Un ordinateur

Un ordinateur est une machine à calculer composée de :

– un processeur (ou unité centrale) qui effectue les calculs

– une mémoire qui conserve les données et les résultats des calculs

– des périphériques qui permettent à l'utilisateur de communiquer avec la machine.

Il existe différents types de périphériques :

–Les périphériques d'entrée qui permettent de transmettre une information de l'utilisateur vers la machine : clavier, souris, crayon optique, …

–Les périphériques de sortie qui permettent à la machine de transmettre une information vers l'utilisateur : écran, imprimante, enceintes, ...

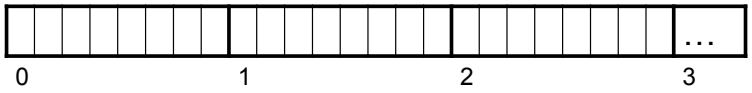
–Les périphériques d'entrée-sortie qui peuvent transmettre une information dans les deux sens : lecteurs de disquettes, disque dur, lecteurs/graveurs cdrom …

La mémoire

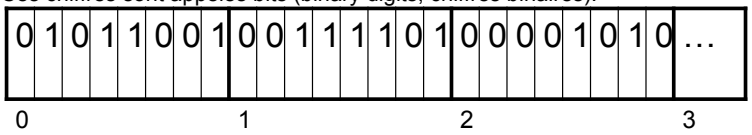
La mémoire peut être vue comme une simple suite de cases contiguës.

Le numéro d'une case dans cette suite est appelé son adresse.

Chaque case (mot) peut contenir une valeur entière écrite sur un nombre fixe de chiffres (mettons 8).



Contrairement aux humains qui écrivent les nombres avec 10 chiffres différents (de 0 à 9), les ordinateurs les écrivent avec seulement deux chiffres (le 0 et le 1). Ces chiffres sont appelés bits (binary digits, chiffres binaires).



Le bit

En numérotation binaire le bit est la plus petite unité d'information manipulable par une machine numérique.

Avec un bit il est ainsi possible d'obtenir deux états (2):

0 - 1

Avec 2 bits il est possible d'obtenir quatre états (2*2):

0 0 - 0 1- 1 0 - 1 1

Avec 3 bits il est possible d'obtenir huit états (2*2*2):

0 0 0 - 0 0 1 - 0 1 0 - 0 1 1

1 0 0 - 1 0 1 - 1 1 0 - 1 1 1

Pour un groupe de n bits, il est possible de représenter 2^n états (valeurs).

L'octet

L'octet est une unité d'information composée de 8 bits. Il permet de stocker plusieurs types de données :

– un caractère

– une lettre

– un nombre

– …

Ce regroupement de bits par série de 8 permet une lisibilité plus grande, au même titre que l'on apprécie, en base décimale, de regrouper les nombres par trois pour pouvoir distinguer les milliers. Par exemple le nombre 1 256 245 est plus lisible que 1256245.

Codage

Afin de pouvoir stocker en mémoire des nombres entiers, des caractères, ou tout autre type, il faut établir une correspondance (un codage) entre chaque nombre binaire et chaque donnée.

Par exemple, pour un codage des nombres entiers sur 2 bits, nous pouvons utiliser la correspondance suivante :

00 code le 0
01 code le 1
10 code le 2
11 code le 3

Ce codage en base 2 (ou codage binaire) des entiers est utilisé par l'ordinateur pour effectuer tous les calculs.

Codage

Dans le codage que nous venons d'utiliser chaque bit à un poids. Ce poids dépend de la position du bit en partant de la droite.

Comme pour les dizaines, les centaines et les milliers pour un nombre décimal, le poids d'un bit croît d'une puissance de deux en allant de la droite vers la gauche.

$01 = 0 * 2^1 + 1 * 2^0 = 0 * 2 + 1 * 1 = 1$
$10 = 1 * 2^1 + 0 * 2^0 = 1 * 2 + 0 * 1 = 2$
$0101 = 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 0 * 8 + 1 * 4 + 0 * 2 + 1 * 1 = 5$
$1101 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 1 * 8 + 1 * 4 + 0 * 2 + 1 * 1 = 13$

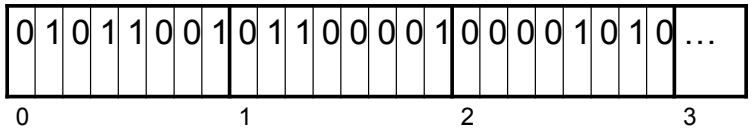
Codage

De la même manière que l'on code des nombres, il est possible de coder des caractères. Sur un octet, on utilise le code ASCII (American Standard Code for Information Interchange). Voici une partie du codage :

