Chapitre 2

Codage numérique de l'information

Sommaire

L'intégralité de ce chapitre, ou presque, est extrait du document Introduction à l'algorithmique et à la programmation avec Python *de Laurent Signac disponible ici : https://deptinfo-ensip.univ-poitiers.fr*

Les machines électroniques nous entourent désormais : de la simple calculatrice aux super-ordinateurs, en passant par les smartphones, les tablettes et les micro-ordinateurs. À partir de quel degré de complexité avons-nous affaire à un ordinateur ?
Comme nous allons le voir, les ordinateurs actuels sont à peu près calqués sur un modèle datant des années 50, le modèle de VON NEUMANN. Si nous devions retenir deux caractéristiques des ordinateurs, ce serait que :

- un ordinateur doit être programmable ;
- son programme doit être enregistré dans sa mémoire.

Par voie de conséquence, une simple calculatrice de poche n'est pas vraiment un ordinateur, alors que les modèles programmables plus perfectionnés en sont. De même que les tablettes et les smartphones...

Dans nos micro-ordinateurs domestiques, les différents composants sont : la carte mère, le processeur, la mémoire, la carte graphique, les disques...
Ces composants électroniques communiquent entre eux par l'intermédiaire de bus et les données qui transitent sont représentées sous forme binaire (0 ou 1).

Dans un ordinateur, absolument tout (données et programmes, en mémoire, sur les disques durs ou sur cd...) est stocké sous forme binaire.
Dans la suite, nous verrons comment utiliser le système binaire, et comment des objets complexes, comme des images ou de la musique, sont codées en binaire.

2.1 Codage numérique

L'information est la matière première de l'informatique [1]. Les algorithmes, qui sont constitués d'informations, stockent, manipulent et transforment d'autres informations, à l'aide de machines.
Tout, en informatique, est représenté comme une séquence de 0 et de 1 : les algorithmes, ou plutôt les programmes, le contenu d'un livre, une photo, une vidéo...

2.1.1 Pourquoi le binaire ?

Il y a une raison théorique et une raison technique à l'utilisation du binaire :

- on ne peut pas faire plus simple que 2 symboles (avec un seul symbole, c'est-à-dire en « unaire » plus rien ne fonctionne) ;
- les deux symboles 0 et 1 sont transposables électroniquement par : le courant passe ou ne passe pas (système assez robuste).

On représente donc toutes les informations sous forme de bits (*binary digit* = chiffre binaire). La quantité d'information est justement le nombre de bits nécessaires pour représenter cette information.

2.1.2 Unités

- le *bit* (*binary digit*) est l'unité d'information : 0 ou 1
- l'*octet* est un groupe de 8 *bits* (attention, *octet* se dit *byte* en anglais)

Tout le reste.... dépend du contexte. En particulier, le préfixe « kilo » correspond dans le système international au multiplicateur 10^3, mais conventionnellement, ilétait utilisé en informatique pour le multiplicateur 2^{10}.
Nous devrions utiliser les préfixes du SI (système international) pour le multiplicateur 10^3 (symboles k,M,G...) et d'autres préfixes (kibi, mébi, gibi,... symboles ki,Mi,Gi...) pour les multiplicateurs 2^{10}, 2^{20}, 2^{30}... Il en est de même pour les débits (bits/seconde, puis kbits par seconde etc...). Dans les symboles, le « o » de octet est parfois remplacé par le « B » de byte, à ne pas confondre avec le « b » de bit...
Enfin, l'explorateur Windows utilise le symbole Ko pour 2^{10} octets (alors que ce devrait être 10^3 octets).
Dans le doute, partez du principe que si on vous vend de la mémoire (disque dur, mémoire flash...) sa taille est indiquée en utilisant le SI...

1. Le mot vient d'ailleurs de là : **infor**mation auto**matique**, et a été adopté en 1967.

2.1.3 Codage des nombres entiers

Codage en base b

L'écriture d'un nombre en base b est formée à partir de b symboles, appelés chiffres.
Ainsi, par exemple et en général :

- les 10 chiffres de la base 10 sont : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9 ;
- les 2 chiffres de la base 2 sont : 0 et 1 ;
- les 16 chiffres de la base 16 sont : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F

L'écriture en base b fonctionne de la manière indiquée par les exemples suivants :

- Le nombre en base 10 écrit 232567 vaut $2\times10^5+3\times10^4+2\times10^3+5\times10^2+6\times10+7$;
- Le nombre en base 2 écrit 10011 vaut $1\times2^4+0\times2^3+0\times2^2+1\times2+1=16+2+1=19$ en base 10 ;
- Le nombre en base 16 écrit 2A1 vaut $2\times16^2+10\times16+1=512+160+1=673$ en base 10.

