Partie 1

Gestion des fichiers et systemes d'exploitation

1.1 Pourquoi des fichiers ?

Le processeur et le coeur du système d'exploitation (le noyau) s'associent pour optimiser au mieux l'usage de la mémoire principale grâce aux espaces de mémoire virtuelle, alloués au processus. Cependant cette mémoire est de taille limitée, et ne conserve pas son état lors d'une coupure de l'alimentation. Pour répondre à ces problèmes, le système d'exploitation propose à l'utilisateur et au processus un système de fichiers.

Un fichier est une zone mémoire physique de taille limitée stockant des données de manière persistante.

Cette mémoire dite secondaire, est une mémoire de stockage ou mémoire de masse. Elle peut être un disque dur un CD une carte mémoire SD etc. Ces données stockées sur un tel support doivent pouvoir être *relus* une fois la mémoire RAM éteinte. On peut aussi vouloir *déplacer* un disque et l'utiliser sur un système d'exploitation qui n'a aucune information a priori sur le contenu de ce disque.

Les applications ont justement besoin de fichiers pour stocker leurs programmes et pour sauvegarder des données.

1.2 Système de fichiers, organisation dans le disque physique

On peut distinguer deux types de fichiers :

- les fichiers texte écrit en ascii ou UTF-8. Chaque octet ou chaque mot de 16 bits correspond à un caractère.
- les fichiers binaires. Le fichier n'est qu'une liste de nombres sans sens si on ne connaît pas le format et le logiciel associé. Ce format est annoncé par une *extension*.

Chaque fichier a un nom, dont une extension (obligatoire sous Windows facultatif sous Linux) : txt, docx, html, json, jpg, mp3, ...

Un fichier est une **collection de données numériques qui se partagent entre des données (principales), et des métadonnées (quelques dizaines d'octets)**. Les metadonnées sont la taille, la date de création et/ou de modifications, le propriétaire, les droits d'accès, et l'adresse sur le disque.

Le **système de fichiers** gère la *structuration* le *nommage l'accès la protection et l'implantation* des fichiers, avec les adresses des blocs sur le disque physique.

Plusieurs systèmes de fichiers co-existent (FAT32 et NTFS pour Windows, EXT4 Linux). Ces informations sur le contenu du disque ainsi que sur les fichiers sont rappelés dans un *super bloc*, au début de la *partition*, ce qui permet de déplacer le disque et le lire avec un autre système. Il comporte toutes les informations nécessaires à la compréhension de son système de fichiers.

1.3 Le rôle du système d'exploitation

Le *système d'exploitation* vérifie les *droits d'accès* en lecture/écriture/exécution. Il peut verrouiller tout ou partie d'un fichier, ainsi que l'accès simultané en écriture à un même fichier par deux processus, ce qui pourrait le corrompre. Enfin, il *copie une partie* du fichier, la plus interessante selon le *processus*, sur la *mémoire RAM*. Au final, un processus n'a jamais vraiment accès au fichier lui-même, mais à une représentation, sur la RAM.

Pour faciliter l'accès aux fichiers, les systèmes d'exploitation présentent ceux-ci selon une **arborescence** de dossiers appelés aussi repertoires.

Pour accéder à un fichier, il faut donc indiquer un **chemin**.

Un **dossier** est alors un simple fichier indiquant les fichiers et les dossiers (donc des fichiers), à l'interieur.

L'opération dite de montage revient à associer à un dossier un système de fichiers différents. Cela se fait lorsque l'on connecte par exemple une clé usb.

Les **systèmes d'exploitation graphiques** (MS Windows, Linux Ubuntu, Mac OS, ...) disposent de logiciel d'exploration des fichiers, et associent à chaque type de fichier un pictogramme appelé *icône*.

1.4 Linux

Sous Linux, tout est fichier. Il y a 4 catégories de fichiers :

- *fichiers normaux* (fichiers textes, sources de programmes, fichiers binaires, ...)
- *fichiers répertoires* : des fichiers containers, qui contiennent des références vers d'autres fichiers
- *fichiers speciaux* : situés sous /dev, ce sont les points d'accès vers les peripheriques.
- *les fichiers liens symboliques* : ne contiennent qu'une référence vers un autre fichier. (les raccourcis sous windows) Cela permet d'avoir un même fichier, à differents emplacements, avec des noms différents.

Ainsi, lorsque l'on branche un nouveau disque physique, un chemin d'accès est defini dans l'arborescence du système (par exemple /media/fred)

1.5 Windows

Dans le monde Windows, contrairement aux systèmes Linux, les partitions et périphériques de stockage possèdent leur propre arborescence, dont la racine porte un nom d'une seule lettre : A : C : D : ...

