Console Linux

~

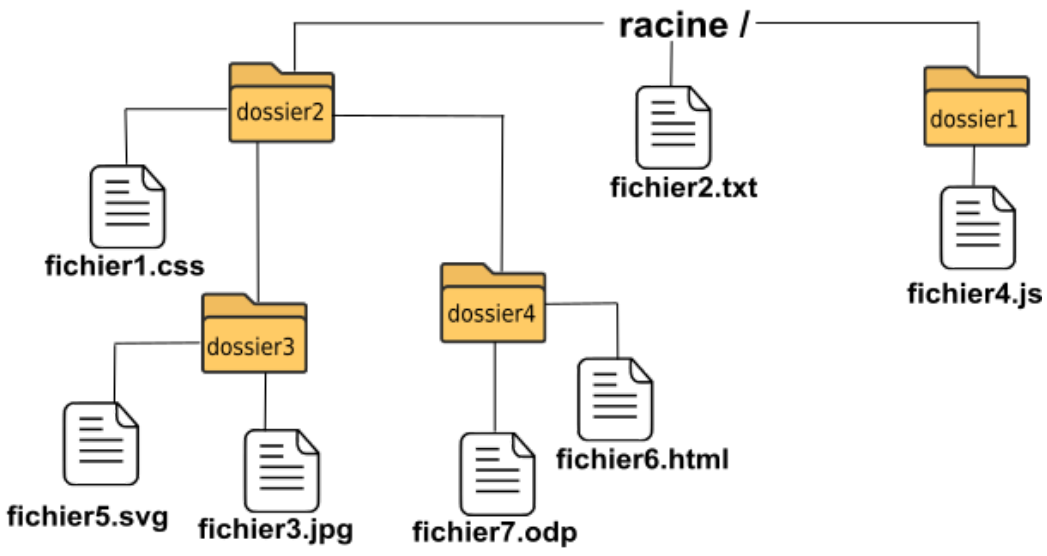
La console, sous Linux, est l'interpréteur de commandes. Il s'agit d'une application faisant partie des composants de base du système d'exploitation. Sa fonction est d'interpréter les commandes qu'un utilisateur tape au clavier dans l'interface en ligne de commande. Généralement, sous Linux cette application est **bash**. Elle permet d'exécuter des commandes de base, mais il s'agit également d'un langage de script permettant d'exécuter des tâches complexes.
Nous nous contenterons ici de voir les commandes de base, ainsi que la gestion des droits et permissions d'accès aux fichiers.
Au préalable, nous rappelons (ou expliquons) la notion d'adresse relative ou absolue qui a pu être vue en SNT en seconde pour les URL.

I. URL, chemin absolu ou relatif

Dans la barre d'adresse de votre navigateur web vous trouvez, quand vous visitez un site, des choses du genre : `https://upload.wikimedia.org/wikipedia/commons/0/05/Url.png`

- La partie `https://` indique le protocole de communication utilisé pour recevoir la ressource.
- La partie `upload.wikimedia.org` contient le nom de domaine du site (wikimedia), le sous-domaine (upload) et l'extension (.org).
- La partie `/wikipedia/commons/0/05/Url.png` est une URL (Uniform Resource Locator). C'est ce qui permet d'identifier une ressource (par exemple un fichier html ou autre, ici un fichier image .png) sur un réseau.

L'URL indique l'endroit où se trouve une ressource sur un ordinateur. Un fichier peut se trouver dans un dossier qui peut lui-même se trouver dans un autre dossier, etc. On parle d'une structure en arborescence, car elle ressemble à un arbre (à l'envers si on met la racine en haut) :



Il existe deux types de chemins (on dit aussi adresses) dans une arborescence : **absolu** et **relatif**.

1. Chemin absolu

Le chemin absolu doit indiquer le chemin depuis la racine.
Par exemple l'URL du fichier fichier3.jpg est : `/dossier2/dossier3/fichier3.jpg`
Remarquez que nous démarrons bien de la racine `/`.

 Les symboles de séparation sont aussi des `/`.

2. Chemin relatif

Imaginons maintenant que le fichier `fichier1.css` fasse appel au fichier `fichier3.jpg` (comme un fichier HTML peut faire appel à un fichier CSS). Il est possible d'indiquer le chemin non pas depuis la racine, mais depuis le dossier (`dossier2`) qui accueille le fichier `fichier1.css`. Nous parlerons alors de chemin relatif.
Dans notre cas, le chemin relatif est le suivant : `dossier3/fichier3.jpg`
Car depuis le `dossier2`, il faut aller dans le `dossier3`, dans lequel se trouve le fichier voulu.
Remarquez l'absence du `/` au début du chemin : c'est ce qui distingue les adresses relatives et absolues.