La base est en général indiquée en indice à la fin du nombre. En l'absence de cette indication, c'est la base 10 qui est utilisée. Ainsi, au lieu d'écrire « le nombre en base 2 écrit 10011 » on notera $10011|_2$.
Pour passer de la base 10 à la base 2, on peut procéder par divisions successives (par 2) jusqu'à ce que le dernier quotient obtenu soit 0.
Voici comment trouver l'écriture binaire de 90 :

$$\begin{array}{rcrcl} 90 & = & 2\times45 & + & 0 \\ 45 & = & 2\times22 & + & 1 \\ 22 & = & 2\times11 & + & 0 \\ 11 & = & 2\times5 & + & 1 \\ 5 & = & 2\times2 & + & 1 \\ 2 & = & 2\times1 & + & 0 \\ 1 & = & 2\times0 & + & 1 : \text{le quotient est 0; on arrête} \end{array}$$

La séquence des restes successifs, lue du dernier au premier donne : 1011010 qui est l'écriture en base 2 de 90.
En effet :

$$\begin{aligned} 90 &= 2\times45+0 \\ &= 2\times(2\times22+1)+0=2^2\times22+2\times1+0 \\ &= 2^2\times(2\times11+0)+2+0=2^3\times11+2^2\times0+2\times1+0 \\ &= 2^3\times(2\times5+1)+2^2\times0+2\times1+0=2^4\times5+2^3\times1+2^2\times0+2\times1+0 \\ &= 2^4\times(2\times2+1)+2^3\times1+2^2\times0+2\times1+0=2^5\times2+2^4\times1+2^3\times1+2^2\times0+2\times1+0 \\ &= 2^5\times(2\times1+0)+2^4\times1+2^3\times1+2^2\times0+2\times1+0=2^6\times1+2^5\times0+2^4\times1+2^3\times1+2^2\times0+2\times1+0 \\ &= 2^6\times(2\times0+1)+2^5\times0+2^4\times1+2^3\times1+2^2\times0+2\times1+0 \\ &= 2^7\times0+2^6\times1+2^5\times0+2^4\times1+2^3\times1+2^2\times0+2\times1+0 \\ &= 01011010|_2=1011010|_2 \end{aligned}$$

L'écriture de 90 est donc bien 1011010 en binaire qui est bien la séquence des restes successifs, lue du dernier au premier.
On peut procéder de même en base 16 (en faisant des divisions par 16). Les restes sont alors entre 0 et 15, ce qui correspond bien aux valeurs des chiffres de la base 16.

Quelques opérations effectuées en binaire

L'addition : Quand on *pose* une addition en décimal, on part de la position des unités et on poursuit vers la position des dizaines puis des centaines, etc. donc de la droite vers la gauche.

À chaque position, vous faites la somme des chiffres se trouvant à la même position (colonne).

Si cette somme est inférieure à 10, alors vous écrivez le chiffre obtenu à cette position et passez à la position suivante.

Si la somme est supérieure à 10, alors vous écrivez le chiffre des unités obtenu et *retenez* le chiffre des dizaines de cette somme pour l'additionner à la position suivante (si vous faites la somme de 2 chiffres, étant chacun compris entre 0 et 9, alors cette somme sera comprise entre 0 et 18 et la *retenue* ne pourra être supérieure à 1).

Le principe est exactement le même en binaire. Vous faites la somme, écrivez le chiffre des unités et retenez le chiffre de la dizaine de cette somme (s'il y en a) qu'il faudra ajouter à la position suivante.