Caractère	Nombre entier	Nombre binaire
Y	89	1011001
Z	90	1011010
[	91	1011011
\	92	1011100
]	93	1011101
^	94	1011110
_	95	1011111
`	96	1100000
a	97	1100001
b	98	1100010
c	99	1100011

Codage / mémoire / adresse

Pour résumer, la mémoire est une suite de cases ou mots (d'un octet par exemple) et chacun de ces mots possède une adresse. Chacun de ces mots peut coder un nombre, un caractère, …



Dans cet exemple, si **le programme qui utilise la mémoire** stocke un entier à l'adresse 2 et un caractère à l'adresse 1, alors on peut voir que le nombre 10 est stocké à l'adresse 2 et que le caractère 'a' est stocké à l'adresse 1.

C'est dire que le programme **interprète** le contenu de la mémoire.

La capacité d'un programme à accéder à un mot d'une case mémoire directement par son adresse donne son nom à la RAM : Random Access Memory.

KiloOctets, MégaOctets

– Un Kilo-octet (ko) = 2^{10} octets = 1024 octets

– Un Méga-octet (Mo) = 2^{20} octets = 1024 Ko = 1 048 576 octets

– Un Giga-octet (Go) = 2^{30} octets = 1024 Mo = 1 073 741 824 octets

– Un Téra-octet (To) = 2^{40} octets = 1024 Go = 1 099 511 627 776 octets

Capacité de quelques mémoires informatiques usuelles

- disquette : 1,44 Mo
- mémoire vive (RAM) : quelques Go
- disques durs : quelques centaines de Go
- CD-audio standards : 650 Mo
- DVD (simple face, simple couche) : 4,7 Go

Notion de programme 'binaire'

Un programme peut être vu comme une suite de mots stockés consécutivement en mémoire. Cette suite est appelée « un binaire ». Ces mots codent des instructions élémentaires reconnues et exécutables par le processeur.

Un mot binaire de la mémoire peut donc représenter un entier, un caractère, d'autres types de données, mais également **une instruction**. Par exemple :

– recopier une valeur entière stockée dans un endroit (adresse) de la mémoire vers un autre endroit (une autre adresse) de la mémoire
– opérations arithmétiques :
  • additionner 2 entiers
  • soustraire 2 entiers
  • multiplier 2 entiers
– comparer 2 entiers (égalité, inférieur, inférieur ou égal)
– afficher un entier à l'écran
– passer à l'instruction suivante si une condition est vérifiée
– …

Un petit programme

Pour effectuer des opérations, le processeur est muni d'un petit nombre de registres qui, comme la mémoire, contiennent des mots. Ces registres de l'ALU (unité logique et arithmétique) ont pour rôle de stocker les opérandes et le résultat des opérations arithmétiques.

Par exemple, si on veut additionner un nombre situé à l'adresse A et un nombre situé à l'adresse B puis mettre le résultat à l'adresse C, on ne peut pas directement additionner, il faut procéder en plusieurs étapes :

– Transférer le nombre situé à l'adresse A dans le registre 1 de l'ALU
– Transférer le nombre situé à l'adresse B dans le registre 2 de l'ALU
– Additionner les registres 1 et 2 de l'ALU puis mettre le résultat dans le registre 3
– Transférer le nombre du registre 3 de l'ALU vers l'adresse C

Assembleur et langages évolués

Pour écrire ce programme, on peut :

– Utiliser un langage de programmation de bas niveau comme le langage d'assemblage qui permet de manipuler directement les registres de l'ALU, mais les instructions sont élémentaires (et dépendent de l'architecture de la machine, nombres et noms des registres, variantes du jeu d'instruction).

L'assembleur se contente alors d'effectuer le codage de chaque instruction texte en séquences d'entiers.

Trans A R1, Trans B R2, ADD R1 R2 R3, Trans R3 C

– Utiliser un langage de programmation évolué comme le langage C. Les langages évolués permettent au programmeur d'écrire directement des expressions complexes. On ne manipule plus les registres, et le texte est alors relativement indépendant de l'architecture.

c = a + b;

Le texte des instructions est compilé par un compilateur, c'est à dire traduit en une série de codes binaires.