Imaginons maintenant que nous désirions indiquer le chemin relatif du fichier `fichier1.css` depuis l'intérieur du dossier `dossier1`.
Comment faire ?
Il faut « remonter » d'un niveau dans l'arborescence pour pouvoir repartir vers la bonne « branche ».
Pour ce faire on utilise une notation avec 2 points : `..`
Le chemin relatif est alors le suivant : `../dossier2/fichier1.css`
Il est tout à fait possible de remonter de plusieurs niveau :
`../../` depuis le dossier `dossier4` permet de remonter jusqu'à la racine.

3. En résumé

Chemin absolu :
- débute par `/`
- indique l'adresse depuis la racine

Chemin relatif :
- ne débute pas par `/`
- indique l'adresse depuis la position actuelle (autrement dit depuis le répertoire contenant le fichier dans lequel est donnée l'adresse vers le fichier vers lequel il pointe)
- `../` signifie un niveau plus haut dans l'arborescence

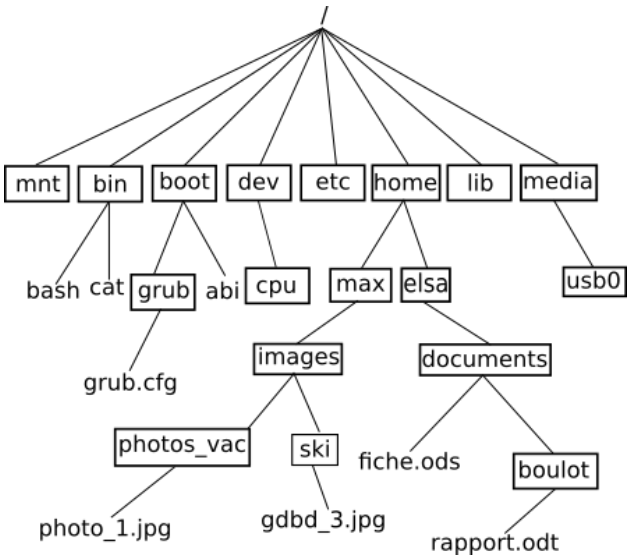
4. Exercices

Exercice 1
On considère encore la structure en arborescence donnée plus haut.

1. Donner le chemin absolu du fichier `fichier5.svg`.
2. Le fichier `fichier1.css` contient une adresse relative pointant vers le fichier `fichier5.svg`. Quelle est cette adresse ?

3. Encore dans le fichier `fichier1.css` se trouve une adresse relative pointant vers le fichier `fichier2.txt`.
   Quelle est cette adresse ?
4. Le fichier `fichier6.html` contient une adresse relative vers le fichier `fichier1.css`.
   Quelle est cette adresse ?
5. Encore dans le fichier `fichier6.html` se trouve une adresse relative pointant vers le fichier `fichier4.js`.
   Quelle est cette adresse ?

Exercice 2
On considère ici l'arborescence qui pourrait être celle d'un système UNIX :



Donner :

1. les chemins absolus pour les fichiers suivants :
   1. `cat`
   2. `fiche.ods`
2. les chemins relatifs pour les fichiers suivants :
   1. À partir de `boulot`, le fichier `fiche.ods`
   2. À partir de `images`, le fichier `gdbd_3.jpg`
   3. À partir de `ski`, le fichier `photo_1.jpg`
   4. À partir de `boulot`, le fichier `abi`

II. Bash

Pour accéder à la console, il suffit généralement de lancer le programme Terminal dont l'icône est une fenêtre noire.

1. Commandes de base

Exercice 3
Tester au fur et à mesure les commandes et exemples ci-dessous :

1. Pour savoir dans quel répertoire de l'arborescence de fichiers on se situe, autrement dit le répertoire courant, on utilise la commande `pwd` (Print Working Directory).
   Généralement en démarrant la console on se situe dans le répertoire personnel, à l'adresse `/home/nomutilisateur`. Ce répertoire est noté sur le prompt (invite de commande) de manière raccourcie avec l'unique caractère « `~` » :

```
user@machine:~$ pwd
/home/user
user@machine:~$
```

2. Pour changer de répertoire, on utilise la commande `cd` (Change Directory).
   - Utilisée sans argument (commande « `cd` »), elle permet de retourner à son répertoire personnel. On peut aussi utiliser la commande « `cd ~` » pour cela.