Si vous faites la somme de 2 chiffres en binaire, alors il n'y a que 4 cas possibles :

Chiffres en binaire			Chiffres équivalents en décimal		
0	0	$0+0=0$	0	0	$0+0=0$
0	1	$0+1=1$	0	1	$0+1=1$
1	0	$1+0=1$	1	0	$1+0=1$
1	1	$1+1=10$	1	1	$1+1=2$

Dans le dernier cas, le chiffre à écrire dans la position est 0 et on ajoute 1 (la retenue) à la position suivante.

Exemple. On cherche à additionner les deux nombres suivants en décimal puis en binaire et à vérifier que le résultat obtenu est le même.

1. Calculer $71 + 19$.
2. On admet que $71 = 1000111|_2$ et que $19 = 10011|_2$.
   (a) Calculer la somme en binaire.
   (b) Transformer le résultat obtenu en écriture décimale et comparez.

La multiplication : Lorsqu'on pose une multiplication en décimale, on exécute automatiquement tout un tas de calculs complexent qui peuvent s'expliquer par la distributivité de la multiplication sur l'addition. Prenons par exemple 46×23. L'opération, traditionnellement, se pose ainsi :

$$\begin{array}{cccc} & & 4 & 6 \\ & \times & 2 & 3 \\ \hline & 1 & 3 & 8 \\ + & 9 & 2 & 0 \\ \hline 1 & 0 & 5 & 8 \end{array}$$

Décomposons les calculs :

				$(4\times 10$	$+$	$6\times 1)$	
			$\times$	$(2\times 10$	$+$	$3\times 1)$	
		$(4\times 10$	$+$	$6\times 1)$	$\times$	3×1	
	$+$	$(4\times 10$	$+$	$6\times 1)$	$\times$	2×10	(distributivité)
				12×10	$+$	18×1	(distributivité)
	$+$	8×10^2	$+$	12×10	$+$	0×1	(distributivité)
				13×10	$+$	8×1	(*retenue*)
	$+$	9×10^2	$+$	2×10	$+$	0×1	(*retenue*)
		9×10^2	$+$	15×10	$+$	0×1	
		10×10^2	$+$	5×10	$+$	0×1	(*retenue*)
1×10^3	$+$	0×10^2	$+$	5×10	$+$	0×1	(*retenue*)

Voilà tout ce qui est mise en œuvre quand on « pose » une multiplication !

Si on remplace les puissances de 10 par des puissances de 2 on obtient exactement les mêmes mécanismes : la multiplication telle qu'on la *pose* en décimal peut se poser à l'identique en binaire.

Examinons les produits possibles :

Chiffres en binaire			Chiffres équivalents en décimal		
0	0	$0\times 0=0$	0	0	$0\times 0=0$
0	1	$0\times 1=0$	0	1	$0\times 1=1$
1	0	$1\times 0=0$	1	0	$1\times 0=1$
1	1	$1\times 1=1$	1	1	$1\times 1=1$

Ce sont les mêmes produits en binaire qu'en décimal (pratique).

On admettra que $46 = 101110|_2$ et que $23 = 10111|_2$.

Posons la multiplication, en utilisant le *point* à la place du 0 pour ne pas se perdre dans les décallages successifs :

					1	0	1	1	1	0
					$\times$	1	0	1	1	1
					1	0	1	1	1	0
				$+$	1	0	1	1	1	0	.
			$+$	1	0	1	1	1	0	.	.
			$+$	0	0	0	0	0	.	.	.
		$+$	1	0	1	1	1	0	.	.	.
1	0	0	0	0	1	0	0	0	1	0

Vérifiez que $1058 = 10000100010|_2$.

2.1.4 Codage des nombres non entiers et stockage des nombres

Codage

Les nombres non-entiers sont codés de la même manière. Chaque chiffre placé après la virgule est associé à une puissance négative de la base :
$101,011|_2 = 1\times 2^2 + 0\times 2 + 1 + 0\times 2^{-1} + 1\times 2^{-2} + 1\times 2^{-3} = 5 + 0,25 + 0,125 = 5,375$

Le passage de la base 10 à la base 2, pour la partie fractionnaire, est réalisé par multiplications successives (on reporte la partie fractionnaire de chaque résultat sur la ligne suivante, et les parties entières des résultats obtenus forment l'écriture en base 2).
Voici comment écrire 0,6875 en base 2 :

$$\begin{array}{rcrcl} 0,6875 & \times & 2 & = & 1,375 \\ 0,375 & \times & 2 & = & 0,75 \\ 0,75 & \times & 2 & = & 1,5 \\ 0,5 & \times & 2 & = & 1 \end{array}$$

Nous prenons les parties entières des résultats dans l'ordre où elles sont obtenues : 1011. Donc $0,6875 = 0,1011|_2$.

