Environnement et cadre de travail

1. Outils de base

a. Fichiers de session

Une fois la connexion réussie, l'utilisateur Ubuntu se trouve donc dans un répertoire nommé le répertoire de travail, c'est-à-dire celui où il se situe (rappel : `/root` pour l'administrateur et `/home/noms_de_l'utilisateur` pour les autres). Au départ, le répertoire courant se confond avec le répertoire de l'utilisateur. Il comprend cinq fichiers cachés (le point devant le nom du fichier indique cet état) qu'un simple `ls` ne peut montrer. Il faut lui ajouter l'option `-a` :

```
ls -a
```

La sortie de la commande montre :

- Un simple point : indique le répertoire lui-même.
- Un double point : indique le répertoire parent.
- Le fichier `.bash_logout` : script d'exécution de commandes juste avant la fin de session.
- Le fichier `.bashrc` : script d'exécution de commandes juste avant le début de session.
- Le fichier `.profile` : script de positionnement de l'environnement utilisateur, il lance notamment `.bashrc`.

Avec les trois derniers fichiers, il est facile de modifier les informations lors de l'entrée et la sortie d'une session avec des commandes de l'interpréteur de commandes (shell) BASH.

Exemple :

Éditez le fichier `.profile` et ajoutez-y les deux lignes ci-dessous en fin de fichier :

```
echo " Bienvenue $USER. "
echo -n " Nous sommes le " ; date
```

Explications :

La prochaine session de travail affichera, après l'identification réussie, un message de bienvenue avec le nom de l'utilisateur (variable d'environnement `$USER`). La deuxième ligne affiche la date et l'heure du jour.

L'apprentissage du `shell` BASH, avec l'écriture de scripts dépasse le cadre de cet ouvrage. Il mérite à lui seul un manuel...

---

b. Manuel en ligne

L'utilisateur, qu'il soit administrateur ou utilisateur, dispose d'un manuel en ligne expliquant et donnant des renseignements sur toutes les commandes Linux. Pour l'appeler il suffit de taper le mot-clé man suivi de la commande désirée :

```
man ls
```

La sortie équivaut à un affichage filtré par la commande `less`. On se déplace donc dans le manuel avec les touches [Page Up] et [Page Down]. Pour rechercher dans les pages un mot, on utilise la commande de recherche de VIM (d'où son utilité). La sortie du manuel - souvent en langue anglaise, car tout n'est pas traduit - s'effectue classiquement par le caractère `q`.

Sur Ubuntu la prise en compte des pages françaises du manuel en ligne passe par un paquet logiciel :

```
aptitude install manpages-fr
```

Une bonne pratique pour l'administrateur consiste à ouvrir (et laisser ouverte) une console supplémentaire (la sixième par exemple avec [Alt][F6]) réservée à l'utilisation du manuel en ligne.

Une raison évidente d'apprendre l'éditeur VIM : la commande de recherche d'un mot dans l'afficheur du manuel est la même ! Cela peut être utile dans le cas d'une trop abondante documentation...

---

c. Complétion

Le mécanisme de complétion (mot peu usité dans la langue française), évite beaucoup de fatigue à l'administrateur. En tapant partiellement une commande, suivi de deux frappes sur la touche de tabulation, celle-ci se trouve naturellement complétée en fonction des prémices de la demande. Elle complète à la fois les commandes et les arguments de ces commandes.

Exemple :

Si l'on tape juste la lettre `l` suivie de deux tabulations, toutes les commandes commençant par `l` vont s'afficher. Si l'on tape `ls` puis un espace, suivi de deux tabulations, cette fois ce sera la liste des fichiers et répertoires qui s'affichera.

Autre exemple : un `cat /etc/net` et deux tabulations complèteront la commande avec le mot `network` mais s'arrêteront là. Deux touches de tabulations supplémentaires montreront qu'il y a en fait le choix entre `/etc/network/` (sous-répertoire) et `/etc/networks` (fichier).

d. Historique des commandes

Autre outil important pour l'administrateur : la mise en mémoire des commandes tapées dans la console. Au lieu du fichier `.bash_history`, on lui préfère directement la commande `history` qui indique en plus un numéro d'indice :

- Tapez `history` et notez le numéro d'une commande quelconque.