- La commande « `cd ..` » permet de remonter d'un niveau dans l'arborescence.

On peut utiliser le **chemin absolu**, autrement dit taper l'adresse complète du répertoire à partir de la racine (`/`), par exemple « `cd /home/user/Documents` ».

On peut aussi utiliser le `chemin relatif`, c'est à dire une adresse déterminée à partir du répertoire courant. Pour cela on peut utiliser les « `..` » permettant de remonter d'un répertoire.

Exemple : « `cd ../Documents` » permet d'aller dans le répertoire `/home/user/Documents` si on était dans le répertoire `/home/user/Bureau`.

Le répertoire spécial « `.` » est le répertoire courant.

```
user@machine:~$ cd Documents
user@machine:~/Documents$ cd ~
user@machine:~$ cd Bureau/
user@machine:~/Bureau$ cd ../Documents
user@machine:~/Documents$ cd
user@machine:~$ cd ..
user@machine:/home$ cd .
user@machine:/home$
```

3. Question sur l'arborescence de l'exercice 2 : Vous êtes l'utilisateur **elsa** et votre terminal pointe sur `/etc`. Quel est le moyen le plus simple pour accéder à `fiche.ods` ?
4. Pour avoir un manuel d'utilisation d'une commande donnée, on utilise la commande `man`.

   Par exemple, « `man pwd` » donne :

```
PWD(1) User Commands PWD(1)

NAME
 pwd - print name of current/working directory

SYNOPSIS
 pwd [OPTION]...

DESCRIPTION
 Print the full filename of the current working directory.

 -L, --logical
 use PWD from environment, even if it contains symlinks

 -P, --physical
 avoid all symlinks

 --help display this help and exit

 --version
 output version information and exit

 If no option is specified, -P is assumed.
Manual page pwd(1) line 1 (press h for help or q to quit)
```

   Les touches du clavier bas et haut permettent de se déplacer dans le document.

   On presse, comme indiqué, sur la touche `q` pour quitter le manuel.
5. La commande `ls` (LiSt) permet de lister les fichiers et répertoires du répertoire courant. Elle possède de nombreuses options (à préciser après un tiret si on en utilise). Pour afficher tous les fichiers (y compris les fichiers cachés) et des détails sur les fichiers, on peut par exemple utiliser la commande « `ls -al` » (voir « `man ls` » pour plus d'informations).

```
user@machine:/home$ ls -al
total 28
drwxr-xr-x 4 rootroot 4096 1 juil. 22:23 .
drwxr-xr-x 19 rootroot 4096 7 oct. 07:30 ..
drwxr-xr-x 35 useruser 4096 13 oct. 17:08 user
drwx------ 2 rootroot16384 28 oct. 2020 lost+found
user@machine:/home$
```

6. La commande `mkdir` (MaKe Directory) permet de créer un répertoire.
   Syntaxe : « `mkdir Nom_Repertoire` ».

```
user@machine:/home$ cd ~/Documents
user@machine:~/Documents$ mkdir tmp
user@machine:~/Documents$ cd tmp
user@machine:~/Documents/tmp$
```

7. La commande `touch` permet de créer un fichier vide ou, si le fichier existe, de le « toucher », c'est à dire d'y accéder, ce qui change la date du dernier accès au fichier.
   Exemple, « `touch exo` » crée un fichier exo si celui-ci n'existe pas.

```
user@machine:~/Documents/tmp$ touch exo
user@machine:~/Documents/tmp$ ls
exo
```

8. La commande `cp` (CoPy) permet de copier des fichiers ou des répertoires.
   Exemple : « `cp exo ~/Documents/copieexo` » copie le fichier `exo` dans le répertoire `Documents` du répertoire personnel.
   On peut également donner un nom différent à cette copie dans la commande :

```
user@machine:~/Documents/tmp$ cp exo ~/Documents
user@machine:~/Documents/tmp$ cp exo ~/Documents/copieexo
user@machine:~/Documents/tmp$ ls ~/Documents
copieexo exo
user@machine:~/Documents/tmp$
```

9. La commande `mv` (MoVe) permet de déplacer (en renommant éventuellement) un fichier.
   Exemple :
   « `mv ../exo copie2` » déplace le fichier `exo` dans du répertoire supérieur dans le répertoire actuel en le renommant `copie2`.

```
user@machine:~/Documents/tmp$ mv ../exo copie2
user@machine:~/Documents/tmp$ cd ..
user@machine:~/Documents$ mv copieexo tmp/copie1
user@machine:~/Documents$ ls
user@machine:~/Documents/tmp$ cd tmp
user@machine:~/Documents/tmp$ ls
copie1 copie2 exo
```

