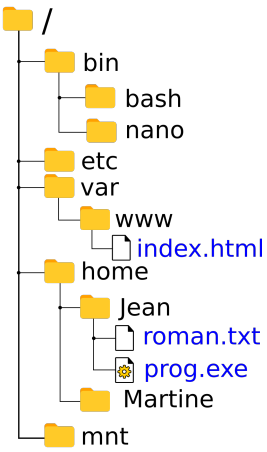
Console Linux

terminal

L'arborescence



Le système d'exploitation *(OS = Operating System)* met à disposition **une racine = root**. Contrairement à Windows, cette racine est **virtuelle**, elle ne correspond pas à un disque particulier (disque dur, clé usb...)

Linux répartit les fichiers en différents dossiers. Quelques exemples :

- `/bin/` contient les fichiers binaires, les exécutables des logiciels.
- `/etc/` contient les fichiers de configuration.
- `/var/` contient les données. Par exemple `/var/www/` contient les pages web que le PC héberge en tant que serveur.
- `home` contient les répertoires utilisateurs. Par exemple `/home/Jean/` contient les fichiers de l'utilisateur Jean.
- `/mnt/` contient des dossiers virtuels qui pointent vers des lecteurs **montés**. Par exemple, si on charge un CD, un dossier apparaîtra dans `/mnt/`

Contrairement à Windows qui fera apparaître un lecteur `D:` dès lors qu'un lecteur CD existe même s'il est vide, avec Linux il faut mettre un CD et demander de créer un lien vers le CD. On dit que l'on **monte le CD**. Aujourd'hui, le plus souvent, le montage et le démontage sont automatique.

Liste - non exhaustive - des commandes terminal

Commande	Effet	Exemple d'utilisation
cd	Changer de répertoire courant	`cd ./sub/sub2` `cd ..` pour remonter
mkdir	Créer un ou plusieurs dossiers	`mkdir rep1 rep2`
cp	Créer une copie	`cp fileSource fileDest`
ls	Lister le contenu d'un répertoire	`ls`
mv	Déplacer ou renommer un fichier	`mv fileSource fileDest`
rm	Supprimer un fichier	`rm monfichier.txt`
rmdir	Supprimer un répertoire	`rmdir rep`
cat	Afficher le contenu d'un fichier	`cat fichier.txt`
echo	Affiche vers la sortie	`echo "Salut !"`
touch	Créer un fichier vide	`touch monfichier.txt`
man	Décrire la commande	`man cd`
tree	Afficher l'arborescence du dossier en cours	`tree`
pwd	Afficher le chemin courant	`pwd`
grep	Recherche une chaîne dans un fichier	`grep chaine *` `grep chaine *.txt`

Terminal

Linux utilise le **terminal**. Malgré son aspect austère, le terminal est extrêmement puissant.

Chemin absolu

Un chemin donné à partir d'une racine est dit absolu.
Exemple : `/home/Jean/roman.txt` Si je suis Jean, `~` est un raccourci pour `/home/Jean` et donc `~/roman.txt` est chemin absolu.

Chemin relatif

Si on a besoin d'un chemin, c'est que l'on est en train d'exécuter un programme. **Exemple :** On exécute `prog.exe`. Ce programme (par défaut) s'exécute dans le `/home/Jean/`. C'est le répertoire courant.

On peur spécifier un chemin par rapport à la position courante. Quelques exemples :

- `roman.txt` : on cherche ce fichier dans le répertoire courant, donc c'est équivalent à `/home/Jean/`
- `.\roman.txt` : `.\` désigne le répertoire en cours, donc c'est équivalent au précédent.
- `..\` : désigne le répertoire parent au répertoire courant, donc `/home/`
- `../Martine/musique.mp3` : depuis le répertoire courant, on remonte au parent puis on descend dans `Martine` et on accède à `musique.mp3`.
- `../../var/www/index.html`

Gestion des flux

Certaines commandes envoient leur résultat vers la **sortie standard**, c'est à dire l'écran. Par exemple `dir`. On peut orienter l'affichage ailleurs, par exemple dans un fichier texte : `dir > fichier.txt` va créer le fichier et le remplir avec le résultat de la commande - S'il existait il est remplacé.

Avec `dir >> fichier.txt` ajoute en fin de fichier s'il existait déjà.

En Linux, on dispose également de *pipe* : `|`. Exemple : `ls | tr 'a-z' 'A-Z' > fichier.txt`
Ici, `ls` liste le contenu du dossier courant, `|` transmet le résultat à la commande suivante qui remplace dans le texte les minuscules par des majuscules. Enfin, le résultat est envoyé dans le fichier texte.

Les jokers

- `*` désigne n'importe quelle suite de caractères
  Par exemple `dir *.exe` affichera tous les fichiers terminant par `.exe`
- `?` désigne n'importe quel caractère exactement une fois.
  Par exemple `dir w?n*` affichera les fichiers avec `w` en premier, un caractère quelconque en deuxième, `n` en troisième et n'importe quelle fin.
- `[ae]` désigne le caractère `a` ou `e`. `ls [ae]*[xy]` affichera tous les fichiers dont le nom commence par a ou e et termine par x ou y.
- L'utilisation de `egrep` donne toute la puissance des **expressions régulières**.

Les droits

On peu dérfinir les droits de lecture `r = 4`, écriture / suppression / ajout `w = 2` et exécution / accès `x = 1`. Si j'ai le droit de lire et exécuter alors j'ai `r-x = 5`
Attention : - ne signifie pas moins !
Un fichier / dossier a des droits associés à plusieurs types d'utilisateurs (dans l'ordre)

u le **propriétaire**, souvent celui qui a créé le fichier.
g les utilisateurs sont placés dans des groupes. Chaque utilisateur peut appartenir à plusieurs groupes. Les fichiers / dossiers sont associés à **un** groupe. Des droits sont liés aux membres de ce groupe.
o *others*, les autres.

Ainsi, le droit `744 = rwx r-- r--` signifie que **u** à tous les droits et les autres n'ont que la lecture.
On ajoute un premier symbole pour distinguer fichier / dossier :

- fichier 744 : `-rwxr--r--`
- dossier 744 : `drwxr--r--`

Commandes :

- `ls -l` liste en affichant les droits
- `chmod g+x fichier` ajoute les droit `x` à `g`
- `chmod 750 fichier` change les droits pour `750 = -rwsr-x---`