En premier lieu, vous voyez la syntaxe exacte de la commande recherchée (utilisez le mécanisme des [Page Up]/[Page Down] si la liste est trop longue). Mais en plus, il n'est pas nécessaire de la retaper entièrement si vous ne désirez pas faire de modifications. Exemple, j'ai une commande de téléchargement d'un répertoire (très longue) que je veux refaire et que voici :

```
lftp -c "open http://archive.ubuntu.com/ubuntu/dists/lucid/main/ins
taller-i386/current/images/netboot/; mirror"
```

En tapant la commande history dans la console je repère cette commande et je note son indice : 169. Pour relancer la commande sans la retaper, il suffit de noter un point d'exclamation suivi du numéro : `!169`

Autre méthode

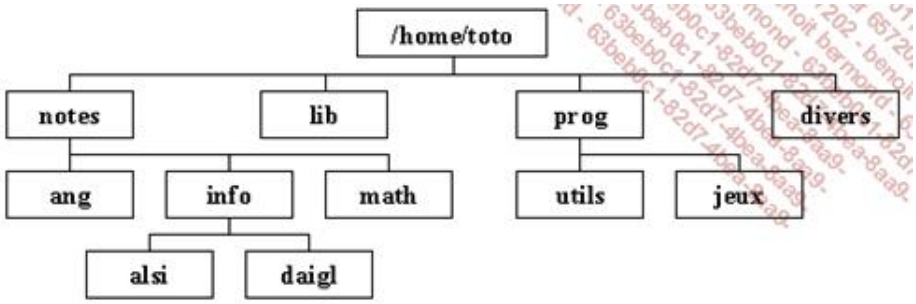
Plus simple, elle se réserve aux commandes tapées il y a peu de temps. En appuyant sur les flèches de direction (haut et bas) vous faites tout simplement défiler les commandes. Il suffit de valider à nouveau la commande pour qu'elle s'exécute.

2. Travailler sur les répertoires

a. Organisation des fichiers

La hiérarchie des répertoires sous Linux commence à la racine (notée par /) et se divise en répertoires, chacun séparé du précédent par une barre oblique / (slash ou barre de division). Travailler au sein de son espace, c'est-à-dire son répertoire de travail, revient à en gérer les sous-répertoires et fichiers. Pour l'administrateur, le champ d'investigation s'étend à tout le système alors que l'utilisateur se limite à son répertoire personnel.

Prenons l'exemple d'un étudiant nommé `toto` :



Cette arborescence montre la structure des répertoires de l'utilisateur. Un document `/home/toto/document.txt` indique que le fichier `document.txt` se trouve dans le répertoire `toto`, lui-même dans le répertoire `home` qui, on l'a vu, est dans la racine matérialisée par la barre de division.

Notions de chemin absolu et de chemin relatif

Cette notion, très importante, donne deux façons différentes pour se situer au sein de l'arborescence des fichiers :

- le chemin **absolu** commence avec la spécification du répertoire racine.
- le chemin **relatif** commence avec le répertoire de travail c'est-à-dire celui **où l'on est** (à ne pas confondre avec le répertoire de l'utilisateur, résultat de la commande `pwd`).

Exemple :

Si vous vous situez dans le répertoire `/home/toto`, la référence au fichier `cours.txt` situé dans le répertoire `alsi` se note :

- `/home/toto/notes/info/alsi/cours.txt` (chemin absolu)
- `notes/info/alsi/cours.txt` (chemin relatif)

Notez bien l'absence du / au début dans le chemin relatif car le départ ou répertoire de travail se note `/home/toto/`.

Les abréviations utiles

La première abréviation se traduit par un double point (..) et fait référence au répertoire parent de celui dans lequel on est.

Exemple :

Si vous êtes dans `/home/toto/ang`, la référence au fichier `script.sh` situé dans `prog` se note :

- `/home/toto/prog/script.sh` (chemin absolu : ne change pas)
- `../../prog/script.sh` (chemin relatif)

En lisant de la droite vers la gauche, le premier double point indique en fait le répertoire parent de `ang`, c'est-à-dire `notes` ; le deuxième indique `toto` (là où vous vous situez).

La deuxième abréviation, un seul point (.) fait référence au répertoire de travail lui-même.

