Bases du codage en informatique

Balliot - décembre 2023

Table des matières

I – Le binaire s’impose

Pour compter nous utilisons un système décimal, parce que l’on a commencé par compter sur nos doigts et que nous avons dix doigts (sans utiliser les pieds). On a donc inventé dix chiffres et on utilise leur position dans l’écriture des nombres. Ainsi 235 signifie 5 unités plus 3 dizaines plus 2 dizaines de dizaines.
D’autres systèmes ont existé (base douze, base vingt..)
En informatique tout est codé suivant un système binaire. Dans un ordinateur on a un ensemble d’éléments qui ont deux états possibles (tension 0 volt ou tension 5 volts, courant passe ou pas, polarité positive ou négative, etc..).
Le défi : comment représenter une information (nombre, texte, image, etc) en n’utilisant que des signes binaires ?

Élément binaire - Bit

Binaire : objet n’ayant que deux états possibles s’excluant

oui/non, blanc/noir, 0/1, allumé/éteint, absent/présent, ouvert/fermé..

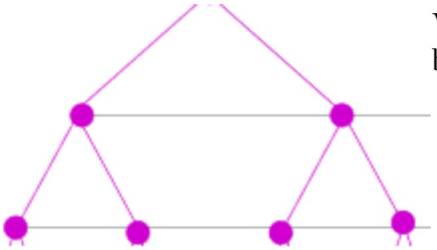
On peut se le représenter sous forme d'une ampoule (allumée ou non), d'un interrupteur. En informatique on a l'habitude, pour simplifier, d'utiliser les deux symboles 0 et 1. Il n'y a pas de 0 ni de 1 dans un ordinateur mais c'est une notation pratique pour se représenter les élément binaires, que l'on appelle des **bits** (bit = abréviation de binary digit).

Avec un bit on ne peut donc représenter deux informations. En en associant plusieurs on augmente le nombre d'informations.

Avec 2 bits : on a les éventualités suivantes :

- le premier est à 0 et le deuxième est à 0
- le premier est à 0 et le deuxième est à 1
- le premier est à 1 et le deuxième est à 0
- le premier est à 1 et le deuxième est à 1

