L'interface graphique de Linux est très pratique pour des opérations simples, mais pas adapté aux opérations nombreuses ou complexes (renommer tous les fichiers dont le nom contient toto dans un dossier ou ses sous-dossiers). Travailler en ligne de commande dans un terminal est beaucoup plus puissante.

1 Quelques commandes utiles

- **pwd : print working directory** affiche le chemin d'accès vers le répertoire courant.
  `$ pwd` → affiche par exemple `/home/root`
- **cd : change directory** change le répertoire courant.
  `$ cd chemin_du_repertoire`
- **ls : list** liste le contenu d'un répertoire
  `$ ls` → liste le contenu du répertoire courant
  `$ ls dossier1` → liste les fichiers du `dossier1`
  `ls -la` → liste le dossier courant avec tous les détails
  `ls -a` → affiche les fichiers cachés
  `ls -l` → affiche les détails (sans les fichiers cachés)
- **mkdir : make directory** crée des répertoires (ou dossiers)
  `$ mkdir dir1` → crée un répertoire `dir1`
  `$ mkdir dir1 dir2 dir3` → crée trois répertoires
- **rmdir : remove directory** supprime des répertoires non vide
  `$ rmdir dir1` → supprime le répertoire `dir1`
  `$ rmdir dir*` → supprime tous les répertoires commençant par `dir`
- **mv : move** déplace ou renomme un fichier
  `$ mv myfile newname` → renomme le fichier `myfile` en `newname`
  `$ mv dir1/f1 dir2/f2` → déplace le fichier `f1` du répertoire `dir1` dans le répertoire `dir2` et le renomme `f2`
- **cp : copy** copie des fichiers
  `$ cp monfichier mondossier` → copie `monfichier` dans `mondossier`
- **rm : remove** supprime des fichiers, à utiliser avec précaution. Il n'y a pas de "undo".
  `$ rm mondossier/monfichier` → supprime le fichier `monfichier` dans `mondossier`
  `$ rm -r dir1` → supprimer le dossier `dir1` avec ses sous-dossiers et fichiers (très dangereux !)
- **echo** : affiche un message sur l'écran
  `$ echo Hello World` → affiche Hello World
  `$ echo Hello World > f.txt` → écrit Hello World dans le fichier `f.txt` (écrase le fichier)
  `$ echo Other line >> f.txt` → ajoute une deuxième ligne dans le fichier
- **cat : catenate** renvoie le contenu de fichiers texte et concaténe des fichiers
  `$ cat f.txt` → affiche le contenu de `f.txt`
  `$ cat fichier1 fichier2` → affiche le contenu des deux fichiers concaténés
  `$ cat file1 file2 > file3` → concatène `file1 file2`, enregistre le résultat dans `file3`
  `$ cat f* > nfichier` → concatène tous les fichiers commençant par `f`
  `$ cat > nouveaufichier` → crée un nouveau fichier, entrer le texte puis `CTRL-d`
- **less** : affiche un fichier page par page (l'ancienne version s'appelait more car more s'affiche en bas de page pour indiquer qu'on peut voir la suite).
  `$ less nfichier`
- **emacs : editor macros** ouvre un fichier dans un éditeur
  `$ emacs monfichier` → ouvre `monfichier` dans un éditeur
  D'autres éditeurs existent, notamment `vi` (l'éditeur de base, pas très convivial), `vim`, une version améliorée et `nano`.
- **sudo : switch user and do** passe en mode admin pour exécuter une commande
  `$ sudo apt install tree` → installe l'application `tree`
  `$ tree` → exécute l'application `tree`
  `$ tree | less` → exécute `tree` et affiche le résultat via `less`
- **man : manual** affiche le manuel d'instruction d'une commande
  `$ man mv` → affiche l'aide de la commande `mv`
  `$ tree` → exécute l'application `tree`
  `$ tree | less` → exécute `tree` et affiche le résultat via `less`

- **chmod : change mode** modifie les droits de fichiers (voir plus loin)
- Autres commandes à tester : il faudra peut-être les installer
  - `whoami` : renvoie le nom de l'utilisateur
  - `df` : affiche les disques avec l'espace mémoire utilisé
  - `lstopo` : montre la topologie du système
  - `ps, pstree, top` : affiche tous les processus