Remarque. Notons que si l'écriture d'un nombre non entier en base 2 est finie, ce sera aussi le cas en base 10 (car les puissances négatives de 2 ont toutes une écriture décimale finie). En revanche, si l'écriture décimale est finie, l'écriture en base 2 ne le sera pas forcément (elle sera alors périodique)

Exemple. Écrire 0,2 en base 2 en utilisant la méthode indiquée ci-dessus. Que remarque-t-on ?

Stockage

Nous avons déjà vu dans le paraphe précédent quelques types de codage ; nous avons aussi vu que beaucoup de nombres non entiers avaient une écriture infinie en binaire, qui est le codage le plus répandu. Mais les capacités de stockage des machines ne le sont pas, elles, infinies, aussi convient-on de ne conserver qu'une partie du codage du nombre, le plus proche possible de la valeur exacte de ce nombre.
Deux types de méthodes sont possibles :

À virgule fixe : Cela consiste à réserver un certain nombre de chiffres pour la partie entière et un certain nombre de chiffres pour la partie décimale. La précision des calculs est un défaut de ce type de codage.

À virgule flottante : Nous ne détaillerons pas ici ce codage, assez technique, mais il est important de retenir :

- que c'est le codage le plus utilisé pour les nombres non entiers ;
- qu'il représente les nombres de manière approchée, et qu'en virgule flottante, les calculs ne sont (pratiquement) jamais exacts ; cependant ils sont beaucoup plus précis, en général, qu'avec un stockage à virgule fixe.

L'utilisation très fréquente des calculs à l'aide de nombres non entiers et la nécessité d'avoir une erreur « relative » peu importante ont fait que seul le codage en virgule flottante a été conservé.

2.2 Numérique vs analogique

Beaucoup d'objets quotidiens existent à la fois en version numérique et analogique, même si la première tend à remplacer petit à petit la seconde.
Ainsi, nous pouvons opposer les disques compacts audio et les disques vinyles, la photo numérique et la photo argen- tique, les e-book et les livres traditionnels. Dans tous ces cas, l'objet numérique n'est qu'une succession de chiffres 0 et 1...

2.2.1 Codage des couleurs

- Les couleurs, sur un écran, sont formées par synthèse additive (addition de la lumiére). L'impression, au contraire, est réalisée par synthèse soustractive.
- Les couleurs primaires en synthèse additive sont : le rouge, le vert et le bleu (en synthèse soustractive, il s'agit du cyan, du magenta, et du jaune).
- Supposons qu'on dispose de 3 leds, une rouge, une verte et une bleue, et que chacune puisse être allumée ou éteinte (on ne peut pas varier leur intensité).

À partir de ces 3 leds, nous pouvons obtenir $2^3 = 8$ couleurs puisque chaque led peut être soit allumée, soit éteinte.
Nous obtenons ainsi du noir si toutes les leds sont éteintes, du blanc si elles sont toutes allumées ou du cyan si seules les leds bleues et vertes sont allumées.
Puisqu'à chacune des 8 couleurs obtenues, correspond un état de chacune des trois leds, nous pouvons convenir de coder l'état des leds dans l'ordre rouge, vert, bleu par 0 (led éteinte) ou 1 (led allumée). Ainsi, une série de 3 chiffres binaires, (autrement dit un entier entre 0 et 7) est associé à une couleur :

0	000	noir	4	100	rouge
1	001	bleu	5	101	magenta
2	010	vert	6	110	jaune
3	011	cyan	7	111	blanc

Les différents moyens de codage reposent sur un certain nombre de conventions. Par exemple, modifier l'ordre des couleurs primaires dans le codage des couleurs en vert, bleu, rouge à la place de rouge, vert, bleu modifie bien entendu les couleurs associées aux nombres. Respecter ces conventions, c'est adopter un standard, et cette adoption facilitera grandement les échanges de données entre les personnes ou les programmes. En particulier, si le standard est publié, on dit qu'il est ouvert [2], par opposition à un standard fermé, qui restreint, de manière légale ou en ne publiant pas les spécifications du standard, l'écriture d'applications compatibles qui pourraient utiliser les mêmes fichiers de données, par exemple.
Les standards de représentation et de codage des informations sont extrêmement nombreux. Pour chaque type d'objet numérique, il existe de nombreux standards : pour le texte (latin, utf, *ascii*...), pour les documents mis en page (Open Document Format, Office Open xml...), les images (jpeg, png...), l'audio (mp3, ogg, flac...), la vidéo (mpegv2, Vorbis, h-264...).

