# Encodage des données

## Introduction

Vous le savez peut-être déjà mais un ordinateur ce n'est pas très intelligent. Quand votre ordinateur affiche des vidéos, du texte, joue du son, ou fait tourner un jeu il n'utilise en réalité que des nombres. Et de plus il ne sait même pas compter par dix, il ne le fait que par deux ! Il travaille donc en ce qu'on appelle des informations *binaires*. La seule chose qu'il sache faire avec ces nombres c'est faire des calculs : en somme un ordinateur c'est une grosse calculette.

Pour vous permettre de mieux vous représenter la situation sachez que toute mémoire d'un ordinateur (que ce soit la « RAM » c'est-à-dire la mémoire vive, ou celle de votre HDD, disque dur, ou encore votre SSD) n'est composée que de millions voir milliards de composants électroniques qui peuvent garder des charges électriques. C'est en représentant ces charges que nous sommes arrivés à la notion des « 1 » et « 0 », notre binaire, car à la base votre ordinateur ne contient pas de chiffres, c'est simplement une convention. Donc un composant gardant une charge électrique est représentée avec le chiffre « 1 » et un composant sans charge électrique l'est avec un « 0 ».

Or avec deux possibilités on ne va pas très loin. C'est pourquoi on a décidé de les regrouper par huit. Un seul chiffre binaire porte le nom de « bit » et un groupe de huit est ce qu'on appelle un « byte » (attention à ne pas confondre les deux !) ou « octet » en Français, et c'est d'eux qu'on parle lorsqu'on fait référence aux paquets durant les requêtes internet. Si avec un bit on a 2 possibilités, combien nous en donnent 8 bits ?

$2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8 = 256$

Attention, il y a 256 possibilités mais on ne peut représenter que les chiffres jusqu'à 255 car on doit aussi compter le 0 !

On a donc 256 possibilités à partir de un byte, ou huit bits. Pour ceux qui se sont demandés « Et si je veux 972 ? », pas de stress ! Dans ce cas on peut alors faire des groupes de plusieurs bytes. Sachant qu'un octet à 256 valeurs différentes possibles on peut calculer que deux octets ont : 256 * 256 = 65 536 possibilités.

Au final comment décoder facilement un octet ? Dans le cas où vous recevez le nombre « 00110011 » que représente-t-il pour nous ?

Afin de simplifier les choses, commençons par partir de notre système de nombres. Notre système est décimal car il contient 10 chiffres différents qui nous servent de base. Si vous tombez sur le nombre « 62 », par habitude, vous ne trouverez aucune difficulté à l'appréhender, mais quelles étapes accomplissez-vous ? En découpant ce nombre on a le nombre des unités qui vaut « 2 » et le nombre des dizaines qui vaut « 6 » et c'est ce dernier qui nous intéresse. Il vaut *6 groupes de 10 unités* (car système décimal). Même principe avec « 2532 », on a 2 unités, *3 groupes de 10 unités*, *5 groupes de 100 unités* (10 * 10) et *2 groupes de 1000 unités* (10 * 10 * 10).

Chaque déplacement d'un chiffre vers la gauche le multiplierait par 10, ou en d'autres mots ce sont des puissances de 10. En version imagée :

| | | | |
|---|---|---|---|
| **2** | **5** | **3** | **2** |
| **2*(10*10*10)** | **5*(10*10)** | **3*(10)** | **2*(1)** |
| **$2*(10^3)$** | **$5*(10^2)$** | **$3*(10^1)$** | **$2*(10^0)$** |

Appliquons le même principe à notre nombre binaire « 10110011 ». La différence ici est que ce n'est plus du décimal mais du binaire avec des puissances de 2. Vu qu'on a ici 8 chiffres passons direct à la représentation visuelle :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **$1*(2^7)$** | **$0*(2^6)$** | **$1*(2^5)$** | **$1*(2^4)$** | **$0*(2^3)$** | **$0*(2^2)$** | **$1*(2^1)$** | **$1*(2^0)$** |
| **128** | **0** | **32** | **16** | **0** | **0** | **2** | **1** |

Ce qui nous donne au final = 128 + 32 + 16 + 2 + 1 = 179

« 10110011 » en binaire vaut donc 179 en décimal.

<u>Quizz</u>

1) Que vaut « 0101 » en décimal ?

2) Que vaut « 11111100 » en décimal ?

