Leçon 22 : Stockage et manipulation de données, des fichiers aux bases de données

La mémoire d'un programme meurt avec lui. Néanmoins, on souhaite garder des données plus pérennement. La mémoire d'un ordinateur est alors une série de milliards de bits, parmi lesquels coexistent tout et n'importe quoi. On veut alors les organiser de façon à rendre leur accès le plus simple et rapide possible.

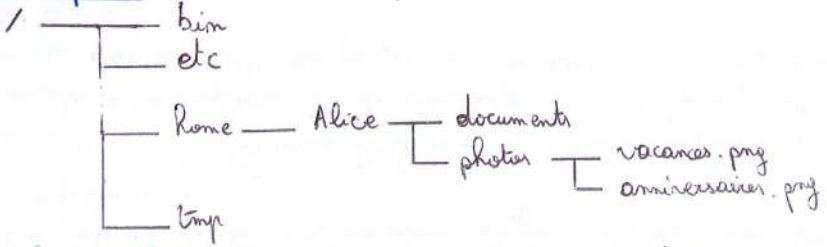
I - Fichiers

I-1) Organisation et manipulation

Définition 1 : Un fichier est un ensemble de données. C'est l'unité de stockage manipulé par l'utilisateur. Ils forment un système de fichier.

Définition 2 : Pour organiser des données de manière persistante sur un disque on utilise une arborescence de fichier. La norme POSIX, suivie par la plupart des OS (Linux, Mac, android) définit cette organisation et sa manipulation.

Exemple 3 : Arborescence de fichier Linux.



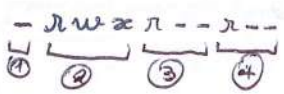
Définition 4 : Le chemin d'accès vers un fichier est soit exprimé de manière absolu (depuis la racine /), soit depuis le répertoire courant (noté .)

Syntaxe 5 : On peut accéder et manipuler l'arborescence de fichier depuis un invité de commande (i.e. shell i.e. terminal) avec les commandes suivantes :

- pwd : affiche le répertoire courant
- cd chemin : change le répertoire courant pour la cible du chemin
- mkdir / touch : créer un dossier / un fichier
- cp / mv / rm : copie / déplace et renomme / supprime.
- ls : liste le contenu d'un répertoire.

TP6 : Prise en main des commandes du terminal en s'appuyant sur le man.

Syntaxe 7 : La commande ls -l permet de faire apparaître des informations supplémentaires sur les fichiers : propriétaire, groupe propriétaire, taille, dernière modification et autorisations ;



① type de fichier : d : répertoire, l : lien symbolique, - : autre

②③④ : r : read, w : write, x : execute

② : permissions propriétaire ③ : permissions groupe ④ : permissions tous.

chmod permet de modifier ces autorisations.

Remarque 8 : Pour Linux « tout est fichier » : les codes sources, les exécutables, mais aussi les dossiers, les périphériques (souris, écran, clavier, ...)

Propriété 9 : Plusieurs système de fichiers (volume dans la mémoire) peuvent cohabiter sur un même ordinateur, avec chacun leur racine.

Exemple 10 : Les C:\, D:\, E:\, etc... sous Windows sont autant de système de fichiers.

Remarque 11 : Sous Linux tous les systèmes de fichiers ont la même racine. Quand on « connecte » un système sur le principal, on dit qu'on le monte.

I-2) Stockage

Il existe plusieurs manières de faire un système de fichiers (FAT32, ext4, ...), définissant des primitives de gestion des fichiers ainsi que des structures pour la gestion des espaces libres

Définition 12 : On découpe alors les fichiers en blocs de quelques Ko. Le système de fichier ne manipulera alors que des blocs.

Définition 13 : Un fichier étant souvent trop grand pour un unique bloc, il est séparé en plusieurs blocs :

- allocation contigüe : les blocs sont contigües en mémoire

↳ accès séquentiel rapide mais fragmentation et difficulté à créer ou étendre des fichiers.

- allocation chaînée : les blocs peuvent être n'importe où, chaque bloc contenant l'adresse du suivant.
↳ bonne utilisation de la mémoire, création, extension facile mais accès séquentiel lent.
On dispose parfois d'une table d'allocations de fichiers.

- l'allocation indexée : les adresses des blocs constituant un même fichier sont rangées dans une table appelée index, elle-même contenue dans un ou plusieurs blocs.

↳ bon accès séquentiel et extension facile, mais taille de fichier maximale et utilisation de mémoire annexe (visible surtout pour les petits fichiers)

Définition 14 : Chaque fichier se voit associé un numéro inode à un emplacement de stockage. L'inode permet de retrouver dans une table du périphérique de stockage des infos données par ls -l.

Définition 15 : Pour économiser de la place, des liens peuvent être créer entre des fichiers avec :
ln : lien physique, l'inode est partagé mais la suppression d'un des fichiers n'impacte pas l'autre.
ln -s : lien symbolique, un nouvel inode est utilisé et le fichier ne contient que les chemins vers sa source.

