# T.P. 1 : Premier pas avec Linux dans la distribution Ubuntu

## 1 Arborescence du système de fichiers

Un système de fichiers, quelle que soit sa nature physique, est organisé en répertoires ou dossiers (*directories or folders*), dans lesquels on trouve des fichiers ou d'autres répertoires.

### 1.1 Deux façons de naviguer dans ces répertoires.

a) A l'aide de l'interface graphique et de la souris. A gauche, il y a un *dock* avec des icônes. Cliquez sur la deuxième : Dossier Personnel. Vous voyez votre *répertoire personnel utilisateur*. Si vous voulez savoir comment il se situe, allez voir `Système de fichiers` dans la colonne de gauche.

b) A l'aide d'un `Terminal`. Dans le dock, cliquer sur l'icône `Tableau de bord` puis taper Terminal. La commande centrale pour se déplacer dans l'arborescence est alors la commande `cd` (*change directory*). Celle pour voir le contenu du répertoire courant est `ls` (*list*). Jouons un peu avec ces commandes.

Il ne faut pas croire que l'usage du Terminal soit dépassé par celui de l'interface graphique. Dès qu'on a une tâche un peu plus complexe à effectuer, il peut être plus performant.

### 1.2 Descriptions des chemins

• Le niveau le plus haut du système de fichiers est appelé la *racine.*[1] Elle est symbolisée par un / (slash)[2].

• Le répertoire personnel de l'utilisateur, ici /home/prepas, peut aussi être symbolisé par un ~ (tilde) (il apparaît sur le *prompt* du Terminal).

La commande `cd` sans argument vous ramene dans votre répertoire « perso. » /home/prepas

Un fichier ou un dossier est localisé par son chemin (*path*).

a) Le chemin *absolu* est repéré par rapport à la racine du système de fichiers. Exemple à l'écran avec l'interface graphique (en cliquant sur les petits triangles à droites des fichiers en présentation par liste).

   Dans le terminal, la commande `pwd` donne le chemin absolu du répertoire courant. Par exemple :

   `/home/prepas/Documents`

b) Le chemin *relatif* relie le dossier considéré au dossier dans lequel on se trouve. Cela est important si on se place dans un terminal.

   Ainsi, si on est dans le dossier `/home`, le chemin vers `fpc` est `cpge/fpc`. *Notez que le chemin* relatif *ne commence pas par un slash.*

   `../` représente un remontée d'un cran dans les répertoires. Ainsi, si on veut remonter de deux répertoires, avant de repartir dans un autre, on écrira `../../`

`./` représente le répertoire courant : nous y reviendrons

1. La racine en haut, c'est bizarre : nous y reviendrons, en info. les arbres poussent à l'envers.
2. En Windows c'est très différent : à la base on repère dans quel système de stockage on est par exemple `C :` et après en Windows, il y a des *backslash*

### 1.3 Pour mieux jouer à se déplacer, créons des répertoires et des fichiers

Dans le Terminal, avec la commande `mkdir` (*make directory*) créer un répertoire avec votre nom de famille. A l'intérieur de ce répertoire créer un répertoire `TP1`

A l'intérieur du `TP1` créer un fichier vide du nom de `vide` à l'aide de la commande `touch` suivie du nom du fichier.

Dans le Terminal, taper `gedit` qui est un petit éditeur de texte et taper le petit programme suivant (en respectant l'indentation) qui est du code pour le langage Python :

```
i=0
while i<100:
    i=i+1
    print(i)
```

L'enregistrer sous le nom `test.py`

Modifier le fichier en remplaçant le `i+1` par `i-1` et l'enregistrer sous le nom `test2.py`

L'attribut `.py` sera utile pour savoir qu'il s'agit d'un code pour PYTHON

Vous avez maintenant trois fichiers dans `TP1`.

