Cours SNT Seconde 1 2022-2023

Augustin WENGER

22 septembre 2022

Table des matières

Chapitre 1

Codage d'informations sur un ordinateur

Ce chapitre d'introduction présente rapidement les différentes manières de stocker des données sur un ordinateur.

1.1 Structure d'un disque dur

Un disque dur d'ordinateur est constitué d'un grand nombre de cellules magnétiques dont chacune contient une information binaire : si une cellule est aimantée dans un certain sens, on considère qu'elle contient un 1, sinon on considère qu'elle contient un 0.

Un disque dur dispose de têtes de lecture, qui sont capables de mesurer la charge magnétique d'une cellule pour voir si elle contient un 1 ou un 0, ainsi que de têtes d'écriture, qui sont capables de modifier la charge magnétique d'une cellule en fonction des informations à inscrire dans le disque dur.

1.1.1 Combien d'informations peut-on stocker dans un disque dur ?

Un disque dur est donc comme un grand livre, dont chaque caractère serait constitué uniquement des lettres "1" et "0". On se demande combien il y a de manières d'écrire ce livre en fonction du nombre de cellules du disque dur.

Activité 1

Vous disposez de 3 jetons qui sont blanc d'un côté et noir de l'autre. Pour transmettre un message à un camarade qui est sorti de la pièce, vous devez les placer dans 3 cases, sur la face de votre choix. Combien de messages différents pouvez vous envoyer ?

Prenons l'exemple d'un disque dur ayant 3 cellules. Il y a exactement $2^3 = 8$ mots de 3 lettres constitués exclusivement des lettres "0" et "1" :

000	100
001	101
010	110
011	111

Pour chacune des cellules, il y a 2 possibilités différentes. Plus généralement, s'il y a n cellules le nombre de messages différents qui peut être stocké dans un ordinateur est donc de 2^n.

On voit que le nombre de cellules est le paramètre qui permet de déterminer le nombre de messages différents qui peuvent être codés. On appelle le nombre de cellules le nombre de **bits** du disque dur.

1.1.2 Comment parle-t-on de la quantité de mémoire d'un ordinateur ?

Définition 2

Pour décrire la mémoire d'un ordinateur, on parle plus souvent en **octets** qu'en bits. Un octet représente 8 cellules. On utilise donc la formule de conversion suivante :

1 octet = 8 bits

De plus, il y a énormément de cellules dans un ordinateur. Pour compter le nombre d'octets, on utilise les préfixes classiques multiplicateurs :

préfixe	quantité	puissance de 10
kilo-	1 000	10^3
mega-	1 000 000	10^6
giga-	1 000 000 000	10^9
tera-	1 000 000 000 000	10^{12}
peta-	1 000 000 000 000 000	10^{15}

Et l'on obtient le tableau suivant

nom	symbole	nombre d'octets	nombre de bits
kilooctet	Ko	1 000 octets	8 000 bits
megaoctet	Mo	1 000 000 octets	8 000 000 000bits
gigaoctet	Go	1 000 000 000 octets	8 000 000 000 bits
teraoctet	To	1 000 000 000 000 octets	8 000 000 000 000 bits
petaoctet	Po	1 000 000 000 000 000 octets	8 000 000 000 000 000 bits

Par exemple, un disque dur de 50 Go fait 50 Gigaoctets, donc 50 milliard d'octets donc 400 milliards de cellules.

1.2 Stocker des informations

On dispose maintenant d'un disque dur qui sait stocker des grandes quantités d'informations et y accéder. Une autre étape nécessaire est celle de coder les informations sous la forme binaire.

La manière dont toute information est codée dépend du format du fichier. On va voir quelques exemples :

1.3 Stocker du texte

Pour stocker du texte dans un disque dur, il faut inventer un codage : un algorithme de transformation qui permette de transformer un texte en une série de 0 et de 1, et inversement.

Définition 3

Un algorithme est une suite d'instructions détaillées qui, si elles sont correctement exécutées, conduit à un résultat donné.

Le côté inversible de l'opération est très important : pour pouvoir récupérer les informations présentes dans le disque dur, il faut que l'on puisse repasser de l'écriture binaire à une écriture sous la forme de texte.

Essayons de fabriquer un dictionnaire qui code chaque lettre sous la forme d'un mot constitué de 0 et de 1.

1.3.1 Une méthode qui ne marche pas : utiliser le code Morse

Une approche naïve consisterait à prendre un alphabet existant qui ne contient que deux lettres différentes, par exemple l'alphabet morse (cf Figure 1.1)