10. La commande `rm` (ReMove) est une commande risquée puisqu'elle permet de supprimer un fichier. Celui-ci est totalement supprimé, il ne passe pas par la « corbeille », et selon la configuration de la console, elle ne demande pas de confirmation.
    Exemple : « `rm -R tmp` » supprime le répertoire `tmp` ainsi que (récursivement) son contenu.
    On peut utiliser le symbole `*` pour autoriser le remplacement par n'importe quelle ensemble de caractères.

```
user@machine:~/Documents/tmp$ rm copie*
user@machine:~/Documents/tmp$ ls
exo
user@machine:~/Documents/tmp$ cd ..
user@machine:~/Documents$ rm -R tmp
user@machine:~/Documents$ cd tmp
bash: cd: tmp: Aucun fichier ou dossier de ce type
user@machine:~/Documents/tmp$
```

11. La commande `find` permet de rechercher des fichiers.
    Syntaxe : « `find -name nom_a_trouver` ».

```
user@machine:~/Documents$ cd ..
user@machine:~$ find -name exo
user@machine:~$ find -name share
./.local/share
user@machine:~$
```

    Le répertoire `share` est situé dans le répertoire `.local` (qui est caché car son nom commence par un point).

2. Gestion des droits et permissions

On peut voir, en utilisant la commande « `ls -l` » que pour chaque fichier il y a des informations de la forme « `-rw-r--r--` » ou bien « `drwxr-xr-x` ». Cela indique les droits associés au fichier (si cela commence par '-') ou répertoire (ci cela commence par 'd'), sous forme de trois fois trois caractères. Dans chacun des trois triplets de caractères, on a dans l'ordre, 'r' (read) pour la lecture, 'w' (write) pour l'écriture et 'x' pour l'accès/ouverture (pour un répertoire) ou l'exécution (pour un fichier).

- Le premier triplet correspond aux droits de l'utilisateur possédant le fichier
- Le second triplet correspond aux droits des membres du groupe auquel appartient le fichier
- Le troisième triplet correspond aux droites des autres utilisateurs

Exemple : un fichier ayant les droits « `-rw-r--r--` » donne les droits de lecture à tout le monde, mais les droits d'écriture ne sont donnés qu'à l'utilisateur qui possède le fichier. Le fichier n'est cependant pas exécutable.
Précisons : chaque fichier (et répertoire) est la propriété d'un utilisateur particulier. Par défaut, celui-ci appartient à l'utilisateur qui a créé le fichier. Les utilisateurs sont réunis en groupe. Un utilisateur pouvant faire partie de plusieurs groupes, pour chaque fichier est spécifié le groupe propriétaire (c'est-à-dire en tant que membre de quel groupe le propriétaire détient le fichier). On distingue alors trois catégories d'utilisateurs pour chaque fichier : le propriétaire, le groupe propriétaire, et les autes. Comme nous l'avons indiqué plus haut, les droits du fichier sont donc définis pour chacune de ces catégories.
Avant de manipuler ces droits, indiquons une commande supplémentaire, permettant de connaître notre identité. Il s'agit de la commande « `id` ». Elle donne les numéros et noms d'utilisateur (uid), de groupe (gid) et la liste des groupes auxquels on appartient.

Exercice 4

1. Déterminer vos groupes d'appartenance à l'aide de la commande `id`.
2. Aller dans le répertoire `/home`, utiliser la commande « `ls -l` » et repérer les droits, le propriétaire et le groupe d'appartenance de chacun des fichiers qui s'y trouvent.

Dans la suite, les deux nouvelles commandes suivantes vont être utilisées :

- la commande `chmod` sert à changer les droits (si on en a les droits suffisants).
  La syntaxe générale est « `chmod liste_droits nom_fichier(s)` »,
  où `liste_droits` est par exemple :
  * `u+r` pour rajouter au propriétaire (user) le droit en lecture,
  * `g-w` pour retirer aux membres du groupe (group) le droit en écriture,
  * `o+x` pour donner aux autres utilisateurs (other) le droit en exécution,
  * ou une combinaison de ces possibilités (exemple : `ug-wx` supprime à l'utilisateur et aux membres du groupe les droits en écriture et exécution).
- À l'aide de la commande « `echo "une phrase" > essai` », on écrit le texte `une phrase` dans le fichier `essai`.