2 Caractères spéciaux

2.1 Caractères de redirections : >, >>, |

- > redirige la sortie vers un fichier (à la place de l'affichage sur l'écran).
- >> ajoute le résultat de la commande précédente au fichier désigné plutôt que de le remplacer.
- | redirige le résultat de la première commande vers la deuxième commande.
  Syntaxe : `commande1 | commande2`
  `$ ls rep | wc -l` → affiche le nombre de lignes renvoyées par `ls rep`, (le nombre de fichiers et dossiers du répertoire rep)

2.2 Caractères de sélection multiple : *, ?, []

Ces caractères permettent de sélectionner un ensemble de fichiers ou de dossiers dont les noms correspondent à un schéma

- * remplace n'importe quel caractère, ou chaine de caractères, ou aucun caractère.
  `$ ls dos*/fi*` → va sélectionner tous les fichiers commençant par `fi` dans tous les dossiers commençant par `dos`.
- ? remplace un caractère quelconque.
  `$ cat t?t?` → affichera le contenu de tous les fichiers de quatre lettres avec un `t` en première et en troisième position.
- une suite de caractères entre crochets [ ] désigne successivement chaque caractère de la suite ; si le répertoire courant contient les fichiers `fic1`, `fic2` et `fic3`, alors `ls fic[123]` sera équivalent à `ls fic1 fic2 fic3`.
- deux caractères séparés par un - entre crochets [ ] (par exemple [a-e]) désigne successivement tous les caractères de l'intervalle de caractères ; par exemple, si le répertoire courant contient des fichiers `fica`, `ficb`, `ficc`, `ficd`, et `fice`, alors la commande `ls fic[a-e]` est équivalente à `ls fica ficb ficc ficd fice`.

2.3 Chemin d'accès : /, ~, ..

- / permet de séparer les dossiers ou d'accéder à la racine de l'arborescence
  `$ ls dir1/dir2/` → liste le contenu de `dir2` qui est dans `dir1`
  `$ cd /` → accède à la racine de l'arborescence
  Un chemin d'accès commençant par / est donné à partir de la racine de l'arborescence. On dit que c'est un chemin absolu. Dans le cas contraire, le chemin part de la position actuelle : il s'agit d'un chemin relatif.
  `$ /media/nomcleusb/` → pour accéder à la clé USB
- ~ désigne le nom du répertoire utilisateur de l'utilisateur courant.
- . désigne le répertoire actif et .. le répertoire parent Si un dossier travail contient deux dossiers `dir1` et `dir2`, et si `dir1` contient un dossier `dir11` dans lequel on se situe, alors `$ ls ../../dir2` va lister le contenu de `dir2`.

Remarque : sous Windows, les chemins commencent par un identifiant de disque ($D:\backslash$) et les noms des dossiers séparés par \ (en Linux, c'est /).

2.4 Fichiers cachés

- Les fichiers dont le nom commence par un "." sont les fichiers cachés en Linux.
  On y accède en incluant le . dans le nom et on les liste avec `ls -a`.

3 Les droits

3.1 Catégories d'utilisateurs

Les droits de chaque fichier ou dossier sont déterminés pour trois catégories d'utilisateurs :

- propriétaire (`u` : user)
- groupe propriétaire (`g` : group)

- tous les autres (`o` : other)

Chaque catégorie possède ses types d'accès `r` (read) `w` (write) `x` (execute).

3.2 Affichage de droits

`ls -l` donne les droits des fichiers au format `- --- --- ---`.
Le premier caractère est le type de fichier :

- `-` : fichier régulier
- `d` : répertoire
- `l` : lien symbolique

Les trois triplets suivants correspondent aux :

- `---` droits du propriétaire (`rwx`)
- `---` droits du groupe (`rwx`)
- `---` droits des autres (`rwx`)

Exemple : `-rw-r-r` signifie que le propriétaire a les droits en lecture et écriture, le groupe et les autres n'ont que le droit en lecture.

