S. Lavirotte, D. Lopez, H. Renard
Polytech'Nice – Sophia
PeiP
2019-2020

TD séances n° 5
Gestion avancée de Fichiers et Processus sous Unix

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

1 Droits d'accès

Les systèmes Unix sont multi-utilisateurs, ceci implique que plusieurs utilisateurs peuvent utiliser la même machine et les mêmes ressources. Pour maintenir une bonne sécurité de ces ressources, les systèmes Unix 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`.

1.1 Type de fichier

Voici un exemple de résultat d'une commande :

```
$ ls -l
-rwxrwxrwx 1 kernel users 3.8M jan 2 2016 /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 (fichier ordinaire, répertoire ou lien que nous étudierons juste après) :

- `-` : décrit un fichier ordinaire
- `d` : décrit un répertoire
- `l` : décrit un lien symbolique

1.2 Accès en lecture, écriture, exécution pour l'utilisateur, groupe ou autres

Les caractères suivants indiquent les droits d'accès associés à ce fichier ou à un répertoire. Mais il y a de légères différences d'interprétation des droits en fonction du fait que ces droits s'appliquent à un répertoire ou à un fichier.

Il existe trois types de droit d'accès :

- L'accès en **lecture** (`r`) : pour autoriser la lecture du contenu d'un fichier ou la visualisation du contenu d'un répertoire.
- L'accès en **écriture** (`w`) : pour autoriser à modifier le contenu d'un fichier ou le contenu d'un répertoire. Mais qu'est-ce que le contenu d'un répertoire ? Et bien c'est la liste des fichiers ou des sous-répertoires qu'il contient.
- 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 le contenu d'un fichier pour l'exécuter. Le droit d'exécution « `x` » sur un répertoire rend ce dernier traversant, c'est-à-dire que l'on va pouvoir s'y déplacer (y faire une commande `cd` pour s'y rendre)

Pour ajouter, renommer ou supprimer un fichier (ou un sous-répertoire) d'un répertoire, il faut le droit d'écriture sur le répertoire qui contient le fichier (ou le sous-répertoire) et par sur le fichier lui-même. Le droit d'écriture sur le fichier permet de modifier son contenu. Le droit d'écriture sur le répertoire qui contient un fichier permet de modifier son nom (voir de le créer ou de le supprimer).

Ces trois types de droits d'accès sont ensuite répartis sur trois niveaux :

- **utilisateur** (`u` : user) : pour le propriétaire du fichier (celui à qui appartient le fichier n'est pas obligatoirement celui qui l'a créé)
- **groupe** (`g` : group) : pour les membres du groupe associé
- **autres** (`o` : others) : pour tous les autres (groupe et propriétaire exclus)

TD séances n° 5
Gestion avancée de Fichiers et Processus sous Unix

Voici comment sont organisées les permissions :

	utilisateur	groupe	autres
Permissions symboliques	`r w x`	`- w x`	`- - x`
Permissions binaires	`1 1 1`	`0 1 1`	`0 0 1`
Permissions octales	`7`	`3`	`1`

Le calcul des permissions en octal se fait de la manière suivante : « `r` » vaut 4, « `w` » vaut 2 et « `x` » vaut 1.

Voici d'autres exemples :

`-rw-r--r--` Lisible et modifiable pour le propriétaire, seulement lisible pour les autres (équivalent octal : 644)

`-rw-r----` Lisible et modifiable pour le propriétaire, seulement lisible pour les utilisateurs appartenant au groupe du fichier (équivalent octal : 640)

`drwx------` Répertoire seulement accessible par son propriétaire (équivalent octal : 700)

`-------r-x` Fichier exécutable seulement par les autres, mais ni par votre groupe ni par vous-même (équivalent octal : 005)

1.3 `chmod` : la commande de modification des permissions

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

```
$ chmod permissions fichiers_ou_répertoires
```

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 et leur contenu).

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

1.3.1 Permission sous forme numérique

En base 2 (binaire), les droits d'accès de chaque niveau 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 telles permissions à un fichier, on écrira la commande :

```
$ chmod 750 fichier
```

1.3.2 Permission sous forme symbolique

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

Il faut distinguer deux choses dans le cas d'un fichier : le contenu du fichier et le nom du fichier. Le contenu est un espace sur le disque dur qui permet de stocker les données (par exemple, le texte du compte rendu que vous tapez

en ce moment). Le nom du fichier est le moyen d'identifier et de retrouver le contenu d'un fichier (donc l'information stockée dans le fichier). Le nom d'un fichier fait donc référence à un contenu.

2.1 Lien physique : `ln`

Lors de la création d'un fichier, on lui associe traditionnellement un seul nom. Mais en fait, il peut 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 liens physiques.

