Lycée polyvalent DIDEROT **Département IRIS**
61, rue David d'Angers
75019 PARIS
http://www.diderot.org

COURS

Numération

Auteur	Version - Date	Nom du fichier
G.VALET	Version 1.0 - Jan 2010	cours-numeration.docx

Cours sur les principes de base de la numération appliquée au codage des données dans les systèmes informatiques. Définition d'un bit, octet, les bases 2,16,10

Sommaire

A. Du numérique à l'analogique

En informatique, les informations échangées sont numériques. Cela signifie qu'il existe **un nombre fini d'états**. Par exemple, en binaire, il existe 2 états différents : 0 et 1.

La représentation de signaux numériques a toujours un nombre bien déterminé d'états. Par opposition, **les signaux analogiques ont un nombre infini d'états**. Un signal analogique peut-être représenté par une sinusoïde qui peut prendre une infinité de valeurs.

Les informations numériques représentent un monde parfait dans lequel un système peut avoir un nombre bien déterminé d'états. Une porte est fermée ou ouverte, une lampe est allumée ou éteinte.

Les signaux analogiques représentent un monde réel dans lequel un système peut avoir un nombre infini d'état. Dans ce monde, la porte n'est pas seulement fermée ou ouverte mais peut-être partiellement ouverte avec un angle dont les valeurs vont de 0° à 180° (Un nombre infini de valeurs).

Dans l'étude des réseaux, les signaux analogiques nous servent à transporter des données numériques.

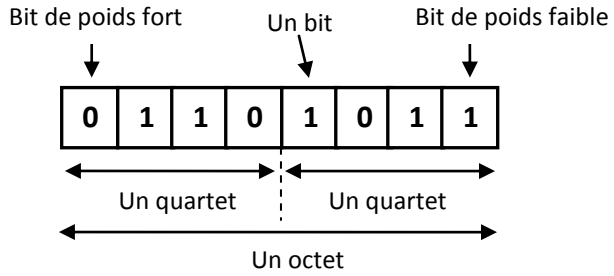
B. Un peu de vocabulaire

B.1. Un bit

C'est le plus petit élément numérique pouvant prendre 2 valeurs : 0 ou 1. Ce terme est repris dans la définition d'un système informatisé pour spécifier le nombre d'éléments numériques (bits) pouvant être traité simultanément (Système 32,64 ou 128 bits)

B.2. Un octet

Souvent rassemblés par paquet de 8 bits, l'octet symbolise un nombre binaire. Un système 32 bits est capable de traiter 4 octets simultanément. **Le bit de poids fort est toujours situé à gauche** et **le bit de poids faible à droite**. Le nombre se lit de gauche à droite :



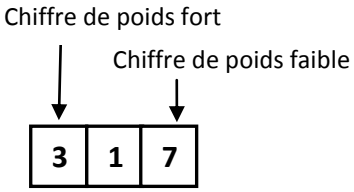
Représentation d'un octet

Les systèmes informatiques **ne sont pas capables de traiter autre chose que des nombres binaires**. Par exemple, une adresse IP (ex : 192.168.0.1) sera stockée dans un ordinateur sous la forme de 4 octets de 8 bits chacun.

Attention : En anglais, un octet se dit « Byte ». Il ne faut pas confondre un « bit » avec un « Byte »

B.3. Le poids fort et faible

La notion de poids est très employée dans le jargon informatique. Il s'agit tout simplement de donner un poids à chaque chiffre d'un nombre. Ce nombre peut être représenté en binaire ou en décimal ou en tout autre base numérique. Si nous prenons l'exemple d'un nombre décimal :



Dans cet exemple, augmenter de 1 le chiffre des centaines à pour conséquence d'incrémenter de 100 le nombre lui-même, alors qu'augmenter de 1 le chiffre des dizaines renvient à incrémenter de 10. On voit bien que chaque chiffre n'a pas le même poids par rapport au nombre représenté.

Le poids fort est donc le chiffre le plus à gauche qui, s'il est changé, provoquera une modification plus importante quant au nombre représenté que tous les autres situés plus à droite. **Il s'agit donc du chiffre le plus significatif.**

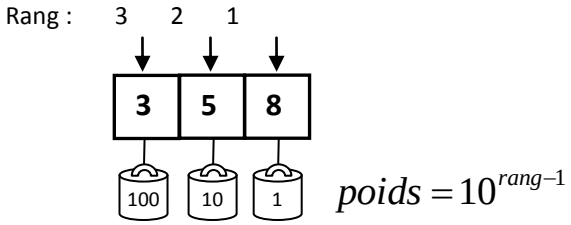
A l'inverse, **le poids faible d'un nombre est celui qui est le moins significatif**.