3.3 Droits sur un répertoire

`ls -ld` affiche les droits du répertoire :

- `r` permet de consulter ses entrées (autorise `ls`)
- `w` donne le droit de création, renommage, suppression d'une entrée
- `x` donne le droit de le traverser avec cd pour aller dans un sous-répertoire

Pour renommer un fichier, il faut le droit d'écriture sur le répertoire, pas sur le fichier.

3.4 Modification des droits sur un fichier/répertoire existant

On utilise `chmod droit nomfichier`. Droits à appliquer :

- Catégorie : `u` (user), `g` (group), `o` (other), ou `a` (all)
- Opérations : `+` (ajout), `-` (retrait), `=` (affectation)

Exemple : `chmod u+rx,o=r monfichier`

On peut aussi donner les droits en octal : `1` (exécution), `2` (écriture) et `4` (lecture)

Exemple : `641` pour `rw-r--x`, `752` pour `rwxr-x-w-`

4 TD - Découverte Linux

4.1 Premières commandes

Exécuter les commandes suivantes :

- Tester la commande "list", qui donne le contenu d'un répertoire : `ls` puis `ls -a` et `ls -l` et enfin `ls -a -l` et `ls -al`
- Créer un dossier avec la commande : `mkdir mon_dossier`
- Tester à nouveau `ls` puis `ls -a` et `ls -l` et enfin `ls -a -l` et `ls -al`
- Aller dans le dossier avec la commande : `cd mon_dossier`
- Créer un fichier avec la commande : `touch mon_fichier.txt`
- Éditer le fichier avec avec la commande : `vim mon_fichier.txt` (`vim` est un éditeur de texte)
  - Aide de vim `:help` (les : font partie de la commande)
  - Rentrer du texte touche `insertion` puis écrire
  - Quitter avec `escape` puis pour sauver `:x`
- Afficher le contenu du fichier avec la commande : `more mon_fichier.txt`
- Changer les autorisations de `mon_fichier.txt` : `sudo chmod u-w mon_fichier.txt`
- Éditer à nouveau le fichier avec `vim mon_fichier.txt`
- Rentrer du texte touche `insertion` puis écrire, que constatez-vous ?
- Quitter avec `escape` puis, sans sauver avec `:q`, en sauvant `:x`, et finalement en force avec `:qa!`
- Copier le fichier avec la commande : `cp mon_fichier2.txt mon_fichier3.txt`
- Changer le nom avec la commande : `mv mon_fichier2.txt mon_fichier3.txt`
- Supprimer avec la commande : `rm mon_fichier3.txt` (que constatez-vous ?)
- Aller dans le dossier Documents étape par étape, vous pouvez faire des `ls -a` pour explorer le contenu des dossiers. Commandes : `cd / cd mnt/ cd c/` puis pour aller plus vite `cd  /Documents` ou `cd /home/nsi/Documents/`
- Essayer `cd ..`
- Essayer `cd`
- Utiliser la commande pwd pour savoir où on est (print working directory)
- Recopier un fichier du dossier documents vers `mon_dossier`, commande : `cp mon_fichier /mon_dossier/ mon_fichier` `cd /` revient à la racine du disque puis `ls`
- Revenir au dossier racine linux `cd`
- Essayer `rmdir mon_dossier` pour supprimer ce qu'on vient de faire. Comme ça ne marche pas, alors `rm {r mon_dossier`

4.2 Compléments

4.2.1 Les opérateurs de redirection et le "pipe"

Les opérateurs > et >> permettent d'écrire les sorties d'une commande non pas à l'écran, mais dans un fichier. Placez-vous dans un répertoire non vide et testez `ls -al > liste_fichier.log`
Vous pouvez ensuite ouvrir le fichier crée avec `vim`, ou simplement l'afficher avec `less`.
L'opérateur > crée le fichier de sortie, éventuellement en écrasant un fichier de même nom préalablement existant. L'opérateur >> écrit à la fin du fichier sans l'écraser.
L'opérateur | permet d'enchaîner les commandes. Par exemple, `ps -e` (ou `aux`) permet de lister tous les processus actifs sur le noyau Linux.
La commande `wc` permet de compter (word count) :

- `wc {l` compte les lignes
- `wc -c` les octets,
- `wc -w` les mots.