```
$ touch find
$ ls -l find
-rwxr-xr-- 1 user user 0 sept. 12 2016 find
```

C'est la deuxième colonne qui l'indique (en gras sur la ligne ci-dessus). 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
$ ls -l find
-rwxr-xr-- 2 user user 0 sept. 12 2016 find
$ ls -l search
-rwxr-xr-- 2 user user 0 sept. 12 2016 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 2016 search
```

S'il n'est pas nul, il y a toujours au moins un nom permettant d'accéder au contenu. En fait, quand on supprime un fichier à l'aide de la commande `rm`, on supprime un nom qui permet d'accéder au contenu. Et quand on a supprimé tous les noms qui permettaient d'accéder à un contenu, on ne peut plus y accéder. Mais on ne supprime pas vraiment le contenu pour autant sur le disque dur (donc un informaticien avec les bons outils pourra récupérer tout ou partie du contenu même si vous croyez avoir supprimé le fichier… Avis aux fraudeurs…).

2.2 Lien symbolique : `ln -s`

Il existe un autre type de lien, les liens symboliques (cette notion est à rapprocher du raccourci sous Windows, même si dans leur mise en œuvre, ce n'est pas complètement identique). Un lien symbolique est en fait un type de fichier spécial qui contient le chemin vers un nom de fichier (donc vers un de ses liens physiques). On les crée aussi avec la commande `ln` mais en utilisant l'option `-s`.

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 2016 search
$ ls -l cherche
lrwxrwxrwx 1 Tian users 6 sept. 21 2016 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. Ce sont en fait celles du fichier destination qui seront utilisées pour vérifier les autorisations d'accès.

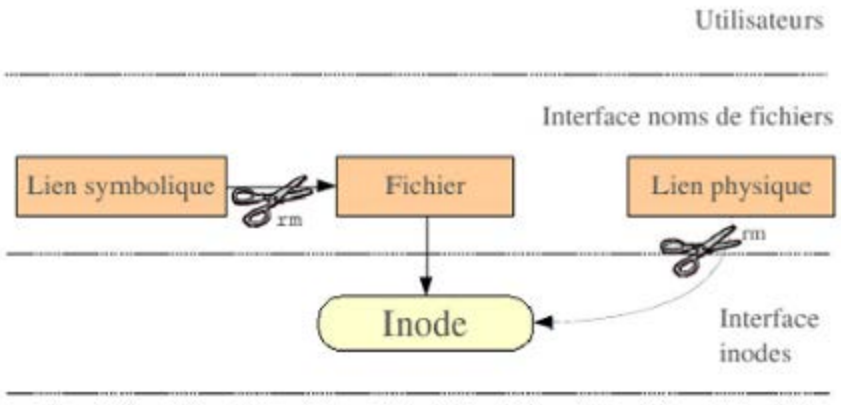
Et enfin on a l'indication de la cible du lien symbolique. Lors de la création, on aurait pu utiliser une indication de chemin. Celle-ci aurait été conservée telle quelle dans le lien symbolique y compris si le chemin était donné de manière relative.

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 qui n'existe 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 Pour aller plus loin : les i-nœuds (ou *i-node*)

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 Gestion des Processus

Quand vous interagissez avec l'interprète de commande, vous avez dû remarquer que vous devez attendre que l'exécution de la commande précédente soit terminée pour avoir à nouveau l'affichage du prompt et ainsi pouvoir taper une nouvelle commande. Comme les commandes que vous avez utilisées jusqu'à présent sont très rapides à exécuter, vous ne vous en êtes peut-être pas rendu compte. Pour le constater, lancez par exemple la commande `gedit`. Tant que l'application n'est pas terminée (que vous n'avez pas fermé la fenêtre), vous ne pouvez plus taper de nouvelles commandes. Il est possible de lancer une commande en arrière-plan c'est-à-dire qu'elle poursuive son exécution sans bloquer l'interprète de commandes. Pour réaliser cette opération vous utiliserez le symbole « & » à

la fin de la commande. Ceci indique à l'interprète de commande qu'il doit lancer la commande et ne pas attendre sa fin.

```
$ gedit &
$
```

Quand une commande a été lancée en oubliant le symbole « & » pour dire de lancer cette commande en arrière-plan (sans bloquer l'interprète de commandes), il est possible de suspendre l'exécution de cette commande. Pour cela, vous devez utiliser la combinaison de touches « Ctrl+Z » dans le terminal. Essayez de taper des caractères ou de cliquer sur un bouton de l'interface graphique de `gedit`. Si vous n'y arrivez pas c'est normal, l'application a été suspendue (son exécution a été gelée). Vous pouvez aussi constater que vous avez à nouveau le prompt qui s'affiche dans le terminal et que vous pouvez taper une nouvelle commande.