Exercice 5

1. Créer, dans le répertoire `~/Documents`, un répertoire `rep`, puis créer un fichier `essai` dans le répertoire `rep`.
   En utilisant la commande indiquée plus haut, écrire la phrase de votre choix dans le fichier `essai`.
2. Notez à l'aide de `ls -l` les permissions actuelles du répertoire `rep` et du fichier `essai`.
3. En utilisant la commande `chmod`, retirez-vous le droit en lecture et en écriture sur le fichier `essai`.
   Vérifier l'effet obtenu en essayant d'afficher le contenu du fichier sur la fenêtre du terminal (avec la commande « `cat essai` » par exemple), puis de remplacer ce contenu par une phrase différente.
4. Un fichier exécutable est simplement un fichier dont on possède le droit en exécution.
   Rétablir le droit en écriture puis remplacer à l'aide de la commande `echo` le contenu du fichier `essai` par le texte « `echo "Ceci est un essai"` ».
   Ajoutez-vous le droit en exécution, et exécuter le fichier essai en tapant « `./essai` dans le terminal (depuis le répertoire qui le contient).
   Quel est le problème ?
5. Rétablir enfin le droit en lecture et tenter à nouveau d'exécuter le fichier.
   Si tout ne se passe pas vraiment comme prévu, comment obtenir un résultat plus intéressant ?
6. Modifier enfin les droits du fichier `essai` afin qu'un membre du groupe propriétaire puisse le modifier, puis que les autres ne puissent pas le lire.

Passons maintenant à la manipulation des droits sur les répertoires :

Exercice 6

1. (a) Se placer dans le répertoire `rep`, et retirez-vous le droit en lecture pour ce répertoire (rappel : le répertoire courant est le répertoire « . »).
   (b) Lister le contenu du répertoire avec `ls`, puis exécuter ou afficher le contenu du fichier `essai`.
   (c) Que peut-on en déduire ?
   (d) Rétablir le droit en lecture sur `rep`.
2. (a) Créer dans `rep` un fichier `nouveau` ainsi qu'un répertoire `sstest`.
   (b) Retirer au fichier `nouveau` et au répertoire `rep` le droit en écriture.
   (c) Tenter de modifier le fichier `nouveau`, puis de le supprimer.
   (d) Rétablir ensuite le droit en écriture au répertoire `rep`.
   (e) Tenter de modifier le fichier `nouveau`, puis de le supprimer.

(f) Que peut-on déduire de toutes ces manipulations ?

3. (a) Se positionner dans le répertoire personnel, puis retirer le droit en exécution du répertoire `rep`.

   (b) Tenter de créer, supprimer, ou modifier un fichier dans le répertoire `rep`, de s'y déplacer, d'en lister le contenu, etc...

   (c) Qu'en déduit-on quant au sens du droit en exécution pour les répertoires ?

3. Pour les plus rapides

Exercice 7

Une chasse au trésor et quelques éléments supplémentaires :

1. Télécharger l'archive `tresor.zip`.
2. Décompresser l'archive et ouvrir un terminal dans le répertoire `activite` obtenu.
3. Exécuter alors la commande : `bash init.sh`
   (On pourra à la fin de l'exercice regarder par curiosité le contenu du fichier `init.sh`)
4. Chasse au trésor :
   (a) Utiliser la commande `find -name` vue précédemment pour trouver où se situent les fichiers `tresor.txt` et `mystere.txt`.
   (b) Afficher le contenu des répertoires qui contiennent ces fichiers (avec une commande `ls`).
5. Dévoiler le secret :
   (a) Faire en sorte de pouvoir lire le contenu du fichier `secret.txt` (à rechercher au préalable).
   (b) Afficher son contenu avec les commandes `cat` puis `more` ou `less`.
6. Commandes `wc`, `grep` et `uniq` :
   (a) Exécuter la commande : `ls | wc`
   En comprendre la signification (éventuellement en utilisant la commande `man`).
   (b) Dans le répertoire contenant `tresor.txt`, exécuter la commande :
   `wc tresor.txt`
   puis la commande :
   `uniq tresor.txt | wc`
   Le fichier `tresor.txt` contient-il des doublons ?
   (c) Dans le répertoire contenant le fichier `mystere.txt`, exécuter la commande suivante :
   `grep France mystere.txt`.
   Comment faire connaître le nombre de villes françaises dans ce fichier ?
   Comment donner la liste de ces villes dans l'ordre alphabétique ?