# TP : Liens, redirections, montages

## Liens physiques et symboliques

TP réalisé d'après un travail similaire de Nicolas Pécheux. Un compte-rendu sur feuille est attendu. Si une question ne comporte pas de phrase interrogative, donner la ou les commandes qui permettent de réaliser l'instruction.

**Exercice 1.** Cet exercice est à réaliser sasn ouvrir aucun éditeur de texte (coùùe `vi, emacs, gedit, nano...`.

1. Créer un répertoire `MyRep` et s'y rendre.
2. Avec `echo` et des redirections écrire deux lignes dans un fichier `fich1.txt` :

   ```
   $ cat fich1.txt
   toto et gogo
   tata et tutu
   ```

   Rendez-vous dans `MyRep`
3. Afficher le numéro d'inode de de `fich1.txt`. Copier `fich1.txt` dans le même répertoire sous le nom `fich2.txt`. Comparer son inode et celui de `fich1.txt`.
4. Déplacer `fich2.txt` en `fich3.txt` dans `MyRep`. Que dire de l'inode de `fich3.txt` ?
5. Modifier avec votre éditeur de texte, le contenu de `fich3.txt`. Est-ce que le contenu de `fich1.txt` a été modifié ?

La commande `ln` fonctionne de la même manière que les commandes `mv` et `cp` mais, plutôt que de déplacer ou copier le fichier, elle se contente de créer un nouveau lien vers ce fichier, appelé *lien physique*. On obtient donc le même fichier, mais avec un nouveau nom (en fait, il s'agit d'un alias). En réalité, le« nom d'un fichier » est stocké dans le répertoire parent. Plus d'un répertoire peut se référer à un même fichier, un même répertoire peut se référer à un même fichier plusieurs fois, et chaque référence peut avoir un nom différent. Lorsque l'on dit qu'un répertoire « contient » un fichier, ce n'est en réalité qu'une illusion (bien pratique).

On reste dans `MyRep`.

5. Exécuter `ln fich3.txt fich4.txt`.
6. Afficher juste les inodes des fichiers. Que constatez-vous ?
7. Afficher les inodes et les droits des fichiers. Combien de liens vers `fich1.txt`,`fich3.txt` et `fich4.txt` (le nombre de liens vers un fichier d'inode donné apparaît après les droits) ?
8. Modifier le contenu de `fich3.txt` en ajoutant « titi » à la fin par redirection.
   Que dire du contenu de `fich4.txt` ?

9. Supprimer `fich3.txt`. Quel est le nombre de liens vers `fich4.txt` ?
10. Les liens physiques ne peuvent cependant pas aller d'une partition à une autre, d'un système de fichiers à un autre. Pourquoi à votre avis ?
11. Quel est le nombre de liens vers `MyRep` ? Pourquoi y a-t-il toujours au moins deux liens vers un répertoire.

On ne peut pas, en général, créer de liens physiques pour les répertoires, ceci pourrait causer des problèmes de cycles et des cas dégénérés, mais on dispose d'un autre outil : les *liens symboliques*.

Les liens symboliques sont simplement des noms de chemins qui redirigent vers le fichier ou répertoire ainsi pointé. On peut créer un lien avec l'option `-s` (pour symbolic) et on utilise toujours `rm` pour la suppression.

12. Créer un répertoire `MyRep2`, frère de `MyRep`.
13. Placez vous dans `MyRep2`. Creez un lien symbolique de nom `fich5.txt` vers `fich1.tex`
14. Regardez les droits et l'inode de `fich5.txt`, les droits et l'inode de `fich1.txt`. Le nombre de lien symbolique apparaît-il ?
15. Comparez avec `cat`, les contenus de `fich1.txt` et `fich5.txt`.
16. Est-ce que les modifications appliquées à `fich5.txt` impactent `fich1.txt` (testez) ? Et l'inverse (testez) ?
17. Comparer la taille des fichiers `fich1.txt` et `fich5.txt`. Que constate-t-on ?
18. Déplacer le fichier `fich1.txt` dans `MyRep2` puis lister le contenu `fich5.txt`.
    Que se passe-t-il ?
19. Supprimer le lien `fich5.txt`.
20. Revenir au père de `Myrep2`. Entrer `ln -s MyRep2/fich1.txt MyRep/fich6.txt`. Vérifier par `ls` que `fich6.txt` est bien créé.
21. Lister le contenu de `fich6.txt`. Expliquer le comportement observé.