Pour reprendre l'exécution de l'application suspendue vous avez deux commandes : soit la commande `fg`, soit la commande `bg`. La commande `fg` (comme *foreground* en anglais) relancera l'exécution de la commande suspendue et rebloquera le terminal jusqu'à la fin de l'application (mais cela vous permettra de taper d'autres commandes entre temps). La commande `bg` (comme *background* en anglais) reprendra l'exécution de la commande, mais en arrière-plan (comme si vous aviez la commande initiale avec le symbole « & » en fin de commande.

Enfin, quand une commande bloque votre terminal, vous pouvez aussi utiliser la combinaison de touches Ctrl+C qui vous permet d'arrêter une commande en cours d'exécution.

La commande `jobs` vous permet de lister les processus que vous avez lancés depuis votre interprète de commandes.

L'exécution d'un programme sur un ordinateur s'appelle un processus.

S. Lavirotte, D. Lopez, H. Renard
Polytech'Nice – Sophia
PeiP
2019-2020

TD séances n° 5
Gestion avancée de Fichiers et Processus sous Unix

Exercices

1 Permissions sur les fichiers

Exercice n°1:

1. Dans votre dossier personnel, créez un répertoire `Linux` et déplacez-vous dans celui-ci.
2. Créez le fichier vide `mon_fichier`, et examinez ensuite ses permissions.
3. Pour chacun des exercices suivants, donnez la commande `chmod` correspondante avec le changement de permissions en symbolique et en numérique. Donnez successivement au fichier les droits nécessaires pour que vous puissiez :
    a. Lire, modifier et exécuter votre fichier
    b. Lire, modifier mais ne pas exécuter votre fichier
    c. Lire mais ne pas modifier ou exécuter votre fichier
4. Accordez maintenant toutes les permissions au propriétaire et la lecture seulement pour le groupe et rien pour les autres.
5. Maintenant tentez de consulter le fichier `mon_fichier` créé par votre voisin et testez ce que vous pouvez faire sur ce fichier. Expliquez.
6. Positionnez les permissions nécessaires pour qu'un utilisateur de votre groupe puisse lire, modifier mais ne pas supprimer votre fichier. Que faut-il faire pour pouvoir supprimer le fichier ?

2 Liens physiques et symboliques

Exercice n°2:

1. Créez dans votre dossier personnel 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 dossier personnel (~), 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 de nom `cherche` dans `/tmp` 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 Gestion des processus

Exercice n°3:

1. Lancez la commande `xeyes`. Que constatez-vous sur l'interprète de commandes ?
2. Quelle action faire pour récupérer la main dans l'interprète de commandes. Que constatez-vous pour l'application `xeyes` ?
3. Quelle commande utiliser pour relancer l'exécution de `xeyes` (pas lancer un nouveau programme, mais continuer l'exécution de celui qui était lancé) en perdant la main à nouveau dans le terminal ?

4. Quelle action faire pour terminer l'exécution de `xeyes` depuis le terminal (sans fermer directement la fenêtre) ?
5. Lancez à nouveau la commande `xeyes` puis suspendez son exécution.
6. Relancer l'exécution de `xeyes`, mais cette fois en récupérant l'invité de commande (l'affichage du prompt) dans le terminal.
7. Quelle commande utiliser pour lancer la commande xeyes en récupérant tout de suite la main dans le terminal ?

S. Lavirotte, D. Lopez, H. Renard
Polytech'Nice – Sophia
PeiP
2019-2020

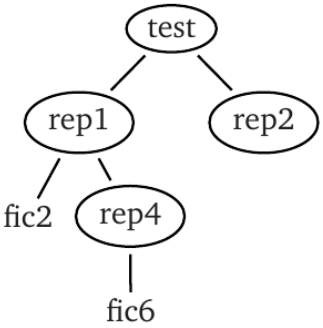
TD séances n° 5
Gestion avancée de Fichiers et Processus sous Unix

Exercices Complémentaires

Nous allons explorer un peu plus en détail les possibilités du système de fichier UNIX à travers le concept de lien et la notion de numéro d'i-nœud.

Exercice A :

- Pour commencer, créez l'arborescence suivante dans votre dossier personnel.



- Donnez trois façons de désigner le fichier `fic6` depuis votre dossier personnel.
- À l'aide d'un éditeur de texte, écrivez *Il fait beau aujourd'hui !* dans le fichier *fic6*.
- À l'aide de la commande `cat`, affichez le contenu du fichier `fic6` depuis votre répertoire `rep2`.
- L'option `-l` de la commande `ls` permet entre d'autre d'observer les droits d'un fichier/répertoire. À quoi correspondent les autres informations que l'on obtient grâce à cette commande ?