Système d'Exploitation

Au niveau le plus bas, les périphériques (disque dur, clavier, écran, imprimante etc.) ne comprennent que des ordres de type assembleur. Par exemple :

- positionner la tête du bras du disque dur sur telle piste de tel secteur
- écrire telle séquence
- …

Le système d'exploitation est le programme chargé de fournir à l'utilisateur (et au programmeur) une appréhension abstraite des mécanismes matériels de la machine et de ses périphériques.

Par exemple, les données sur disque ne sont plus vues comme des données tronçonnées en secteurs et en pistes, mais en systèmes de fichiers avec des noms, que l'on peut grouper dans des répertoires, avec des droits d'accès pour certains utilisateurs.

Les programmes sont vus comme des processus qui se partagent les ressources du processeur et ne peuvent pas pénétrer dans les zones mémoires réservées par d'autres processus (sauf par malveillance ou accident).

Arborescence des fichiers

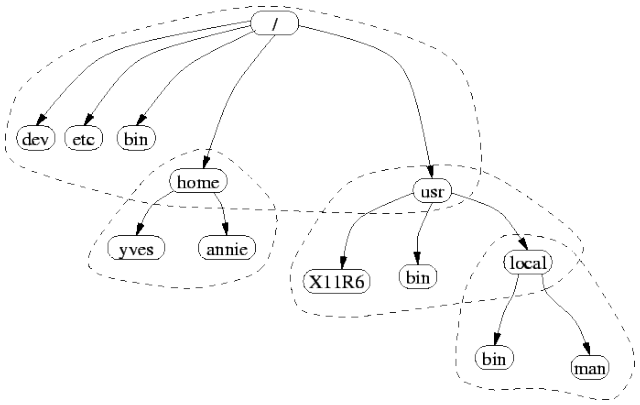
En salles de TP, nous utiliserons le système Unix.

Unix organise les données stockées sur les disques en fichiers (files) et répertoires (directories). Les fichiers contiennent les données (images, textes, programmes, code partagé . . .) et les répertoires permettent de les classer. Le classement prend pour analogie un arbre généalogique de paternité.

Un répertoire peut contenir plusieurs fichiers et plusieurs répertoires. On dit que les fichiers et répertoires contenus dans un répertoire P sont les fils de P. Si un répertoire P contient un fichier ou un répertoire F on dit également que P est le père de F. Dans un répertoire, tous les fils doivent avoir des noms différents.

Il existe un répertoire qui contient tous les autres fichiers et répertoires : il est appelé répertoire racine (root) de l'arborescence des fichiers. Par convention la racine est son propre père. Chaque fichier ou répertoire possède un nom (une suite de caractères aussi longue que l'on veut). Par convention la racine se nomme « / ».

Arborescence des fichiers



Arborescence des fichiers

On voit que deux fichiers/répertoires de père différents peuvent avoir le même nom, comme ici les répertoires bin.

Pour désigner un fichier/répertoire de façon unique, on utilise son chemin (path), c'est-à-dire son nom précédé de toute la série de ses ancêtres depuis la racine.

Chaque nom est séparé par autant de slash (/) que l'on veut. Ainsi, on distingue par exemple les différents répertoires bin :

- /bin
- /usr/bin
- /usr/local/bin
- …

Identification

Pour pouvoir utiliser un ordinateur sous Unix, il faut se loguer, c'est à dire s'identifier comme une personne autorisée par les administrateurs du système à utiliser la machine.

Se loguer consiste à saisir son login (identifiant utilisateur) au clavier ainsi que son mot de passe secret (à frapper à l'aveugle).

Pour vous en TP, le login est l'initiale de votre nom de famille (en minuscule) suivie du numéro de votre carte d'étudiant. Le mot de passe initial est le numéro INE, qui vous a été donné lors de votre inscription (la première chose à faire sera d'ailleurs de changer ce mot de passe en TP).

Ces informations se trouvent sur vos certificats de scolarité.

Interpréteur de commandes

Lorsqu'un utilisateur logué ouvre une fenêtre de terminal de commandes, 'xterm' par exemple, celle-ci apparaît avec une invite de commande : le prompt. C'est un signe qui indique que le terminal est prêt à saisir une commande.

Ce prompt est souvent le signe dollar '$'. Le curseur texte clignote juste à droite, prêt pour la saisie d'une commande au clavier.

```
karim@12806543287 ~ $
```

La fenêtre de terminal de commandes est aussi appelée 'console'

La commande pwd

L'utilisateur se situe toujours à un endroit dans l'arborescence des fichiers : le répertoire courant (working directory). On peut demander quel est ce répertoire avec la commande pwd (print working directory).