2.2.2 Codage du texte

Les caractères sont généralement codés par des nombres, par une simple table de correspondance. Le standard le plus ancien est l'*ascii* (American Standard Code for Information Interchange), qui contient les caractères latins non accentués, les chiffres, des symboles de ponctuation, codés sur 7 bits (au total, la figure 2.1 page suivante contient donc 128 symboles, caractères non imprimables compris).
Le code *ascii* a rapidement été insuffisant, faute de caractères accentués, grecs, cyrilliques, hébreux, chinois...
Deux autres types de codage ont donc fait leur apparition :

2. Précisément, un standard ouvert est publié mais ne doit pas imposer non plus de restrictions quant à son utilisation.

FIGURE 2.1: Table ascii (numérotée de 0 à 7F, en héxa) tirée de Wikipédia

PDF : en [archive] v · d · m	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
U+0010	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
U+0020	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
U+0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
U+0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
U+0050	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
U+0060	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
U+0070	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

- Les extensions rajoutent des caractères à la suite de la table (chaque alphabet possède alors son extension) : c'est le cas du latin-1 (iso-8859-1). Il y a des extensions pour le chinois, l'hébreu... Il faut donc indiquer quelle est l'extension utilisée pour pouvoir lire correctement le fichier.

- *Unicode* est une norme qui recense tous les caractères existant dans une table (il y en a plus de 100 000 actuellement). Les codes sont ensuite représentés dans un codage particulier comme utf-8, ou utf-16. C'est ce type de codage qu'il convient d'utiliser maintenant.

Le texte peut aussi être enrichi : changement de la taille des caractères, de la graisse, utilisation de l'italique... Là aussi, il existe une multitude de manières de représenter du texte enrichi. Nous pouvons citer le format Rtf (Rich Text Format), le Html (HyperText Markup Language), et dans une certaine mesure les formats issus de logiciels de bureautique.
Nous y reviendrons plus tard dans l'année.

2.2.3 Codage du son

Le codage d'un son, ou d'une courbe en général, est réalisé en échantillonnant dans le temps le signal continu. Chaque échantillon est représenté par un nombre, avec une certaine précision (souvent exprimée en *bits*). Ce codage se nomme *pcm* (Pulse Code Modulation).
Selon la nature du signal ainsi numérisé, il est possible d'utiliser des codages plus économes en mémoires (compression), comme Flac (compression sans perte) ou mp3 (compression avec pertes).

2.2.4 Codage d'une image

Nous reviendrons plus tard dans l'année sur le codage d'une image.

2.3 Exercices

EXERCICE 2.1.
Les questions sont indépendantes.

1. Écrire 144 en binaire.
2. Écrire 263 en base trois.
3. Écrire 3455 en base huit.
4. Écrire 12254 en duodécimal (base douze utilisant les chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A et B).
5. Écrire 28945 en hexadécimal.
6. Écrire $1001110|_2$ en base décimale.
7. Écrire $10200|_3$ en base décimale.
8. Écrire $2024|_8$ en base décimale.
9. Écrire $2A880|_{12}$ en base décimale.
10. Écrire $30D40|_{16}$ en base décimale.

EXERCICE 2.2.
Dans chacun des calculs suivants :

(a) Effectuer le calcul en décimal ;

(b) Coder les opérateurs en binaire ;

(c) Effectuer le calcul en binaire ;

(d) Transformer le résultat binaire en décimal et comparer.

1. $34 + 6$
2. $100 + 28$
3. 21×5
4. 13×11

EXERCICE 2.3.
Les questions sont indépendantes.

1. Écrire 0,5 en binaire.
2. Écrire 0,675 en binaire.