétaire seulement.

On peut ensuite choisir de changer ces droits par défaut à l'aide de l'*umask*, dont l'accès se fait par la commande `umask` (en utilisant la même syntaxe que pour `chmod`). Pour visualiser le masque courant, utilisez `umask -S`. Par exemple, pour retirer aux utilisateurs ne faisant pas partie de son propre groupe l'accès en lecture aux nouveaux fichiers, on tape `umask o-r`.

Exercice 7 – *Réglage du masque*

1. 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.

   Correction.

   ```
   umask go-rwx
   ```

2. 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.

   Correction.

```
umask go=rx
```

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

Correction.

```
umask g=rx,o=
```

Droits écrits en octal

Les commandes `chmod` et `umask` peuvent également être utilisées avec une autre syntaxe : `chmod abc fic` où `a`, `b`, et `c` sont des chiffres compris entre 0 et 7. `a` représente la somme des chiffres représentants les droits du propriétaire. De même `b` représente la somme des droits du groupe propriétaire, et `c` représente celle des autres utilisateurs.
Par exemple `rwxr-xr--` correspond à 754 $((4+2+1)(4+1)(4))$.

Exercice 8 –

1. Transcrivez les commandes suivantes de la notation classique à la notation octale ou vice-versa.
- `chmod u=rx,g=wx,o=r fic`
- `chmod uo+w,g-rx fic` en sachant que les droits initiaux de `fic` sont `r--r-x---`
- `chmod 653 fic` en sachant que les droits initiaux de `fic` sont `711`
- `chmod u+x,g=w,o-r fic` en sachant que les droits initiaux de `fic` sont `r--r-x---`

Correction.

```
chmod 534 fic
chmod 670 fic
chmod u-x,g+r,o+w fic
chmod 520 fic
```

2. Remplacez la liste de commandes suivantes par une commande unique ayant le même résultat.

```
chmod 653 fic
chmod u-r,g+w,o-r fic
```

Correction.

```
chmod 273 fic
```

Les permissions particulières

Exercice 9 – *t*

1. Le répertoire `/tmp` peut être utilisé par tous les utilisateurs pour y stocker temporairement des fichiers. En général ce répertoire est vidé à chaque redémarrage de la machine. Vérifiez que vous pouvez créer et supprimer des fichiers dans ce répertoire.

2. Pouvez-vous supprimer les fichiers des autres utilisateurs ?

Correction. Non, il n'est pas possible de les renommer non plus.

3. Comment est-ce possible, possédez vous les droits en écriture sur le répertoire `/tmp` ou non ?

Correction. Le répertoire `/tmp` est muni des droits `rwxrwxrwt`, obtenu avec `chmod +t`, le "restricted flag" s'applique aux répertoires et restreint la suppression ou le renommage de fichiers qui n'appartiennent pas à l'utilisateur (`man chown`).

Exercice 10 (Dangeureux !) – *s*

1. Faites une copie de la commande `/bin/touch` dans votre répertoire maison. Renommez la en `mytouch` pour bien la distinguer de la commande fournie par le système. Vérifiez que votre copie est munie des droits nécessaires pour qu'elle puisse être exécutée. Utilisez cette commande `mytouch` pour créer un nouveau fichier vide dans votre répertoire maison.
2. Les autres utilisateurs peuvent-ils utiliser votre commande nommée `mytouch` ? Peuvent-ils créer de nouveaux fichiers vides dans leurs propres répertoires grâce à celle-ci ? Peuvent-ils avec celle-ci créer de nouveaux fichiers vides dans vos répertoires ?

   Correction. Elle fonctionne pour les autres utilisateurs exactement comme la commande `touch` fournie par le système.

3. Pour des besoins expérimentaux nous allons donner des droits particuliers à la copie `mytouch` : **n'oubliez pas d'enlever ces droits ou de supprimer cette copie rapidement après ces tests**. Utilisez la commande `chmod u+s mytouch` pour donner le droit *setuid* à la commande `mytouch`. Demandez de nouveau à votre voisin(e) d'utiliser cette commande. À qui appartiennent les fichiers crées ? Où peuvent-ils être crées ? Qu'est-ce que le droit *setuid* permet ?

   Correction. Une commande munie du bit *setuid* est exécutée avec les permissions de son propriétaire même si c'est un autre utilisateur qui l'exécute. Il existe aussi un bit *setgid* (chmod `g+s`) qui offre les permissions de son groupe d'appartenance à l'utilisateur qui l'exécute.

4. Supprimez la commande `mytouch` et imaginez ce qu'auraient pu faire vos voisins si vous aviez proposé une copie de `ls`, de `cat` ou de votre éditeur favori muni de la permission *setuid*.

   Correction. `ls` aurait permis aux autres utilisateurs de parcourir les arborescence de fichiers auxquelles vous avez accès, même si ceux-ci n'ont habituellement pas les permissions suffisantes pour le faire. `cat` aurait pu permettre aux autres utilisateurs d'accéder à vos données privées. Un éditeur muni du bit *setuid* aurait de plus permis aux autres utilisateurs de modifier vos fichiers et d'exécuter plus ou moins indirectement des commandes arbitraires avec vos droits (comme `passwd`, par exemple).

5. Certaines commandes du système qui se trouvent dans `/bin` ou `/usr/bin` possèdent des droits *setuid* ou *setgid*. Trouvez-les et devinez pourquoi elles ont été munies de ces droits.

   Correction. L'exemple typique est la commande `/bin/su` qui a besoin des droits du super-utilisateur `root` pour permettre à un nouvel utilisateur de se logger.

Exercice 11 (Optionnel) – *Limites du système de permissions*

1. Si, dans l'exercice 6, on remplaçait le fichier `voisin` par un sous-répertoire, cela permettrait à votre voisin de vous transférer des fichiers ou d'autres sous-répertoires. Testez cette idée et demandez à votre voisin(e) de créer des fichiers dans ce répertoire.
2. Essayez à présent de supprimer le répertoire `voisin` vous-même. Que remarquez-vous ? Pouvez-vous suggérer une solution ?
3. De la même manière, est-il possible de partager un fichier entre deux personnes sans le rendre accessible au reste du groupe ? Commentez.
4. Faites une recherche sur Internet au sujet du mécanisme d'ACL (*Access Control List*), et cherchez une solution aux problèmes évoqués précédemment [2].

2. Le mécanisme d'ACL n'est pas installé par défaut sur notre version de FreeBSD.