II- Format

II-1) Fichier texte

Définition 16 : Un fichier texte représente uniquement une suite de caractères (type char en C ou en OCaml) codé en ASCII.

Remarque 17 : C'est le format de fichier basique.

Remarque 18 : Il arrive que l'on veuille représenter plus que les 128 caractères qu'autorise l'ASCII (ou 256 pour l'ASCII étendue). On peut alors utiliser des codages sur plus de bits (16 pour l'unicode).

Définition 19 : Étant le format de fichier le plus basique, il est le plus simple à manipuler. On pourra alors accéder à ces fichiers en langage de programmation.

Fonction	En C	En OCaml
ouvrir un fichier	fopen (chemin, mode)	open_in
fermer un fichier	fclose (fichier)	close_in
écrire dans un fichier	fprintf	input_line
lire dans un fichier	fscanf	output_line

Remarque 20 : Lorsqu'on utilise printf en C, on écrit dans un fichier particulier : la sortie standard (stdout) qui correspond à l'invite de commande.. printf est donc équivalent à fprintf(stdout, ...) De même scanf lit l'entrée standard (stdin).

Définition 21 : Pour rediriger la sortie standard on peut utiliser des commandes :
- commande > filename : la sortie standard de la commande est écrite dans le fichier, qui est écrasé.
- commande >> filename : même chose mais sans écraser le fichier
- commande < fichier : le fichier devient l'entrée standard de la commande
- commande1 | commande2 : la sortie standard de la première commande devient l'entrée standard de la deuxième.

Remarque 22 : L'écriture étant lente, un tampon est utilisé. On peut forcer l'écriture des tampons avec flush en Ocaml et fflush en C.

II-2) Formats de fichiers

Pour représenter plus que des chaînes de caractères, on a besoin de définir des formats de fichiers qui indiquent comment interpréter les bits de données.

Définition 23 : Un format de fichier est une convention de représentation de données.

Remarque 24 : Pour gagner de l'espace, ces formats utilisent souvent des méthodes de compression, avec ou sans perte.

Exemple 25 : Quelques formats particuliers :
- Le format png stocke et compresse les images sans pertes
- Le format jpeg stocke des images compressées avec pertes
- Le format MP3 stocke des sons compressés avec pertes
- Le format MP4 combine audio et vidéo (avec ou sans pertes)
- Le format zip compresse sans pertes des fichiers quelconques.

Remarque 26 : Le format zip utilise l'algorithme de compression LZW, mais aussi le codage de Huffman.

Développement 1 : Présentation de l'algorithme LZW

Remarque 27 : Il y a toujours un compromis à faire entre compression et facilité d'utilisation. Ainsi, la plupart des formats se spécialisent dans une utilisation.
- Quand on ouvre une image en python, on la transforme en tableau de triplets, quand on la sauvegarde on la recompresse, dans un format moins manipulable.
- Quand on édite une vidéo, on doit l'exporter à la fin pour passer d'un format manipulable à un format pour la lecture
- Quand on transforme des fichiers en .zip, on ne peut pas les modifier ou les lire, il faut donc les réexporter, mais cela est plus compact.

Exemple 28 : Le format CSV (pour comma separated values) est un format de fichier texte permettant de stocker des données sous forme de table, permettant naturellement la suppression, l'ajout, etc...
Pour des commandes : {produit : tomate, prise : 3, quantité : 50, client : Le navet navigant, adresse : 13 rue du charisme Tarbes}, {produit : patate, prise : 1, quantité 30, client : Le navet navigant, adresse : 13 rue du charisme à Tarbes}, ...

Pourrait-on faire mieux ?

III - Bases de données

Souvent les données d'une table ont des redondances, et des liens entre elles (cf. exemple 28). Pour manipuler de gros volumes de données on ne se contente alors plus de fichiers en texte brut.

Définition 29 : Le modèle relationnel est une manière de représenter les données en exploitant les relations entre elles.

Exemple 30 : Un grossiste gérant des commandes.

Produit (num_produit, nom, prise, poids)
Clients (num_client, nom, adresse, ville)
Commande (# num_produit, # num_client, quantite)

attributs

num_prod	nom	prise	poids
1	patate	1	1
2	canard	8	0,4
3	Raricot	4	2

Table produit

unique ↳ clé primaire

num_client	nom	adresse	ville
1	Radis radieuse	3 rue A	Tarbes
2	Navet navigant	12 rue B	Chartres

Tables clients

Tables

clé étrangère

num_client	num_prod	quantité
1	1	3
1	2	150
3	1	2

Table commandes

↳ que des entiers ↳ contraintes de domaines.

Définition 31 : Un SGBD (système de gestion de bases de données) est utilisé pour manipuler des données relationnelles. On l'utilise à travers le langage SQL.

Exemple 32 : Le SQL permet de sélectionner certaines données, de les trier, de les sélectionner suivant des conditions, etc...

Théorème 33 : (Codd) SQL est très expressifs.

TD 34 : Trouver toutes les manières de calculer le max et la division.

Développement 2 : Correction du TD 34