Exemple :

En reprenant le même exemple :

- `./prog/chose.txt` (chemin absolu)
- `notes/info/alsi/truc.txt` (chemin relatif : ne change pas)

Le point au niveau du chemin absolu remplace la séquence `/home/toto`.

b. Commandes usuelles

1) Par rapport à l'arborescence du système de fichiers :

```
pwd         # retourne le chemin absolu du répertoire dans lequel
            # vous vous trouvez
cd          # cette commande permet de se déplacer dans le système de
            # fichiers
```

Exemples :

Je suis dans `/home/toto/prog/utils` et je veux aller dans le répertoire `alsi` :

- `cd /home/toto/notes/info/alsi` (par le chemin absolu)
- ou `cd ./notes/info/alsi` (par le chemin absolu)
- `cd ../../notes/info/alsi` (par le chemin relatif)

Je veux simplement remonter d'un niveau :

- `cd /home/toto/prog` (par le chemin absolu)
- ou `cd ./prog` (par le chemin absolu)
- `cd ..` (par le chemin relatif)

Les utilisateurs du système Windows et du DOS notent la présence obligatoire sur Ubuntu de l'espace entre la commande et le double point. Espace non nécessaire sur le système de Microsoft.

2) Lister le contenu des répertoires (déjà vu) :

`ls [options]# liste fichiers et répertoires`

Les options les plus courantes :

`-a` : liste tous les fichiers, y compris ceux cachés (commençant par un point)

`-i` : liste les fichiers et leurs inodes (voir le concept plus bas)

`-s` : liste les fichiers et leur taille en kilo-octets

`-l` : liste les fichiers avec plus de détails (format dit long)

Toutes les combinaisons d'options sont possibles.

Exemple :

`ls -ail /var` (liste le contenu du répertoire /var)

```
root@hardy:/etc# ls -ail /var/
total 48
 97537 drwxr-xr-x 14 root root  4096 2008-05-22 18:06 .
     2 drwxr-xr-x 22 root root  4096 2008-05-22 18:11 ..
105666 drwxr-xr-x  2 root root  4096 2008-05-22 17:47 backups
 97544 drwxr-xr-x  8 root root  4096 2008-05-22 18:06 cache
 97540 drwxr-xr-x 23 root root  4096 2008-05-25 11:26 lib
105669 drwxrwsr-x  2 root staff 4096 2008-04-15 07:53 local
  5712 drwxrwxrwt  4 root root   100 2008-05-27 18:19 lock
105670 drwxr-xr-x  9 root root  4096 2008-05-27 11:27 log
105759 drwxrwsr-x  2 root mail  4096 2008-05-22 17:46 mail
105758 drwxr-xr-x  2 root root  4096 2008-05-22 17:46 opt
  5708 drwxr-xr-x  8 root root   360 2008-05-27 11:28 run
105671 drwxr-xr-x  3 root root  4096 2008-05-22 17:53 spool
105672 drwxrwxrwt  2 root root  4096 2008-05-26 18:05 tmp
106903 drwxr-xr-x  2 root root  4096 2008-05-22 18:29 www
root@hardy:/etc#
```

3) Commandes sur les répertoires et les fichiers :

```
mkdir    # crée un répertoire
rmdir    # supprime un répertoire (attention : le répertoire doit
         # être vide)
touch    # crée un fichier de type texte (ASCII) vide
rm       # efface un fichier
cp       # copie un fichier ou répertoire
mv       # même démarche que la copie mais simplement un
         # déplacement
```

Exemples :

`mkdir geosi` : crée le répertoire `geosi`.

`rmdir geosi` : supprime le répertoire `geosi`.

`touch essai.txt` : crée le fichier vide `essai.txt`.

`rm essai.txt` : supprime le fichier `essai.txt`.

`rm -f essai.txt` : la même chose sans demande de confirmation.

`cp essai.txt notes/ang` : copie le fichier dans le répertoire `ang`.

`cp essai.txt notes/ang/test.txt` : cette fois en changeant le nom.

4) La recherche d'un fichier :

`find # cherche un fichier`

C'est une commande plus complexe que les précédentes. En voici la forme d'utilisation la plus courante :

```
find -name toc.txt        # cherche le fichier toc.txt dans le
                          # répertoire courant
find /home -name "test*"  # cherche tous les fichiers ou répertoire
                          # commençant par test à partir d
                          # répertoire home (l'étoile remplace le
                          # reste)
```

On lui préfère l'outil `locate`, couplé avec la commande `updatedb` plus simple d'utilisation. La commande `find` retourne beaucoup d'erreurs en allant chercher là où elle ne possède pas les droits suffisants. Sur une distribution serveur Ubuntu, le paquetage logiciel (de nom `locate`) est installé par défaut. La première étape consiste à lancer la simple commande `updatedb` destinée à répertorier dans une base l'ensemble des fichiers :

`updatedb`

Ensuite pour chercher, la commande `locate` est suivie du nom du fichier (partiel ou complet), précédé éventuellement d'une indication de répertoire :

```
locate [rep] motif
```

Exemple :

`locate /etc "conf"` : recherche tous les éléments contenant le motif `conf` dans son intitulé.