La valeur du poids dépend de la base numérique dans laquelle est représenté le nombre. Cette valeur suit la règle suivante où x est égal à la base numérique :

$$poids = base^{rang-1}$$

(base = 10 pour décimal, base=2 pour binaire, base=16 pour hexadécimale)

Par exemple, pour le nombre décimal suivant :



Nous pouvons écrire : $3\times10^2 + 5\times10^1 + 8\times10^0 = 300 + 50 + 8 = 358$

C. Les bases numériques

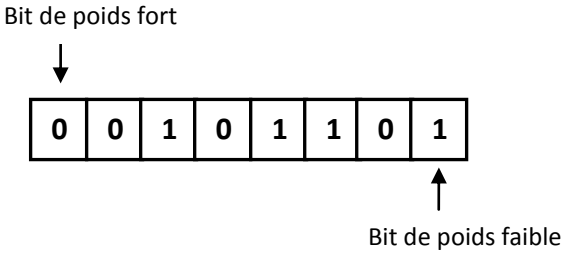
Les bases numériques permettent de représenter des nombres de manière différente. Nous connaissons tous la base décimale qui s'appuie sur 10 éléments pour représenter des nombres (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Il existe d'autres bases utilisées notamment en informatique pour coder des nombres ou des données sous forme de nombres.

En règle générale, **une base *n* utilise *n* éléments différents pour représenter des nombres**

C.1. La base 2 (binaire)

Plus communément appelée base binaire, elle utilise 2 éléments pour coder les nombres : 0 et 1

Voici un exemple d'un nombre binaire codé sur 8 bits :

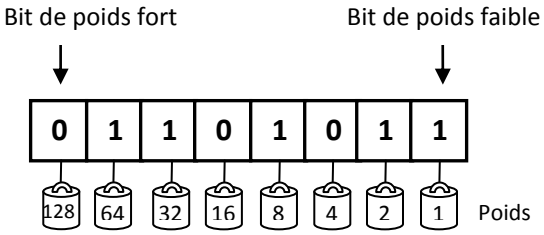


En binaire, à chaque bit, se voit associé un poids. La valeur du poids d'un chiffre dépend de son rang et de la base numérique. Pour calculer la valeur du poids, il suffit d'appliquer la règle suivante pour la base 2 :

$$poids = 2^{rang-1}$$

Pour le rang 1, le poids sera de 0
Pour le rang 2, le poids sera de 2
Pour le rang 3, le poids sera de 4 , etc, …

Le bit de poids faible se verra affecté le poids 0, celui à sa gauche aura le poids 1 et ainsi de suite jusqu'au bit de poids fort :

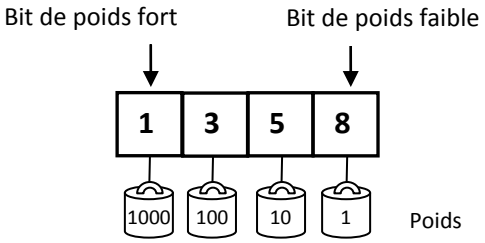


C.2. La base 10 (décimale)

La base décimale, utilisée par tous, se compose de 10 éléments différents servant à représenter des nombres. Le poids donné à chaque chiffre du nombre est égal à :

$$poids = 10^{rang-1}$$

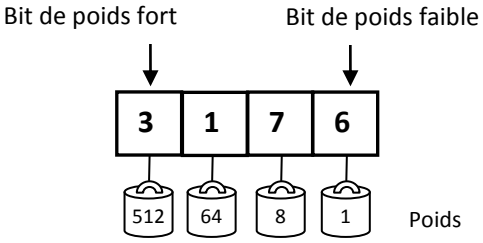
Soit pour le nombre suivant :



C.3. La base 8 (octal)

Le principe est le même que pour les autres bases sauf que la base octale ne dispose que de 8 éléments pour représenter les nombres (0, 1, 2, 3, 4, 5, 6, 7). Le poids de chaque chiffre suit la règle suivante :

$$poids = 8^{rang-1}$$

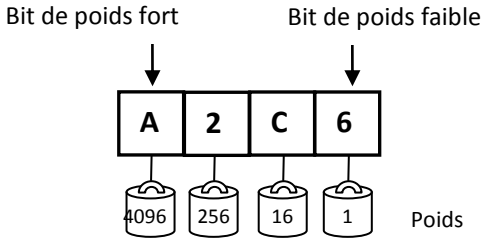


La base octale était souvent utilisée dans les systèmes 8 bits ce qui est rarement le cas aujourd'hui.

