TP n°3
Gestion avancée de Fichiers sous Unix

Ce TP est destiné à vous familiariser avec les droits d'accès et les liens sous l'environnement Unix.

1 Droits d'accès

Comme nous l'avons vu précédemment, les systèmes Unix sont multi utilisateurs, ceci implique que plusieurs utilisateurs peuvent utiliser la même machine et les mêmes ressources. Pour maintenir la bonne sécurité de ces ressources les systèmes linux proposent un système de permissions sur les fichiers. À chaque fichier est associé un ensemble d'informations permettant d'identifier qui est le propriétaire du fichier mais aussi sa taille ou encore les droits d'accès qui lui sont associés. Pour consulter ces informations, on utilise la commande ls -l.

Voici un exemple de résultat d'une commande ls -l
-rwxrwxrwx 1 kernel users 3.8M jan 2 2010 /mnt/multimedia/test.mp3

Les droits d'accès du fichier */mnt/multimedia/test.mp3* sont les suivants : *-rwxrwxrwx*.

Le premier caractère indique le type de fichier.
- \- décrit un fichier ordinaire
- d décrit un répertoire
- l décrit un lien symbolique

Les caractères suivants indiquent les droits d'accès associés à ce fichier. Il existe trois types de droit d'accès :
- l'accès en **lecture** (**r**) : pour autoriser la lecture du fichier ou la visualisation du contenu d'un répertoire.
- l'accès en **écriture** (**w**) : pour autoriser à modifier un fichier ou le contenu d'un répertoire
- le droit **d'exécution** (**x**) : pour autoriser l'exécution d'un fichier. Attention le droit d'exécution x sans le droit de lecture r est autorisé mais ne vaut rien. Il faut pouvoir lire un fichier pour l'exécuter. « x » sur un répertoire rend ce dernier traversant, c'est-à-dire que l'on va pouvoir accéder aux répertoires qu'il contient.

Vous ne pouvez pas renommer, supprimer ou copier des fichiers dans un répertoire si vous n'avez pas accès en écriture à ce répertoire.

Si vous avez accès en écriture à un répertoire, vous POUVEZ supprimer un fichier même si vous ne disposez pas de droits d'écriture pour ce fichier (un répertoire est en fait juste un fichier décrivant une liste de fichiers, donc si vous avez accès en écriture sur le répertoire, vous pouvez modifier les liste des fichiers qu'il contient). Cela permet même de modifier un fichier (le supprimer et le recréer) même protégé en écriture.

Ces trois types de droits d'accès sont ensuite répartis sur trois niveaux :
- **utilisateur** (**u**) : pour le propriétaire du fichier
- **groupe** (**g**) : pour les membres du groupe associé
- **autres** (**o**) : pour tous les autres (groupes et propriétaire exclus)
-

Voici comment sont organisées les permissions des fichiers :

	owner	group	other
Permissions symboliques	r w x	- - x	- - x
Permissions binaires	1 1 1	0 0 1	0 0 1
Permissions octales	7	1	1

Dans ce cas utilisateurs, groupe et autres ont toutes les permissions sur le fichier. Voici d'autres exemples :

`-rw-r--r--`	Lisible et modifiable pour le propriétaire, seulement lisible pour les autres.
`-rw-r----`	Lisible et modifiable pour le propriétaire, seulement lisible pour les utilisateurs appartenant au groupe du fichier.
`drwx------`	Répertoire seulement accessible par son propriétaire
`-------r-x`	Fichier exécutable seulement par les autres, mais ni par votre groupe ni par vous-même. Droits d'accès typique d'un piège !

La commande **chmod** permet de modifier les droits d'accès à un fichier. Un utilisateur peut seulement modifier les droits de ses fichiers (sauf le super-utilisateur qui peut tout faire sur la machine). La commande chmod respecte la syntaxe suivante :
$ chmod permissions fichiers ou *dossier*

L'option *-R* permet d'appliquer la commande *chmod* récursivement (sur tout ce qui est contenu dans le répertoire cible y compris les sous-répertoires).