L'avantage réside dans la rapidité de la recherche, l'inconvénient dans l'obligation de relancer périodiquement la commande `updatedb` afin de mettre à jour la base d'index.

5) Commandes particulières :

```
cat        # concatène des fichiers et les affiche sur la
           # sortie standard (écran par défaut)
```

Bien que plutôt réservée sur les fichiers cette commande très puissante possède bien d'autres fonctionnalités... Nous ne voyons pour l'instant que :

```
cat nom_de_fichier        # affiche le contenu d'un fichier
cat > nom_de_fichier      # crée un fichier
```

Dans ce dernier cas, vous tapez directement les lignes de texte désirées par le clavier à la suite de la commande. La fin de saisie s'effectue par [Ctrl] **d**. L'administrateur utilise la commande `cat` sur des petits fichiers afin d'en visualiser rapidement le contenu.

```
clear            # efface toute la console et replace le curseur
                 # en haut à gauche de l'écran
whoami           # donne le nom de l'utilisateur courant
hostname         # donne le nom de la machine
```

Ces trois dernières commandes ne posent aucune difficulté.

3. Travailler sur les fichiers

a. Types de fichiers

Un fichier au sens UNIX (et Linux) représente à la fois une source de données (à lire) ou une destination pour ces données (à écrire). La structure arborescente se compose de nœuds ou répertoires et de feuilles ou fichiers. On peut donc avoir :

- le fichier **répertoire** : fichier particulier contenant des références vers d'autres fichiers (pointeurs). Au sens littéral du terme, un fichier répertoire ne "contient" pas ses fichiers, mais seulement les liens vers ceux-ci. Un fichier répertoire contient au minimum une référence : celle du répertoire qui le contient (parent), sauf pour la racine principale / évidemment.

- le fichier **ordinaire** : fichier contenant des données. Les plus nombreux, ils peuvent être de type texte ou binaire.

- le fichier **spécial** : fichier dit de "dispositif", il représente un périphérique physique : clavier, écran, disques durs, lecteur USB, etc.

D'autres "types" de fichiers (encore que cela soit impropre comme appellation) existent :

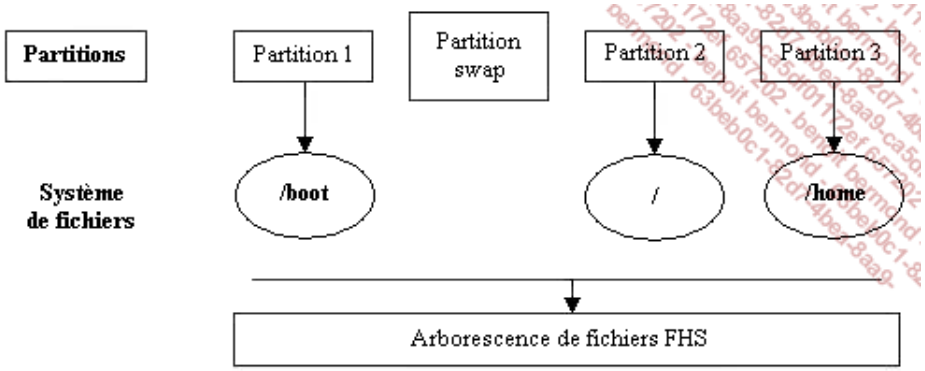
- Les **liens** : pointeurs vers d'autres fichiers.

- Les **tubes** : redirection des entrées/sorties.

- Les **sockets** : canaux de communication inter-processus.

On l'a vu, le système de fichiers Linux se construit en arborescence et il utilise pour enregistrer chaque élément une table des **i-nœuds** ou **inodes**. Chaque création de système de fichiers (sur une partition) sous Linux s'accompagne d'une table des inodes.

En voici un exemple :



La partition swap ne donne pas lieu à une table des inodes. Elle est employée par le système comme extension de la mémoire RAM (mémoire virtuelle). Le *Filesystem Hierachy Standard* définit l'organisation du système de fichiers commune (théoriquement) à toutes les distributions.