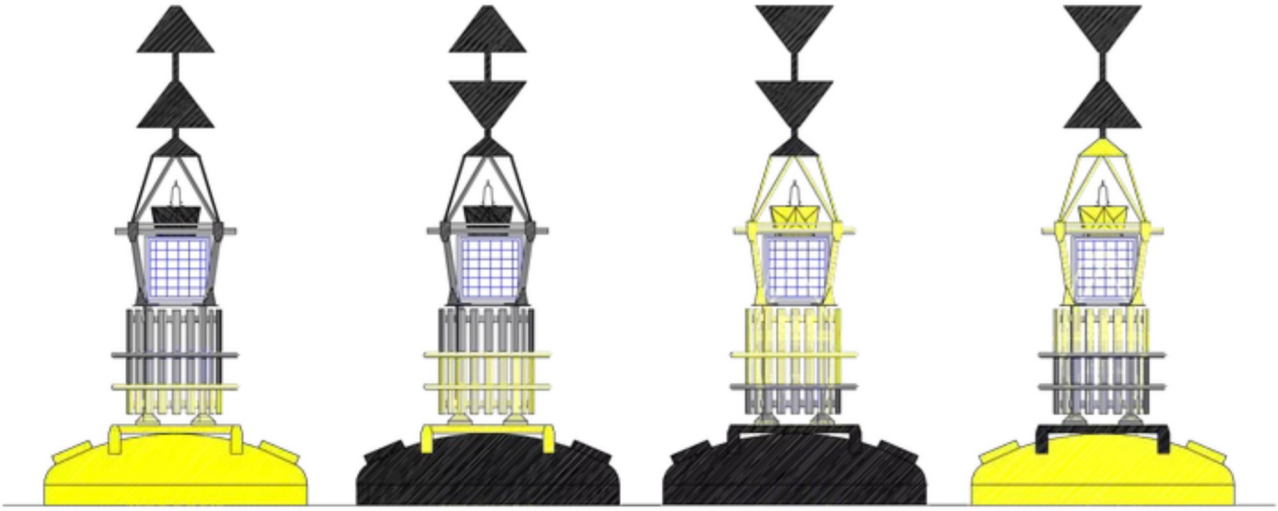
Donc 4 informations différentes 4=2x2



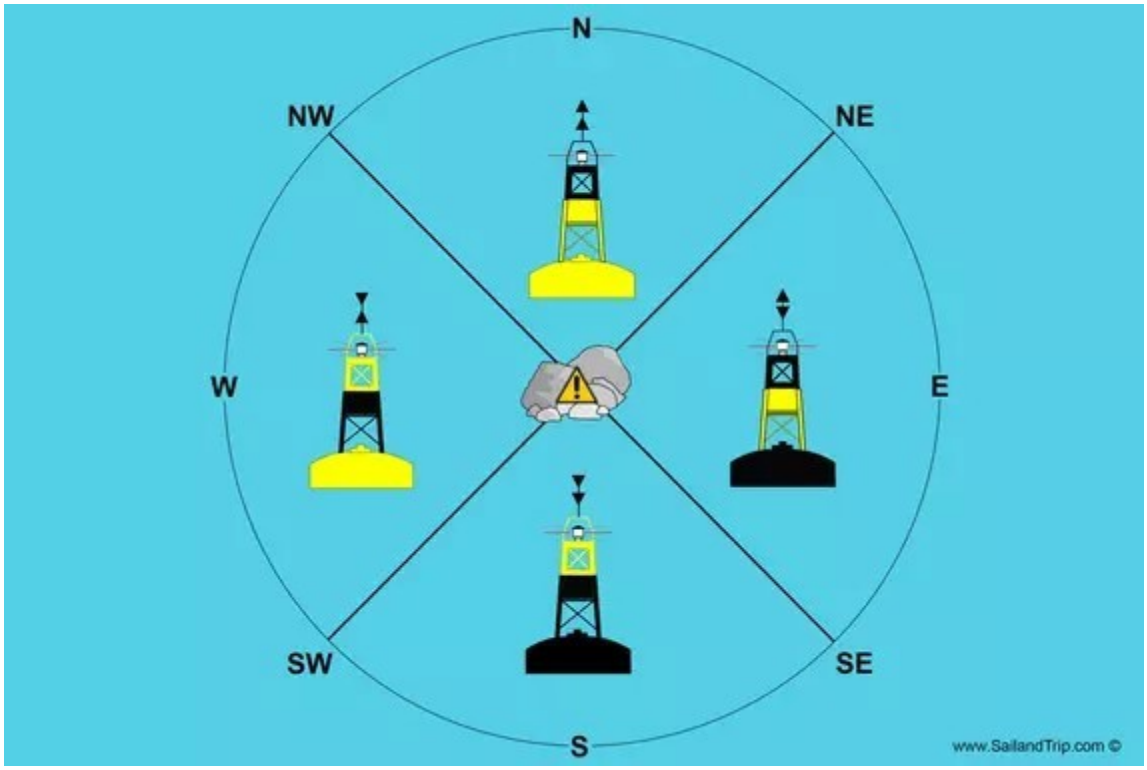
Visualisation avec un arbre binaire

Exemple d'utilisation de deux signaux binaires : le balisage cardinal en mer.

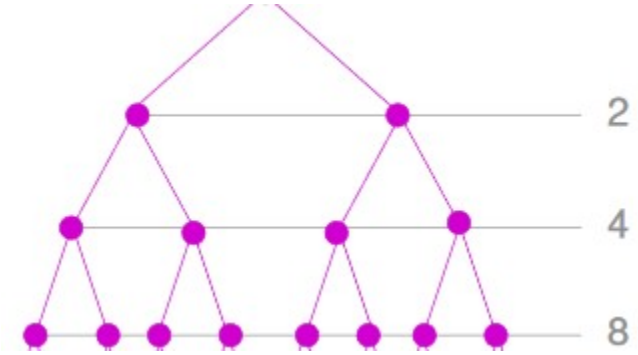
Les deux signes sont un triangle avec pointe vers le haut et un triangle avec pointe vers le bas. En associant deux triangles on désigne les quatre points cardinaux Nord, Est, Sud, Ouest.



Ces balises signalent la présence d'un danger et indiquent de quel côté passer pour l'éviter.



Avec 3 bits : pour chacune des éventualités précédentes on on a deux, donc on obtient la possibilité de coder 8 informations différentes 8=2x2x2



Et avec 4 bits combien d'informations ? Et avec 5, avec 6 ?