Il existe deux formats de permissions possibles : **symbolique** ou **en base 8**.

En base 2, les droits d'accès de chaque niveaux ont pour valeur 0 ou 1. Ainsi par exemple, rwxr-x--- peut être traduit en 111 101 000 en binaire, qui **en base 8** peut être converti en 750. Pour mettre de telle permission à un fichier, on écrira la commande : *chmod 750 fichier*.

En format symbolique, on décrit les permissions pour chaque niveau. Ce format est plus simple à comprendre avec des exemples. En voici quelques uns :

- chmod a+rw : on ajoute le droit pour tous de lire et d'écrire
- chmod u+x : on ajoute pour l'utilisateur le droit d'exécuter
- chmod g-w : on retire pour le groupe le droit d'écrire
- chmod o-rwx : on retire pour les autres le droit de lire, écrire et exécuter

2 Liens physiques et liens symboliques

Un fichier stocké sur le disque dur est un ensemble de données qui sont enregistrées. Il faut ensuite un moyen d'accéder à ces données. C'est le rôle que joue alors le nom du fichier qui permet d'indiquer à quelles données on fait référence.

2.1 Lien physique : ln

Lors de la création d'un fichier, on lui associe traditionnellement un seul nom. Mais en fait il peut y en avoir plusieurs. Chacun de ces noms de fichier est appelé lien physique vers celui-ci. Il faut voir cela comme un point d'accès vers les données se trouvant dans l'arborescence.

Lors de l'utilisation de la commande *ls* avec l'option -l, on peut voir le nombre de ces liens physiques.
$ touch find
$ ls -l find

-rwxr-xr-- 1 user user 0 sept. 12 2013 find

C'est la deuxième colonne qui l'indique. On l'appelle compteur de référence. Ici on sait donc qu'il y a une seule référence existant vers le contenu de ce fichier, celui-ci s'appelant *find*. C'est le cas le plus courant. On peut rajouter un lien physique à l'aide de la commande « *ln* ». On lui passe en paramètre un des liens physiques déjà existant suivi par le nom du nouveau lien à créer. On pourrait par exemple ajouter un lien physique du nom de search. Cet exemple montre aussi le résultat ensuite.

$ ln find search (il faut être super utilisateur pour avoir l'autorisation dans ce cas : ajouter *sudo* devant cette commande pour avoir les droits du super utilisateur, nous verrons cela un peu plus tard)

$ ls -l find

-rwxr-xr-- 2 user user 0 sept. 12 2013 find

$ ls -l search

-rwxr-xr-- 2 user user 0 sept. 12 2013 search

Le nombre de liens physiques est alors passé à 2. Il faut bien voir que tous les liens physiques sont strictement équivalents. Lors de la suppression d'un de ces liens (à l'aide de la commande *rm*), le compteur de référence est décrémenté.

$ rm find
$ ls -l

-rwxr-xr-- 1 user user 0 sept. 12 2013 search

S'il est différent de 0, rien n'est fait au niveau du fichier. La suppression du fichier ne sera effective que lorsque le dernier lien physique vers celui-ci sera supprimé (compteur = 0). Étant donné que généralement on ne crée des fichiers qu'avec un seul lien physique, la suppression de celui-ci avec rm est équivalente à la suppression du fichier.

2.2 Lien symbolique : ln -s

Il existe un autre type de lien, les liens symboliques. Ce sont eux qui s'approchent le plus de la notion intuitive de lien.

Un lien symbolique est en fait un type de fichier spécial qui contient le chemin vers un fichier du disque (en réalité un de ses liens physiques). On les crée aussi avec la commande *ln* mais en utilisant l'option -s (c'est ce qui se rapproche des *raccourcis sous Windows*). Voici un exemple pour illustrer cela. On suppose que le lien physique créé précédemment a été supprimé.

$ ln -s search cherche
$ ls -l search

-rwxr-xr-- 1 user user 0 sept. 12 2013

search $ ls -l cherche

lrwxrwxrwx 1 Tian users 6 sept. 21 2013 cherche -> search

La première chose à observer est le fait que le compteur de référence de *search* n'a pas été modifié. Un fichier n'a absolument pas connaissance du nombre de liens symboliques pointant vers lui.