Il ne pas confondre organisation du système de fichiers et type de systèmes. L'apparition de systèmes de fichiers journalisés plus robustes, comme **ReiserFS** ou **ext3fs** (intégrés au noyau) ou **jfs** (IBM, maintenant très performant) ou **xfs** (plus robuste) ne change rien dans l'arborescence FHS, ni fondamentalement dans le principe des inodes. En gros, ce nouveau type de fichiers utilise le principe du cache tiré du monde des SGBD.

b. Processus de création d'un fichier

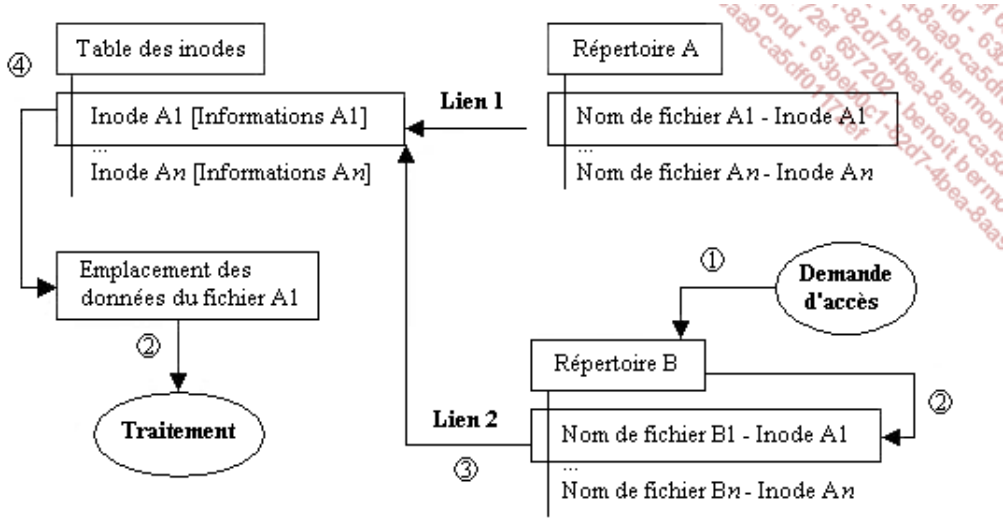
Une fois définie la table des inodes (créée à l'initialisation du système de fichiers), le processus de création d'un fichier est donc le suivant :

- Détermination de l'espace sur le disque pour stocker les données du fichier.
- Création d'un inode unique (appelé aussi nœud d'index) contenant des informations : emplacement, propriétaire, etc.
- Mise en place dans le fichier répertoire du couple **inumber** (numéro) et nom de fichier.

c. Pointeur sur un fichier : le lien

Quand un utilisateur désire manipuler un fichier, le système recherche le nom dans le fichier répertoire, trouve l'inumber et récupère les données correspondantes dans la table des inodes.

Cette relation entre le nom de fichier et son inode s'appelle un lien. Sous Linux, un même fichier peut disposer de plusieurs liens et donc de plusieurs noms différents. C'est un mécanisme important et, même si vous n'en voyez pas l'intérêt de prime abord, il est important d'en connaître les mécanismes. Les liens multiples sont possibles car ce n'est pas le nom qui identifie un fichier mais son inumber. Le plus simple consiste à examiner le schéma suivant :



On constate :

- L'existence de deux noms de fichiers différents avec le même inode.
- Une modification du fichier B1 (départ de la demande en ①) entraîne une modification sur A1 car, en fait, le fichier de données est unique.
- La possibilité de créer un lien avec la commande `ln` (voir ci-dessous), ce qui incrémente un compteur.
- Une copie du fichier créée (un fichier avec un inode différent), ce n'est donc pas un lien supplémentaire. Ce type de liens s'appelle liens "en durs" et ne peuvent s'appliquer que dans le système de fichier. Pour créer un lien entre deux fichiers dans deux systèmes différents, on utilisera des liens dits "symboliques".

d. Commandes usuelles

1) Commandes classiques :

```
cmp        # compare deux fichiers, affiche un message en cas
           # de différence (plutôt pour les fichiers binaires)
diff       # trouve les différences entre des fichiers
file       # donne le type d'un fichier
head       # affiche le début d'un fichier
basename   # élimine le chemin d'accès d'un fichier
paste      # regroupe les lignes de plusieurs fichiers
stat       # affiche des renseignements et statistiques sur un
           # fichier ou répertoire
tail       # affiche les dernières lignes d'un fichier
whereis    # recherche les fichiers exécutables, les sources et les
           # pages de manuel
which      # localise une commande
```

La plupart des commandes sont à découvrir à l'aide du manuel en ligne, mais l'une d'entre elles mérite qu'on s'y arrête. Il s'agit de la commande `tail`, très utile à l'administrateur pour voir le contenu des journaux, plus précisément pour voir en temps réel le résultat d'une manipulation.