II - L’octet

Au minimum on a besoin de savoir coder les chiffres, les lettres minuscules et majuscules, les signes de ponctuation, etc.
On a fait le choix d’utiliser des séquences de 8 bits, appelées octets qui permettent de coder 256 informations différentes. Donc en informatique l’élément de base est **l’octet** (byte en anglais).
$256=2x2x2x2x2x2x2x2= 2^8$

Représentation imagée d’un octet :

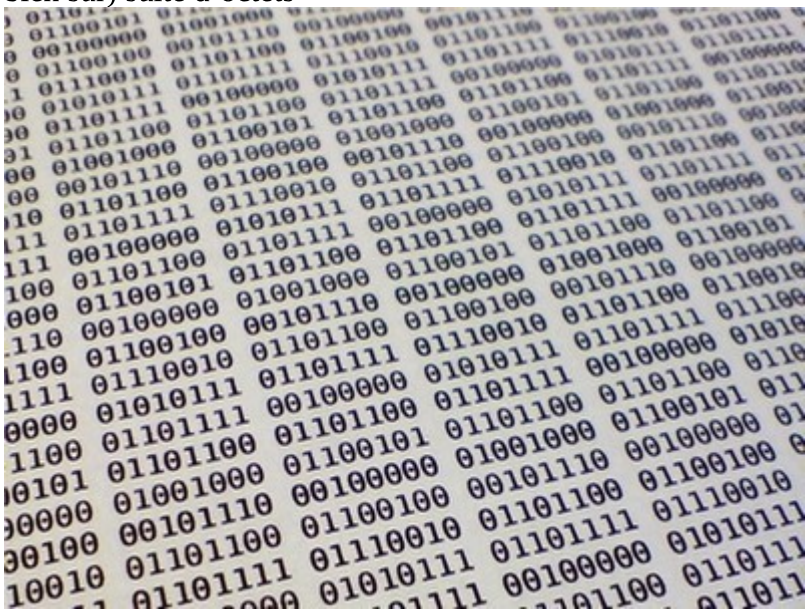
TOUT est codé avec des octets

Par exemple la lettre **A** majuscule est codée 01000001 soit 65 en décimal ou 41 en hexadécimal, tandis que la lettre **a** minuscule est codée 01100001 (code ASCII) , un seul bit les différencie.
Certains caractères spéciaux comme les lettres accentuées nécessitent 2 octets.
Ci-dessous les codes ASCII des caractères de base :

Dec	Hex	char		Dec	Hex	char	Dec	Hex	char	Dec	Hex	char
0	00	NUL	*caract. null*	32	20	espace	64	40	@	96	60	`
1	01	SOH		33	21	!	65	41	A	97	61	a
2	02	STX		34	22	"	66	42	B	98	62	b
3	03	ETX		35	23	#	67	43	C	99	63	c
4	04	EOT		36	24	$	68	44	D	100	64	d
5	05	ENQ		37	25	%	69	45	E	101	65	e
6	06	ACK		38	26	&	70	46	F	102	66	f
7	07	BEL	*bell*	39	27	'	71	47	G	103	67	g
8	08	BS	*backspace*	40	28	(	72	48	H	104	68	h
9	09	TAB	*tabul. horiz*	41	29	)	73	49	I	105	69	i
10	0A	LF	*line feed*	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	*tabul. vertic*	43	2B	+	75	4B	K	107	6B	k
12	0C	FF		44	2C	,	76	4C	L	108	6C	l
13	0D	CR	*retour ligne*	45	2D	-	77	4D	M	109	6D	m
14	0E	SO		46	2E	.	78	4E	N	110	6E	n
15	0F	SI		47	2F	/	79	4F	O	111	6F	o
16	10	DLE		48	30	0	80	50	P	112	70	p
17	11	DC1		49	31	1	81	51	Q	113	71	q
18	12	DC2		50	32	2	82	52	R	114	72	r
19	13	DC3		51	33	3	83	53	S	115	73	s
20	14	DC4		52	34	4	84	54	T	116	74	t
21	15	NAK		53	35	5	85	55	U	117	75	u
22	16	SYN		54	36	6	86	56	V	118	76	v
23	17	ETB		55	37	7	87	57	W	119	77	w
24	18	CAN		56	38	8	88	58	X	120	78	x
25	19	EM		57	39	9	89	59	Y	121	79	y
26	1A	SUB		58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	*escape*	59	3B	;	91	5B	[	123	7B	{
28	1C	FS		60	3C	<	92	5C	\	124	7C	|
29	1D	GS		61	3D	=	93	5D	]	125	7D	}
30	1E	RS		62	3E	>	94	5E	^	126	7E	~
31	1F	US		63	3F	?	95	5F	_	127	7F	DEL

**Voir en annexe 10 la conversion de bases pour l’écriture d’un nombre.*
Ex : 65 en décimal (base dix) s’écrit 01000001 en binaire (base deux) et 41 en hexadécimal (base seize).

Voilà comment on peut s’imaginer la mémoire d’un ordinateur (sans correspondance avec la réalité bien sûr) suite d’octets



Toute l’information est stockée sous forme de suite d’octets que l’on appelle un **fichier**, et c’est ainsi qu’elle circule, jusqu’à un terminal, par exemple un écran, où elle est décodée.