Pour compter le nombre de processus, on peut faire `ps aux | wc -l`

4.2.2 Aller sur internet.

On n'ouvre pas un navigateur, seuls de rares adresses peuvent être lisibles/accessibles, comme celle ci pour la météo :
`curl wttr.in/chinon`

4.2.3 Installer des programmes

Exemple : installation de `cowsay` :

```
sudo apt-get update
sudo apt-get install cowsay
```

Puis test du programme en tapant :

```
cowsay hello world
```

Options à tester après cowsay et avant le texte :
- `-bdgsty` (une seule lettre)
- `-e 88`
- `-T U`
- `-f type_de_vache` (par exemple dragon)

Exemple : installation de fortunes
`sudo apt-get install fortunes fortunes-fr`
Puis test de fortunes avec la commande `fortune`
Linux permet d'enchaîner des instructions, en envoyant le résultat d'une commande dans une autre. On appelle cela un tuyau (pipe en anglais). Essayer : `fortune /usr/share/games/fortunes/fr | cowsay -n`
Pour avoir des informations sur son système, le programme inxi est très complet (`sudo apt-get install inxi`). Plein d'options possibles.

5 Exercices

Exercice n° 1.

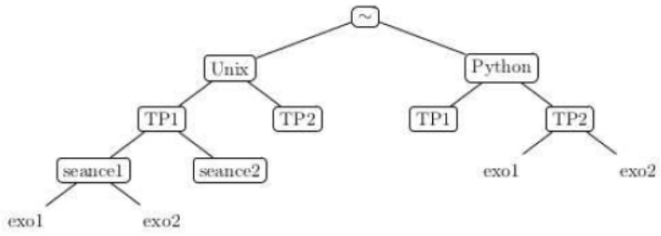
Créer un dossier programmes, et dedans un fichier `bonjour.py`.
Le code du script Python peut être (ce n'est pas une obligation, vous pouvez faire plus sophistiqué si vous le souhaitez) :

```
1 from time import sleep
2 nom = input("Quel est votre nom ? ")
3 print("Bonjour ", nom)
4 sleep(5)
```

Rendre le fichier exécutable pour vous.
Vérifier quelle est la version de Python dans `/usr/bin`. Vous constaterez qu'un "ls" donne trop de résultats. On utilise alors `ls py*` pour trouver tous les fichier dont le nom commence par py (et est suivi de n'importe quoi d'autre).
Lancer le programme avec la commande : `bonne_version_de_python bonjour.py`

Exercice n° 2.

Créer l'arborescence suivante de dossiers et fichiers dans votre répertoire de travail.



On précisera toutes les commandes utilisées.
A l'aide d'un "pipe", lister à partir du répertoire courant tous les répertoires triés par taille décroissante. On pourra consulter le manuel pour les commandes `du` et `sort`.

Exercice n° 3.

Dans le répertoire `./Unix/TP2`, créer deux fichiers `public.txt` et `prive.txt`.
Donner les commandes permettant de mettre les permissions demandées, quelles que soient les permissions initiales sur les répertoires ou fichiers.

1. Le répertoire personnel possède tous les droites pour l'utilisateur, et uniquement le droit d'exécution pour le groupe et les autres.
2. Les répertoires `Unix` et `Unix/TP2` possèdent tous les droits pour l'utilisateur et les droits de lecture et d'exécution pour le groupe et les autres.
3. Le fichier `public.txt` du répertoire `Unix/TP2` possède les droits de lecture et d'écriture pour l'utilisateur et uniquement les droits de lecture pour le groupe et les autres.
4. Le fichier `prive.txt` du répertoire `Unix/TP2` possède les droits de lecture et d'écriture pour l'utilisateur et aucun droit pour le groupe et les autres.

Exercice n° 4.

Sans tester ces commandes dans un terminal, dessiner ci-dessous l'arborescence finale des fichiers et des répertoires. On utilisera TEST comme racine de l'arborescence.

1. `mkdir a b c d`
2. `touch a/t.txt d/foo.txt`
3. `cd c`
4. `mkdir ../b/e f g`
5. `cd ..`
6. `cp */*.txt c/g` # le caractère spécial "*" remplace n'importe quelle chaîne de caractère
7. `rm -r f d`