On veut voir par exemple en temps réel ce qui se passe sur la machine à partir du journal général :

- Ouvrez une console supplémentaire avec une connexion en administrateur ([Alt][F2]).
- Tapez la commande `tail -f /var/log/syslog`.

Le résultat affiche la fin du fichier du démon général `syslog` et ne se ferme pas. Une manipulation ultérieure sur une autre console entraînant un message sera visible immédiatement en s'inscrivant à la fin du fichier.

2) Commandes sur les liens :

La commande `ls` avec l'option `i` pour inode liste les liens :

```
ls -i    # liste les fichiers avec les numéros d'inodes du
         # répertoire courant
```

Il existe aussi la commande spécifique `ln` qui crée un lien supplémentaire :

```
ln [-s] fichier_source fichier_cible
```

Exemple :

`ln nafnaf.txt nifnif.txt` : crée un nouveau pointeur sur le fichier `nafnaf.txt`.

L'option `-s` permet de créer un lien symbolique (à ne pas utiliser pour l'instant !). Par analogie, on peut dire que cette notion s'apparente au raccourci sur un bureau graphique.

3) Autres commandes spécifiques au système de fichiers :

```
du           # affiche des statistiques d'utilisation du disque
```

```
df            # donne la quantité d'espace occupé des systèmes de
              # fichiers
badblocks     # recherche des blocs défectueux sur un périphérique
fdisk         # utilitaire de manipulation de la table de
              # partition d'un disque
mount         # monte un système de fichiers
umount        # démonte un système de fichiers
```

Exemples d'utilisation :

`du -hs /etc` : affiche en Ko ou Mo le poids total du répertoire `/etc`.

`df -h` : affiche par système de fichiers la taille, le poids, l'utilisation et son pourcentage.

4. Entraînement

Comme pour la compréhension des droits utilisateurs, les notions abordées dans ce chapitre nécessitent un minimum de pratique. Afin de vous aider dans la maîtrise des commandes, voici une liste de **questions/exercices** (les solutions se trouvent toujours en Annexe) :

Cadre de travail

Session ouverte en `root` sur une version Ubuntu serveur.

Exercices

- Tapez la commande permettant de voir dans quel répertoire vous êtes.
- Listez normalement le contenu de votre répertoire.
- Listez le contenu du répertoire avec les fichiers cachés et le format long.
- Utilisez le manuel en ligne pour connaître et comprendre la commande `sleep`.
- Tapez maintenant `sleep --help`, quelle en est l'utilité ?
- Si vous tapez `sleep 360`, quelle combinaison de touches permet de l'interrompre ? Testez-la.
- Quelle est la taille du répertoire `/usr` ?
- Allez dans le répertoire `/etc` par le chemin absolu.
- Faites un tube avec d'un côté la commande `ls *.conf`, de l'autre `wc -l`, quel en est le résultat ?
- Tapez tout simplement `cd` et validez. Que s'est-il passé ? Que signifie le tilde (~) ?
- Allez dans votre répertoire et créez deux sous-répertoires : `hobbit` et `elfe`, vérifiez la création par la commande appropriée.
- Allez dans `hobbit` et créez trois fichiers `merry`, `pipin` et `gollum` par la commande `touch`.
- Sans changer de répertoire, créez le sous-répertoire `elfedesbois` dans `elfe` et par le chemin relatif.
- Sans changer de répertoire, créez le fichier `legolas` dans votre répertoire personnel et par le chemin absolu.
- Allez dans `elfedesbois` obligatoirement en trois commandes (niveau par niveau).
- Sans changer de répertoire, déplacez `legolas` pour le mettre là où vous êtes et par le chemin relatif.

- Sans changer de répertoire renommez `gollum` en `sam` par le chemin absolu.
- Trouvez, par la commande `find`, le fichier `interfaces` à partir de `/etc`.
- Retournez dans votre répertoire et créez le fichier `riri.txt`.
- Dupliquez-le par la commande `cp` en changeant son nom en `fifi.txt`.
- Créez un lien par `ln` en changeant son nom en `loulou.txt`.
- Éditez les inodes par `ls` et l'option adéquate et concluez.
- Par VIM, mettez une phrase dans `riri.txt`.
- Éditez `fifi.txt` et `loulou.txt` ; concluez.
- Effacez `riri.txt`, que s'est-il passé pour `fifi.txt` ?
- Quel est le type du fichier `/etc/passwd` ?