Pour cela :

- saisir la commande au clavier.
- puis valider avec la touche Entrée (Return, Enter).

Au lancement, cet endroit est généralement la racine du compte de l'utilisateur : un répertoire où il peut stocker ce qu'il veut (ci-dessous : /home/karim).

Après la validation de la commande le résultat de la commande est affiché, puis la console est de nouveau en attente d'une autre commande.

```
karim@12806543287 ~ $ pwd
/home/karim
karim@12806543287 ~ $
```


La commande ls

La commande ls (list) permet d'afficher le contenu d'un répertoire.

Les arguments de ls sont les répertoires/fichiers à lister.

S'il n'a pas d'argument, ls affiche le contenu du répertoire courant.

```
karim@12806543287 ~ $ ls
salut.c  tp
karim@12806543287 ~ $
```

Ici, le résultat de la commande indique que le répertoire courant est composé de deux fils : 'salut.c' et 'tp'.

Pour savoir s'il s'agit de répertoires ou de fichiers il faut utiliser l'option –F :

```
karim@12806543287 ~ $ ls  -F
salut.c  tp/
karim@12806543287 ~ $
```

On voit que 'salut.c' est un fichier et que 'tp' est un répertoire (termine par un '/')

Interpréteur de commandes : chemin absolu & chemin relatif

Comme il n'est pas pratique de désigner les fichiers en indiquant toujours leur chemin depuis la racine (chemin absolu), on utilise la notion de chemin relatif au répertoire courant.

Les fils immédiats sont accessibles immédiatement par leur nom, et le père est accessible par deux points '..'

```
karim@12806543287 ~ $ pwd                  -> affichage du répertoire courant
/home/karim
karim@12806543287 ~ $ ls tp                -> ls du répertoire tp (chemin relatif)
tp1.c   tp2.c
karim@12806543287 ~ $ ls /home/karim/tp    -> ls du répertoire tp (ch. absolu)
tp1.c   tp2.c

karim@12806543287 ~ $ ls ../               -> ls du répertoire home (ch. relatif)
karim  tmp
karim@12806543287 ~ $ ls /home/            -> ls du répertoire home (ch. absolu)
karim  tmp
karim@12806543287 ~ $
```


La commande cd

On peut se déplacer d'un répertoire vers un autre avec la commande cd (change directory). L'argument est le chemin du répertoire de destination.

```
karim@12806543287 ~ $ pwd                  -> affichage du répertoire courant
/home/karim
karim@12806543287 ~ $ cd tp                -> aller dans le répertoire /home/karim/tp
                                              (chemin relatif)
karim@12806543287 ~ /tp $ pwd              -> affichage du répertoire courant
/home/karim/tp

karim@12806543287 ~ /tp $ cd ..            -> remonter dans le rép. /home/karim
                                              (chemin relatif)
karim@12806543287 ~ $ pwd                  -> affichage du répertoire courant
/home/karim

karim@12806543287 ~ $ cd /home/karim/tp    -> aller dans le rép. /home/karim/tp
                                              (chemin absolu)

karim@12806543287 ~ /tp  $ pwd             -> affichage du répertoire courant
/home/karim/tp
```


Les commandes mkdir & rm & man

On peut créer un répertoire à l'aide de la commande **mkdir** (make directory).

Les arguments de mkdir sont les répertoires à créer.

```
$ ls -F
tp1.c  test.c
$ mkdir INTRO_INFO
$ ls -F
tp1.c  test.c  INTRO_INFO/
```

La commande **rm** permet de supprimer un fichier ou un répertoire. Il faut utiliser l'option **–r** pour un répertoire.

```
$ rm   test.c
$ rm   -r   INTRO_INFO
$ ls
tp1.c
```

Il existe de nombreuses autres commandes. Pour chacune, il existe souvent plusieurs manières de l'utiliser (plusieurs options, plusieurs arguments,…). Pour obtenir le manuel d'utilisation de la commande (s'il existe), il suffit d'utiliser la commande **man** avec en argument le nom de la commande.

Par exemple : **`man ls.`**

Quotas en mémoire

Le système d'exploitation utilise un système de quotas fixant les limites de stockage qu'un utilisateur ne doit pas dépasser.

Deux limites : soft et hard.

Lorsque la limite soft est dépassée, l'utilisateur est prévenu qu'il doit effacer une certaine quantité de données avant un certain temps, sinon des sanctions seront prises (accès au compte supprimé).