## 2 Petite synthèses des commandes Unix utiles :

Dans le Terminal, on peut entrer des commandes qui peuvent être suivies d'un nom de fichier par exemple, qui est l'argument de la commande. On peut intercaler des options sont précédées du signe "-". Une commande aura donc la structure :

`macommande -options arguments`

Pour obtenir de l'aide sur une commande, on utilise la commande `man` : ainsi pour savoir comment utiliser la commande `pwd`, on tape `man pwd`

Commandes les plus utilisées :

- `pwd` : indique le chemin du dossier courant (*path of working directory*)
- `ls` : liste les fichiers dans votre répertoire courant, ou si suivi d'un chemin liste les fichiers du dossier spécifié. Options utiles : l, a
  - Essayer `ls -a` dans votre répertoire utilisateur : il montre *tous* les fichiers, `a= all` mêmes les cachés, ceux qui commencent par un `.`
  - Essayer `ls -l` on va en reparler plus loin.
- `cd` : permet de changer de répertoire courant. Suivi du nom du chemin `cd` "tout court" vous déplace dans votre home directory, c'est-à-dire à la racine de votre dossier personnel.
- `mkdir` : créer un répertoire.
- `cp` : copie de fichiers. Syntaxe cp fichiers à copier destination. Options utiles R (pour les répertoires) et f (pour forcer le remplacement des fichiers déjà existant).
- `rm` : efface le fichier.
- `mv` : déplace le fichier dans le répertoire indiqué. Si on indique le chemin vers un fichier, on va renommer le fichier. Ex : mv test2.py test.py renomme test2.py en test.py
- `chmod` : changer les autorisations (voir 6).
- `man` : `man <nom de commande>` affiche l'aide d'une commande.

# 3 Pour information : les répertoires sur la racine Ubuntu

Voici les principaux répertoires d'un disque Ubuntu :

- **/bin** : Les programmes systèmes importants.
- **/boot** : Les fichiers de démarrage
- **/dev** : Point d'entrée de vos périphériques, utilisé par le système.
- **/etc** : Les fichiers de configuration de votre système.
- **/home** : La maison des utilisateurs ! Chaque utilisateur y a un répertoire à son nom, avec ses documents et ses fichiers personnels de configuration.
- **/lib** : Les bibliothèques indispensables au système.
- **/media** : Point d'accès où sont montés les autres disques durs, les CD, DVD, clés USB...
- **/proc** : C'est un répertoire virtuel qui contient l'état de la machine en temps réel : pages mémoire, IRQ..
- **/root** : La maison de l'administrateur, avec ses fichiers.
- **/sbin** : Les outils GNU indispensables au système.
- **/tmp** : Devinez... Eh oui, les fichiers temporaires. Ils sont effacés à chaque redémarrage de l'ordinateur.
- **/usr** : Un gros morceau, contient les programmes et bibliothèques utiles aux utilisateurs.
- **/var** : Partie variable du système, avec les informations sur ce qui se passe sur votre machine. Utile quand quelque chose ne va pas, plus d'informations sont disponibles alors dans /var/log.

A part le `/home` et le `/media` vous n'aurez pas à les toucher (d'ailleurs vous ne pourrez pas).

# 4 Exercices d'utilisation du Terminal

a) Lister le contenu de votre répertoire TP1. Il doit contenir trois fichiers. Que fait la commande `ls *.py` ?

b) Créer un répertoire Python dans TP1. Y Copier les deux fichiers .py *en un seul coup* en gardant la copie originale.

c) Effacer les deux fichiers que vous venez de créer dans le répertoire Python et recommencer la même manip. avec `mv` au lieu de `cp`.

d) Insérer votre clé U.S.B. et copiez-y les trois fichiers que nous avons créés.

**Attention aux noms de fichiers avec des espaces :** ils sont plutôt à éviter. Si vous devez les manipuler dans le Terminal l'espace doit être remplacé par un backslash `\` suivi d'un espace. Par exemple si votre clé usb s'appelle : ma clef, il faudra rentrer `ma\ clef`

# 5 Première utilisation de Python dans le Terminal

a) Dans le Terminal, tapez `python test.py`. Si cela marche tant mieux, sinon dans quel répertoire vous trouvez-vous ?

b) Faites la même chose avec `test2.py`. Comment l'arrêter ?

Méthode 1 : fermer la fenêtre du Terminal, cela suffit.

Méthode 2 : ouvrir un autre Terminal. Lancer la commande `ps -u` pour voir *tous* les *processus* (`ps`) que vous avez lancés. Identifier celui qui doit être tué et tuez le avec la commande `kill` suivie du *numéro du processus*.

# 6 Un mot sur les permissions sur les fichiers

## 6.1 Mini-cours

Chaque fichier ou dossier dispose de 3 propriétés d'accès :

- Accès en lecture : le contenu du fichier peut être lu, édité dans un logiciel.
- Accès en écriture : le contenu du fichier peut être modifié.

- Accès en exécution : si le fichier est un binaire ou un script, le programme peut être exécuté. Pour un dossier, il s'agit de se déplacer dans ce répertoire (« l'ouvrir » ou utiliser la commande cd).