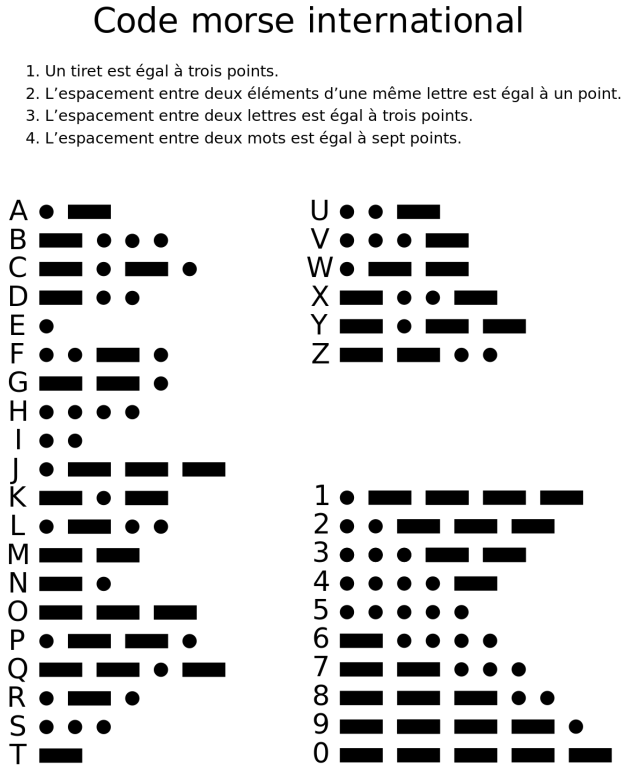

FIGURE 1.1 – L'alphabet morse

On pourrait rêver de transformer cet alphabet en binaire, par exemple en remplaçant les points par des 0 et les traits par des 1. Par exemple, pour coder le mot CHIEN :

Mot	C	H	I	E	N
Morse	-.-.	....	..	.	-.
Binaire	1010	0000	00	0	10
Codage final	1010000000010				

Le mot CHIEN s'écrirait donc dans l'ordinateur comme 1010000000010

Mais on se rend compte que cela ne suffira pas : Quand on communique en morse, on laisse des temps plus longs entre les lettres, mais on ne pourra pas faire la même chose en binaire. par exemple, si on essaye de traduire les mots BASE et DITES, on s'aperçoit qu'ils s'écriraient pareil.

Mot	D	I	T	E	S
Morse	-..	..	-	.	...
Binaire	100	00	1	0	000
Codage final	1000010000				

Mot	B	A	S	E
Morse	-...	.-	...	.
Binaire	1000	01	000	0
Codage final	1000010000			

Le problème, c'est qu'on a bien une liste de lettres qui sont uniquement identifiées, mais comme on ne sait pas quand on a fini d'écrire une lettre, on ne peut pas décoder le code obtenu.

1.3.2 Plusieurs moyens de résoudre ce problème

On peut essayer d'imaginer plusieurs techniques pour passer outre ce problème :

— Faire en sorte que chaque lettre possède le même nombre de caractères binaires. Par exemple, si une lettre est codée sur 8 bits (un octet), on sépare les données en groupes de 8 bits et l'on récupèrera les lettres. Par exemple, pour un alphabet à 4 lettres, on aurait (en codant chaque lettre sur 2 bits) :

Lettre	Code
A	00
B	01
C	10
D	11

— On peut également réserver certaines combinaisons de caractères pour signaler les débuts et fin de lettre. Dans ce cas, On ne peut plus utiliser ces combinaisons à un autre moment. Par exemple, on utilise les 0 pour signaler les débuts et fins de mots :

Lettre	Code
A	00
B	010
C	0110
D	01110

— On peut également utiliser le début de chaque écriture pour dire quand elle va se terminer. Ici, on ajoute au code morse un code qui indique le nombre de bits utilisés pour écrire le mot.

Lettre	Préfixe	Morse	Mot complet
A	01	01	0101
B	11	1000	111000
C	11	1010	111010
D	10	100	10100

1.3.3 Et dans la vraie vie ?

Dans les vrais ordinateurs, c'est une combinaison des deux méthodes qui est utilisée :
— l'ASCII définit 128 caractères, chacun d'entre eux étant codé sur 7 bits.
— Certaines conventions, comme l'ISO 8859-1, étendent l'ASCII en codant sur 8 bits, ce qui permet de rajouter des caractères (c'est une manière de rajouter les accents)
— D'autres conventions, comme l'UTF-8, codent encore plus de caractères, en utilisant entre 1 (pour les caractères standard) et 4 octets. Les premiers bits de chaque lettre indiquent la longueur de la phrase comme le montre la figure 1.2

Définition du nombre d'octets utilisés dans le codage (attention ce tableau de principe contient des séquences invalides)

Caractères codés	Représentation binaire UTF-8	Premier octet valide (hexadécimal)	Signification
U+0000 à U+007F	0bbb·bbbb	00 à 7F	1 octet, codant jusqu'à 7 bits
U+0080 à U+07FF	110b·bbbb 10bb·bbbb	C2 à DF	2 octets, codant jusqu'à 11 bits
U+0800 à U+FFFF	1110·bbbb 10bb·bbbb 10bb·bbbb	E0 à EF	3 octets, codant jusqu'à 16 bits
U+10000 à U+10FFFF	1111·0bbb 10bb·bbbb 10bb·bbbb 10bb·bbbb	F0 à F3	4 octets, codant jusqu'à 21 bits
	1111·0100 1000·bbbb 10bb·bbbb 10bb·bbbb	F4	

FIGURE 1.2 – Le codage de l'UTF-8 selon wikipedia