Dans les permissions du lien appelé *cherche*, on peut voir tout d'abord la lettre l indiquant qu'il s'agit bien d'un type particulier de fichier, un lien symbolique. On remarquera également que toutes les permissions sont

présentes. Ceci car ce sont en fait celles du fichier destination qui seront utilisées pour vérifier les autorisations d'accès.

Et enfin on a l'indication de la cible du lien symbolique. Lors de la création, on aurait pu utiliser une indication de chemin. Celle-ci aurait été conservée telle quelle dans le lien symbolique y compris si le chemin était donné de manière relative.

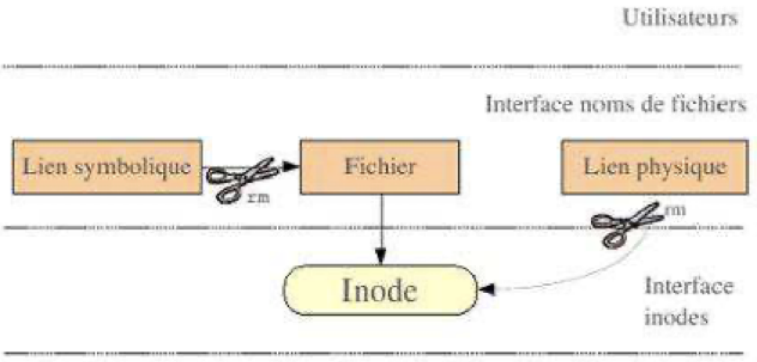
L'utilisation ensuite du lien symbolique sera équivalente à celle du fichier cible pour les commandes l'utilisant.

Si ce dernier point fait apparaître les deux types de liens comme très proches, il reste des différences.

Le lien symbolique est totalement indépendant du fichier lui-même et aussi du lien physique auquel il fait référence. On peut créer un lien symbolique en indiquant un chemin de fichier n'existant pas. Le fichier peut aussi être supprimé ensuite sans que le lien symbolique n'en soit informé. Toutefois les accès futurs au fichier au travers de ce lien renverront bien sûr une erreur. Et enfin la suppression d'un lien symbolique (à l'aide également de la commande rm) n'aura aucune conséquence sur le fichier.

2.3 I-nœuds

Un système UNIX n'identifie pas un fichier par son nom. En effet, ce n'est pas commode à manipuler et comme on l'a vu avec les liens, deux fichiers de noms différents à priori peuvent correspondre au même « bloc mémoire ». Dans un système UNIX, un fichier quel que soit son type, est en fait identifié par un numéro appelé numéro d'i-nœud (« inode » en anglais). Le lien entre le numéro d'i-nœud attribué par le système et le nom attribué par l'utilisateur se situe en réalité dans le contenu du répertoire dans lequel « se trouve » le fichier.



3 Utilisation avancée de l'interprète de commande

3.1 pushd et popd

La commande *cd* - permet de revenir dans le répertoire précédent. Mais il existe aussi deux commandes pour contourner la commande « *cd* » : *pushd* et *popd*.

pushd s'utilise exactement comme *cd* pour se rendre dans un autre répertoire. La différence est que le répertoire quitté est sauvegardé dans une pile. Et, à chaque changement de répertoire, est ajouté en haut de la pile un nouveau répertoire.

popd permet de passer dans le dernier répertoire enregistré par *pushd*. Celui-ci est alors supprimé de la pile et un nouvel appel à popd permet d'aller dans le répertoire qui était visité avant celui dans lequel vous vous trouvez.

Ce comportement est le même que celui utilisé par la plupart des navigateurs internet qui possèdent un bouton ou une commande pour retourner à la page précédente. C'est ce rôle que joue *popd*.