Chacune de ses propriétés peut être affectée ou non à un fichier et l'autorisation d'utiliser ces propriétés peut-être donné ou non à tel ou tel utilisateur. Les systèmes Unix distinguent trois familles d'utilisateurs concentriques :

- Le possesseur du fichier : `u` (comme *user*)
- Le groupe principal auquel appartient le possesseur `g` (comme *group*)
- Les autres utilisateurs `o` (comme *others*).

Tout utilisateur appartient à au moins un groupe. Par défaut, le système peut créer un groupe portant le nom de l'utilisateur lors de sa création. Mais en général un utilisateur appartient à plusieurs groupes. Ainsi, un utilisateur quelconque va appartenir au groupe staff sous MacOSX, users, etc... Au lycée, les élèves d'une même classe appartiennent à un groupe portant le nom de leur classe.

Les propriétés d'accès à un fichier par les utilisateurs sont appelées autorisations, permissions ou encore droits. A priori, les permissions les plus larges sont données au propriétaire du fichier, les plus restreintes aux autres utilisateurs.

Prenons l'exemple suivant : romain est un utilisateur appartenant au groupe staff. La commande ls -l liste un dossier en affichant les autorisations.

```
bash:~/$ ls -l
drwxrw-r-- 1 romain staff 23 ...config
-rwxrw-r-- 1 romain staff 23 ...toto.txt
```

- Le premier caractère indique si on a affaire à un dossier (d comme directory) ou non (-).
- Les trois suivants donnent les autorisations du propriétaire. (`read, write,(e)xecute`).
- Les trois suivants donnent celles de son groupe.
- Les trois derniers ceux des autres utilisateurs.

Ici on a donc : config est un dossier qui peut être lu et modifié par romain. Pour un dossier l'exécution est l'ouverture du dossier et l'affichage de son contenu. Ce dossier peut être lu et modifié par tous les utilisateurs du groupe staff, les autres utilisateurs (ceux qui ne sont ni romain ni membres du groupe staff) ne peuvent que le lire. Concrètement, un utilisateur du groupe staff peut utiliser sur ce dossier la commande ls pour lire le contenu de ce dossier, peut modifier le dossier en le renommant (commande mv) mais n'ayant pas les droits d'exécution il ne peut y déposer ou y enregistrer un fichier.

toto.txt est un fichier, que romain peut lire, modifier et exécuter. Un membre de staff peut le lire et le modifier, les autres ne peuvent que le lire.

Pour modifier les autorisations, le plus simple est d'utiliser la commande `chmod` de la manière suivante : `chmod [options] nomdufichier`

Les options doivent indiquer les modifications de qui sont modifiées (u pour le propriétaire, g pour le groupe, o pour les autres) et quelle autorisation `(r, w ou x)` est ajoutée ( `+` ) ou retirée ( `-` ).

- `chmod g+rw toto.txt` ajoute le droit en lecture et en écriture au groupe, pour le fichier toto.txt.
- `chmod ug-r toto.txt` retire l'autorisation de lecture au propriétaire et au groupe.
- `chmod u+w,o-r toto.txt` ajoute le droit en exécution au propriétaire et retire aux autres (ni propriétaire ni groupe) le droit de lecture.

### 6.2 Mini-exercices

a) Rendre le fichier `test.py` exécutable, à l'aide de la commande `chmod`. Trouver aussi comment le faire avec l'interface graphique.

b) Que se passe-t-il si on double-clique dessus ?

c) Lancer le fichier dans le Terminal *en faisant précéder son nom de* `./` dans le répertoire courant.

d) Cela n'a pas marché ! En fait le `shell` (le programme qui gère le Terminal) ne sait pas qu'il s'agit d'un programme PYTHON. A l'aide de gedit rajouter au début de `test.py` la ligne : `#!/usr/bin/python3`

Recommencer alors le b) et le c).

## 7 Ce qu'on va vraiment utiliser pour faire du Python : I.E.P.

a) A l'aide du terminal, lancer IEP Quelle est la différence entre les commandes `iep` et `iep&` ?

b) Ouvrir firefox et accéder à l'aide en ligne : `http://192.168.numero_de_la_salle.1` Par exemple `http://192.168.18.1`

c) Suivre les instructions pour configurer python3 dans IEP.

d) Ouvrir le script test.py dans IEP. L'exécuter.

e) Faire de même avec test2.py. L'interrompre.