C.4. La base 16 (hexadécimale)

Encore souvent utilisé, l'hexadécimal dispose de 16 éléments pour représenter les nombres (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F). Le poids de chaque chiffre suit la règle suivante :

$$poids = 16^{rang-1}$$



C.5. Tableau de correspondance

Binaire	Octal	Décimal	Hexadécimal
0	0	0	0
01	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

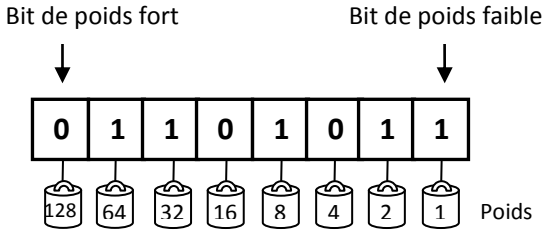
D. Conversions

D.1. Conversion vers la base décimale

C'est sans doute l'opération la plus simple puisqu'il suffit d'appliquer la règle des poids :

$$poids = x^{rang-1}$$

a. Exemple 1 : Convertir un nombre de binaire vers décimal :



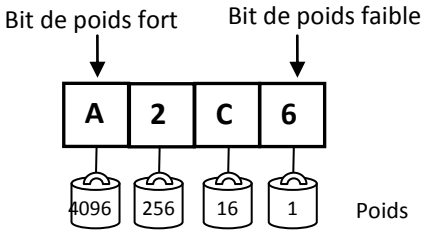
Nous écrirons :

$$0\times128+1\times64+1\times32+0\times16+1\times8+0\times4+1\times2+1\times1=107$$

On notera le résultat comme ceci :

$$(01101011)_2=(107)_{10}$$

b. Exemple 2 : Convertir un nombre hexadécimal en décimal



Nous écrirons :

$$10\times4096+2\times256+12\times16+6\times1=41670$$

On notera le résultat comme ceci :

$$(A2C6)_{16}=(41670)_{10}$$

c. Exemple 3 : Hexadécimal vers binaire

La manipulation de nombres binaires peut vite devenir laborieux étant donné que le nombre de chiffres utilisé peut être très important. Imaginons le nombre binaire suivant :

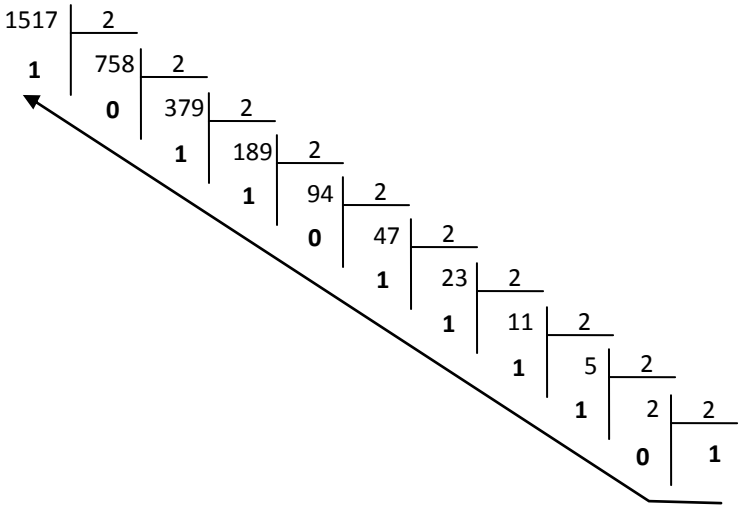
$$(101011101101101010110101)_2=(11459253)_{10}$$

Les bases binaire et hexadécimale (2 et 16) ont une particularite qui permet **regrouper le nombre binaire par paquets de 4 bits** et d'effectuer des correspondances grâce au tableau d'équivalence (page 6) :

$$(1010\ 1110\ 1101\ 1010\ 1011\ 0101)_2$$
$$(A\ \ E\ \ D\ \ A\ \ B\ \ 5)_{16}$$

D.2. Conversion décimal vers binaire

Admettons que nous souhaitions convertir le nombre décimal 1517 en binaire. Le binaire signifie que nous sommes en base 2. Nous allons donc successivement divisé le nombre décimal par 2 jusqu'à ce que le nombre obtenu ne puisse plus être divisé par 2 :