3) Quel autre système de numérotation autre que le décimal utilisez-vous tous les jours ?

4) Que vaut « 7 » en binaire ?

5) Que vaut « 237 » en binaire ?

Vous aurez noté qu'on a vu comment passer du binaire au décimal mais pas le contraire. Pour passer du binaire au décimal on a recherché un chiffre multiplié par sa base qui était exposant sa position en partant de zéro et de la droite. (Base 2 pour le binaire dans notre cas, base 10 pour le décimal)

Pour aller dans le sens inverse on peut trouver quel est la plus haute valeur d'un exposant deux qui rentre dans le chiffre qu'on veut convertir, une fois trouvé on peut placer « 1 » à la place correspondante à l'exposant. On répétera cette opération avec le reste de l'opération précédente jusqu'au point où il ne nous reste rien et on comble ensuite les « trous » avec des 0. Un exemple visuel nous permettra d'y voir plus clair.

Avec pour exemple le nombre « 67 », on peut faire rentrer au maximum 64 ($2^6$) et il nous reste 3. Le chiffre suivant qui peut rentrer est 2 ($2^1$) avec un reste de 1 et finalement c'est le chiffre 1 ($2^0$) qui rentre. On comble les positions pour les exposants 5, 4, 3, et 2 avec des zéros.

Résultat = 1 ($2^6$ soit 64) + 0 ($2^5$) + 0 ($2^4$) + 0 ($2^3$) + 0 ($2^2$) + 1 ($2^1$ soit 2) + 1 ($2^0$ soit 0) = 1000011

Si vous avez eu des difficultés avec les deux dernières questions du quizz, retentez maintenant que vous avez la méthode.

Un dernier point avant de terminer ce chapitre. Vous pouvez remarquer que même en sachant convertir du binaire en décimal il reste toujours dur à lire et prend beaucoup de place. (Par exemple : « 10010011  11111100 ») C'est pourquoi on utilise aussi un autre système de notation pour simplifier la vie des pauvres humains qui devront lire ces nombres. Cette notation s'appelle l'*hexadécimal* et possède une base de 16. Comme il faut à cette

notation 16 symboles pour représenter chaque valeur, on a pris les 10 nombres de notre système décimal et on a comblé en ajoutant les premières lettres de l'alphabet ce qui nous donne :

**0 1 2 3 4 5 6 7 8 9 A B C D E F**

Ce système n'a pas été choisi par hasard, le fait qu'il comporte 16 chiffres lui permet de stocker jusqu'à $2^4$ (16) possibilités dans seulement *un seul caractère*. On peut donc représenter 4 bits (« 1 » ou « 0 ») dans un seul chiffre hexadécimal.

Si on reprend l'exemple binaire plus haut « 10010011  11111100 » on peut le diviser en 4 groupes de 4 bits ce qui nous donne « 1001 », « 0011 », « 1111 », « 1100 ». Ensuite on prend la valeur décimale de chaque groupe et on peut la convertir dans le chiffre hexadécimal qui convient.

**« 1001 » => $2^3 + 2^0 = 9$ => 9**

**« 0011 » => $2^1 + 2^0 = 3$ => 3**

**« 1111 » => $2^3 + 2^2 + 2^1 + 2^0 = 15$ => F**

**« 1100 » => $2^3 + 2^2 = 12$ => C**

Notre chiffre binaire donnera simplement « 93 FC » en hexadécimal, beaucoup plus simple à lire non ?

Quizz

1) Que donne « 10000001  01111110 » en hexadécimal ?

2) En décimal que vaut « FF » ?

3) Quel est le nombre de possibilités maximum que nous offrent 4 chiffres hexadécimaux ?

Après ce cours certaines choses vous seront plus claires, par exemple comment l'espace de stockage est calculé mais surtout côté web comment choisir et pré-visualiser vos couleurs avec l'hexadécimal ou des ensembles de trois valeurs entre 0 et 255 quand vous devez rentrer des valeurs en décimal.*

* Chaque image non-transparente est composée de 3 couleurs principales : le rouge, le vert et le bleu (ce qui nous donne l'acronyme « RGB » en anglais pour « Red Green Blue »)  et

chacune possède une valeur composée d'un byte. (Donc entre 0 et 255) On aurait donc 3 bytes par couleur soit 3 paires de nombres hexadécimaux. Pour les images transparentes il y a un byte en plus pour s'occuper de la transparence. (RGBA, le A signifiant Alpha)