Partie 2

Manipulation des fichiers et dossiers

Les systèmes d'exploitation courants possédent une interface graphique et ont des logiciels d'exploration évolués et intuitifs (Explorateur de fichiers sous Windows, Gestionnaire de fichiers sous Ubuntu, ...). *L'explorateur* permet de :

- **naviguer** aisément dans l'arborescence,
- *déplacer/copier/coller* des fichiers et des dossier, y compris par glisser/déposer,
- *lancer des applications et ouvrir des documents, obtenir des informations avancées sur les fichiers et dossier (*métadonnées*),
- réaliser des *recherches* de fichiers et dossiers,
- créer de *nouveaux documents*,
- ...

Mais ce n'est pas la seule façon de naviguer dans l'arborescence : les systèmes d'exploitation permettent aussi de réaliser ces *fonctions en ligne de commande*, à partir d'un terminal (une fenêtre qui *interface* l'utilisateur avec un *interpreteur*).

Partie 3

Utiliser une console python

Dans votre IDE python, ouvrir un nouveau fichier. Placer l'instruction :

```
1 import os
```

Sauvegarder ce fichier dans votre repertoire de travail. Executer celui-ci. Pour l'IDE, le repertoire courant est alors celui où se trouve votre fichier.

Les fonctions python qui permettent de naviguer entre dossiers et fichiers font partie du module `os.path`, dont la documentation officielle est ici : doc python

On utilisera principalement :

- `os.getcwd()` : str, le dossier courant
- `os.path.abspath(d)` : retourne le chemin absolu du dossier d
- `os.chdir(d)` : change de dossier pour aller au dossier d
- `os.chdir('..')` : remonte d'un niveau
- `os.listdir()` ou `os.listdir(d)` : retourne la liste des fichiers et dossiers du dossier courant ou bien du dossier d
- `os.path.isdir(f)` : booléen True si `f` est un dossier
- `os.path.isfile(f)` : True si f est un fichier
- `f.endswith('.csv')` : True si l'extension de f est .csv

3.1 Chemins

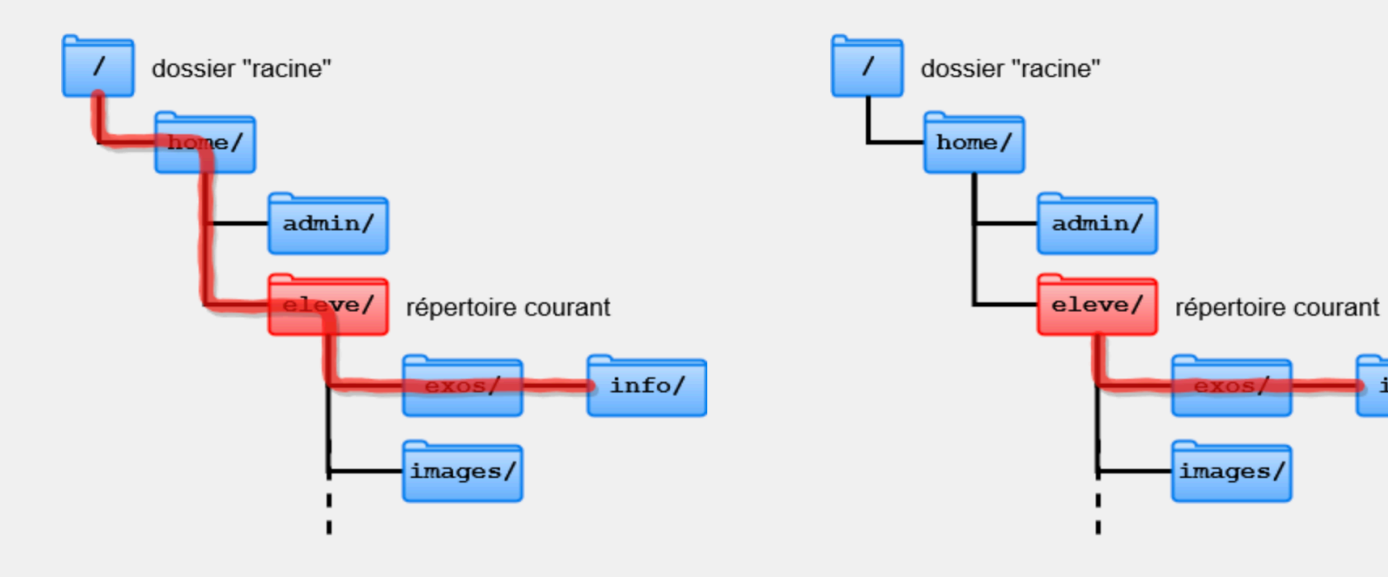

FIGURE 1 – Absolu : '/home/eleve/exos/info/' Relatif : 'exos/info/'

Un chemin est une chaîne de caractères permettant de désigner un Dossier ou un Fichier.

Un chemin peut être absolu ou relatif :

- Absolu (= relatif au dossier racine)

- Relatif (au répertoire courant)

Sous *Windows* le séparateur est le caractère \ alors que sous Linux, il s'agit du caractère /.

En Python les deux notations sont possibles, mais étant donné la signification particulière du caractère \, il faudra en écrire deux : \\ (ce qui correspond en fait à un seul \ dans la chaîne)