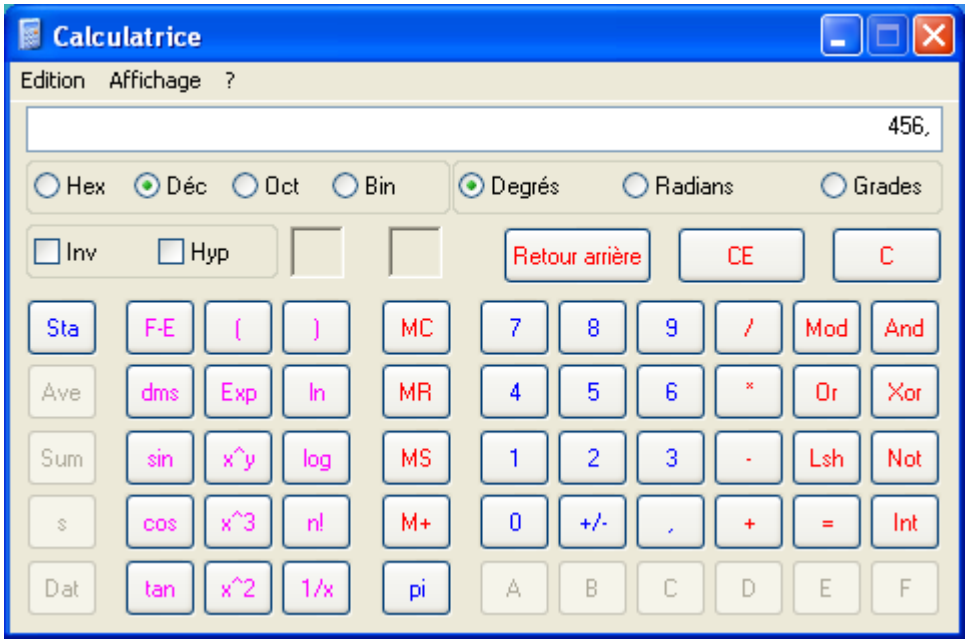


Il suffit alors de lire le nombre binaire **en partant du bas**. Ce qui donne :

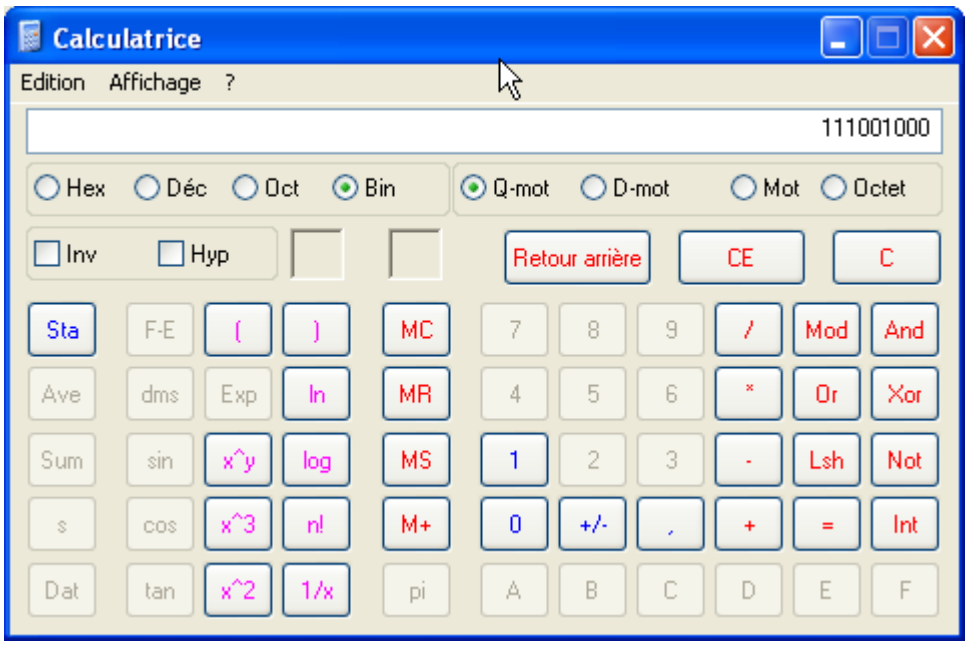
$$(1517)_{10} = (10111101101)_2$$

E. Heureusement, il y a les calculatrices

Pour les réfractaires des bases numériques, il reste la calculatrice intégrée à Windows qui sait convertir depuis et vers les bases 2,8, 10 et 16



Après un clic sur Bin



F. Numération et informatique

De bonnes bases en numération seront nécessaires à la bonne compréhension des concepts suivants :