Exercice B :

- Comme expliqué dans le cours, la commande « `ln` » sert à créer des *liens*. Utilisez-la pour créer un lien **physique** du fichier `fic6` dans *test* sous le nom de `lpfic6`.
- Modifiez le contenu du fichier `lpfic6`. Que constatez-vous pour le fichier `fic6` ? Réciproquement, modifiez `fic6`, lisez `lpfic6`. Concluez.
- Modifiez les droits d'accès au fichier `fic6` pour les membres du groupe. Que constatez-vous pour le fichier `lpfic6` ? Pouvez-vous avancer une explication ?
- La commande « `ln` » peut aussi créer des liens *symboliques* avec l'option `-s`. Créez un lien symbolique du fichier `fic6` dans `test` que vous appelez `lsfic6`.
- Regardez toutes les informations concernant les fichiers `lpfic6` et `lsfic6`. Quelles différences notez-vous ?
- Essayez de modifiez les droits d'accès au fichier `lsfic6`. Pour les mettre à tous les droits pour les utilisateurs et le groupe, mais rien pour les autres. Que constatez-vous ?
- Modifiez les droits d'accès au répertoire `rep1` pour ne plus y avoir accès. Essayez d'afficher le contenu de `lpfic6` et `lsfic6`. Que constatez-vous ? Pouvez-vous avancer une explication ?
- Modifiez de nouveau les droits d'accès au répertoire `rep1` pour y avoir de nouveau accès. Déplacez le fichier `fic6` dans le répertoire `rep1`. Essayez d'afficher le contenu de `lpfic6` et `lsfic6`. Que constatez-vous ?
- Redéplacez le fichier `fic6` dans le répertoire `rep4`. Essayez à nouveau d'afficher les contenus de `lpfic6` et `lsfic6`. Supprimez le fichier `fic6` puis recommencez. Que constatez-vous ?

Pour la suite de la synthèse d'exercices, recréez le fichier `fic6` *dans le répertoire* `rep4`.

TD séances n° 5
Gestion avancée de Fichiers et Processus sous Unix

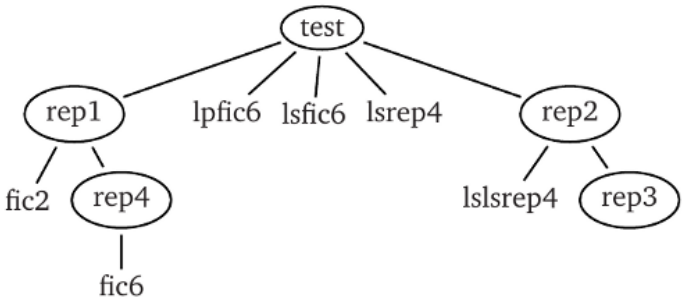
Exercice C :

- Observons maintenant plus attentivement le répertoire `rep2`. Combien y a-t-il de liens sur ce répertoire ? À quoi correspondent-ils ?
- Dans le répertoire `rep2`, créez un sous-répertoire `rep3`. Combien y a-t-il maintenant de liens sur le répertoire `rep2` ? Expliquez.
- Créez un lien physique `lprep4` du répertoire `rep4` dans le répertoire `test`. Que remarquez-vous ?

Avec les implémentations existantes actuellement, seul le super-utilisateur peut créer un lien matériel sur un répertoire, et encore, ce n'est pas toujours possible. Cependant, on peut tout à fait créer un lien symbolique sur un répertoire.

- Créez un lien symbolique `lsrep4` du répertoire `rep4` dans le répertoire `test`.
- Créez un lien symbolique `lslsrep4` du lien `lsrep4` dans le répertoire `rep2`.
- Donnez trois manières différentes de se déplacer dans le répertoire `rep4` à partir du répertoire `test`.
- En utilisant successivement ces trois méthodes, déplacez-vous dans le répertoire `rep4` puis remonter dans le répertoire parent à l'aide de la commande `cd ..`. Que remarquez-vous ?
- Que se passe-t-il si on utilise la commande `ls -Ral` sur `lsrep4` ? et sur `lslsrep4` ?

À la fin de cet exercice, nous avons maintenant l'arborescence suivante :



Exercice D :

- À l'aide de la commande `ls` munie de l'option appropriée, observez le numéro d'i-nœud du fichier `fic2`. Copiez le fichier `fic2` dans le répertoire `rep3`. Quel est son numéro d'i-nœud ?
- Changez le nom de ce dernier fichier, pour l'appeler `fic6`. Le numéro d'i-nœud change-t-il ?
- Comparez les numéros d'i-nœuds entre le fichier `fic6` du répertoire `rep4`, `lpfic6` et `lsfic6`. Que remarquez-vous ? Expliquez maintenant plus clairement les dernières questions de l'exercice B.
- Observez maintenant le numéro d'i-nœud de la racine et de votre répertoire personnel. Que remarquez-vous ?