3.2 Alias

Pour ceux qui sont habitués à l'utilisation de cd pour changer de répertoire mais souhaiteraient bénéficier de ce comportement, il suffit de définir un alias :

(Un alias est une primitive de nombreux shells informatiques qui permet d'afficher ou d'initialiser les substitutions de noms de commandes. Elle est majoritairement employée pour abréger une commande ou rajouter par défaut des options à une commande régulièrement utilisée.)

$ alias cd="pushd"

Un changement de répertoire en utilisant *cd* sauvegardera alors le répertoire quitté. *popd* permettra ensuite de remonter l'historique des répertoires visités.

unalias supprime de façon définitive les alias créés avec la commande *alias* si jamais vous avez des alias dans le fichier d'initialisation de shell (*.bashrc* par exemple, alors il faut supprimer la ligne correspondant à l'alias pour supprimer définitif l'alias sinon l'alias ne sera supprimé que pour la session en cours).

1. J'affiche les alias (ce que j'ai dans *.bashrc*) :
   $ alias
   alias dem='/bin/ls'
   alias vi='/usr/bin/vim'

2. Je crée un alias en ligne de commande :
   $ alias ll='ls -l'

3. J'affiche les alias (.bashrc + le nouveau créé) :
   $ alias
   alias dem='/bin/ls'
   alias ll='ls -l'
   alias vi='/usr/bin/vim'

4. J'affiche les alias se trouvant dans *.bashrc :*
   $ grep alias .bashrc
   # User specific aliases and functions
   alias dem='/bin/ls'
   alias vi='/usr/bin/vim'

5. Je supprime les alias :
   $ unalias {ll,vi}

6. J'affiche les alias après la suppression, vi et ll ne sont plus affichés :
   $ alias
   alias dem='/bin/ls'

7. Je simule le redémarrage de la session :
   $ source .bashrc

8. J'affiche les alias :
   $ alias
   alias dem='/bin/ls'
   alias vi='/usr/bin/vim'

On voit bien que l'alias vi existe toujours.

Exercices

1 Permissions sur les fichiers

Exercice n°1:

1. Créez un répertoire *Linux* et déplacez-vous dans celui-ci
2. Créez le fichier vide *mon_fichier*, et examinez ensuite ses permissions.
3. Donnez-lui successivement les droits nécessaires pour que vous puissiez.
    1. Lire, modifier et exécuter votre fichier.
    2. Lire, modifier mais pas exécuter votre fichier.
    3. Lire mais pas modifier ou exécuter votre fichier.
4. Accordez maintenant toutes les permissions au propriétaire et la lecture seulement pour le groupe.
5. Positionnez les permissions nécessaires pour qu'un utilisateur de votre groupe puisse lire, modifier mais ne pas supprimer votre fichier.

2 Liens physiques et symboliques

Exercice n°2:

1. Créez dans votre répertoire ~ un répertoire *tmp* qui contient un fichier *bidon*. A l'aide de *gedit*, ajoutez une ligne de texte dans le fichier *bidon*.
2. Dans votre home directory (~), créez un lien physique appelé *dhuile* vers le fichier *tmp/bidon*. Comparez les contenus de *tmp/bidon* et de *~/dhuile*. Que contient *dhuile* ?
3. Notez les droits que vous avez actuellement sur le fichier *~/dhuile*. Modifiez les droits sur le fichier *tmp/bidon* pour avoir les permissions suivantes *rw-r-----*. Quels sont les droits d'accès sur le fichier *~/dhuile*.
4. Supprimez le fichier *tmp/bidon* puis consultez le contenu du fichier *dhuile*. Que constatez-vous ?
5. Après avoir effacé le fichier *dhuile*, refaites les questions 1, 2 et 3 de cet exercice, mais au lieu de faire un lien physique, faites un lien symbolique.
6. Quelles sont les différences entre les liens physiques et les liens symboliques ?
7. Faites un lien physique dans votre *home directory* avec le nom *cherche* sur le fichier */usr/bin/find*. Que se passe-t-il ? En déduire dans quel cas on ne peut pas faire de lien physique ? Que faut-il faire alors ?

3 Alias

Exercice n°3:

Faire un alias qui permet de voir les fichiers cachés et d'afficher les fichiers en couleur.