- Capacité mémoire (ko, Mo, Go et To) des systèmes informatique
- Programmation : Codage des données en machine
- Adressage IP et les masques de sous-réseaux
- Programmation : Opérateurs logiques et arithmétiques

F.1. Octets et associés

Rappelons qu'en ce qui concerne les capacités mémoire (RAM, Disque Dur, …), il est d'usage de les représenter avec les unités suivantes :

- Octets (o ou B) :
  - Exemple : 64 o ou 64 B
- Kilo-octets :
  - Exemple : 512ko ou 512kB
- Méga-octets :
  - Exemple : 512 Mo ou 512 MB
- Giga-octets :
  - Exemple : 4 Go ou 4 GB
- Téra-octets :
  - Exemple : 1To ou 1TB

F.2. Combien vaut un 1ko ?

a. C'est un vieux débat !!!

C'est le débat classique qui agite les informaticiens et les mathématiciens. La question est de savoir si 1ko = 1024 octets ou 1ko = 1000 octets ?

Il est d'usage en informatique de considérer que 1ko = 1024 octets. Malheureusement pour les informaticiens, depuis l'école primaire, nous avons appris que :

- 1 déca = 10
- 1 hecto = 10^2
- 1 kilo = 10^3 = 1000
- 1 Méga = 10^6 = 1 000 000
- 1 Giga = 10^9 = 1 000 000 000
- …

Or, les informaticiens prétendent que :

- 1 kilo octets = 1024 octets
- 1 Méga octets = 1024 x 1024 = 1 048 576 octets

- 1 Giga octets = 1 Mo * 1024 = 1 073 741 824 octets
- …

b. Alors, qui a raison et qui a tort ?

En fait, la plupart du temps, les unités de capacité mémoire dans les systèmes informatiques sont représenté en prenant en compte le fait qu'1 ko = 1024 octets.

Mais d'où vient ce nombre ?

En fait, il faut se rappeler que tout système informatique stocke de l'information sous forme de bit (0 ou 1). La base numérique utilisée est donc la base 2 (base binaire).

Le problème est que le résultat est différent suivant que l'on utilise la base 2 ou la base 10. Par exemple, en base 2 :

- 1 koctet = 2^{10} = 1024 octets

Alors qu'en base 10 :

- 1 koctets = 10^3 = 1000 octets

Afin de résoudre le conflit entre mathématiciens et informaticiens, il a donc été défini une nouvelle norme des unités spécifiques à l'informatique. Cette norme, proposée par le CEI (IEC : International Electrotechnical Commission) en 1998, défini les préfixes binaires suivants :

Concernant les multiples de l'octet, cela donne :

- 1 **kibi**octet (Kio) = 2^{10} octets = 1 024 octets
- 1 **mébi**octet (Mio) = 2^{20} octets = 1 024 Kio = 1 048 576 octets
- 1 **gibi**octet (Gio) = 2^{30} octets = 1 024 Mio = 1 073 741 824 octets
- 1 **tébi**octet (Tio) = 2^{40} octets = 1 024 Gio = 1 099 511 627 776 octets
- 1 **pébi**octet (Pio) = 2^{50} octets = 1 024 Tio = 1 125 899 906 842 624 octets
- 1 **exbi**octet (Eio) = 2^{60} octets = 1 024 Pio = 1 152 921 504 606 846 976 octets
- 1 **zébi**octet (Zio) = 2^{70} octets = 1 024 Eio = 1 180 591 620 717 411 303 424 octets
- 1 **yobi**octet (Yio) = 2^{80} octets = 1 024 Zio = 1 208 925 819 614 629 174 706 176 octets

Les préfixes kilo, méga, giga, téra, etc., correspondent aux mêmes multiplicateurs que dans tous les autres domaines : des puissances de 10. Appliqué à l'informatique, cela donne :

- 1 **kilo**octet (ko) = 10^3 octets = 1 000 octets
- 1 **méga**octet (Mo) = 10^6 octets = 1 000 ko = 1 000 000 octets
- 1 **giga**octet (Go) = 10^9 octets = 1 000 Mo = 1 000 000 000 octets
- 1 **téra**octet (To) = 10^{12} octets = 1 000 Go = 1 000 000 000 000 octets
- 1 **péta**octet (Po) = 10^{15} octets = 1 000 To = 1 000 000 000 000 000 octets

Malgré cette norme de 1998, les fabricants de matériel informatiqu, encouragé par les habitudes prisent par les informaticiens et le grand public, a continué à utiliser les anciens préfixes en considérant à tort que 1ko = 1024 octets.