# Redirections

**Exercice 2.** Creer un fichier `toto` contenant

```
abcdef 123
titi et tutu
toto et gogo
```

1. Que fait la commande `tr` ?
2. Par `echo` et un pipe, envoyer la chaîne `"abcdef"` à `tr 'a-c' 'A-C'` et donner le résultat.
3. Comment envoyer par redirection le contenu de `toto` à `tr` de sorte que les minuscules deviennent toutes des majuscules ?

   ABCDEF 123
   TITI ET TUTU
   TOTO ET GOGO

4. Dans le répertoire courant, lister les droits de `toto` et envoyer la sortie vers le fichier `listing` créé pour l'occasion. Puis lister les droits du fichier `tata` (qui n'existe pas) et envoyer la sortie et les erreurs éventuelles vers `listing` sans écrasement.
   On doit obtenir

```
$ cat listing
toto
ls: impossible d'accéder à 'tata': Aucun fichier ou dossier de ce type
```

## Montage d'un système de fichier

**Exercice 3.** La notion de *point de montage* apparaît surtout dans les systèmes UNIX : en effet, dans un système WINDOWS, les périphériques de stockage de données et les partitions sont affichés comme des lecteurs indépendants en haut de leur propre arborescence. Sous UNIX, en revanche, ils sont inclus dans l'arborescence, car UNIX traite aussi les partitions et périphériques de stockage comme des fichiers.

Un *point de montage* est un répertoire (en général vide) du système de fichier courant sur lequel un système de fichier supplémentaire est monté (ou encore *attaché*). Le point de montage $M$ d'un nouveau système de fichier $F_2$ est utilisé comme la racine de $F_2$. Ainsi, les fichiers de $F_2$ sont accessibles depuis $M$. Le contenu de $M$ disparaît du système de fichier courant tant que dure le montage de $F_2$ ; il réapparaît ensuite.

Sous UBUNTU, cette notion de point de montage échappe à l'utilisateur peu expérimenté car le montage et le démontage (de clé USB, de CD-ROM etc.) sont souvent automatiques.

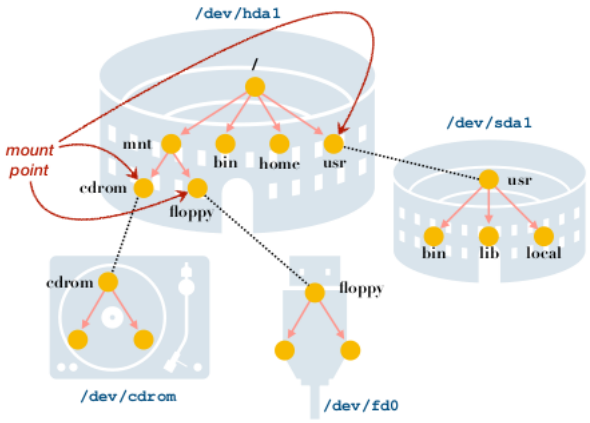


FIGURE 1 – Sur cette image, le disque `/dev/hda1` est monté à la racine, le disque `/dev/sda1` est monté sur le répertoire `/usr`, le lecteur de CDROM est monté sur le répertoire `/mnt/cdrom` et la clé USB est montée sur `/mnt/floppy` (Nicolas Pécheux).

La commande `mount` donne beaucoup d'informations sur les systèmes de fichiers montés à un instant donné.

1. Entrer `mount` dans un terminal. Chercher la partition racine (elle est montée sur `/`) et la partition de boot (montée sur `/boot/efi` sans doute). C'est fastidieux car la commande renvoie beaucoup de lignes.
2. On va envoyer la sortie de `mount` sur l'entrée de la commande `grep` pour trouver plus facilement les informations demandées.
   (a) Chercher les informations sur la racine de LINUX (motif à trouver : « on / »). Quel est le nom du système de fichier (par exemple `/dev/sda1`) Quel est le type de ce système de fichiers (par exemple `ext2`) ?
   (b) Même question avec la partition de démarrage.
3. Vérifier que vous n'avez pas de clé USB branchée sur votre machine. Est-ce que le mot « media » apparaît quelque part dans le retour de `mount` ?
4. Brancher une clé USB et rechercher le mot « media ». Sur UBUNTU les clés sont montées sur le répertoire `/media`. Normalement vous devriez trouver quelque chose comme « `on /media/<votre nom>/<nom de la clé>` ». Donner le nom du système de fichier associé à cette clé ? Quel est son type ?
5. On démonte la clé USB (en fait, le système de fichier associé à cette clé) avec la commande `sudo umount -f /media/<votre nom>/<nom de clé>`. Vérifier l'effet obtenu avec la commande `mount`)