Base64

En informatique, **base64** est un codage de l'information utilisant 64 caractères, choisis pour être disponibles sur la majorité des systèmes. Il est principalement utilisé pour la transmission de messages (courrier électronique et messages de forum Usenet). Il est défini en tant que codage MIME.

Un alphabet de 65 caractères est utilisé pour permettre la représentation de 6 bits par un caractère simple. Le 65^{e} caractère (signe « = ») n'est utilisé qu'en complément final dans le processus de codage d'un message.

Ce processus de codage consiste à coder chaque groupe de 24 bits successifs de données par une chaîne de 4 caractères simples. On procède de gauche à droite, en concaténant 3 octets pour créer un seul groupement de 24 bits (8 bits par octet). Ils sont alors séparés en 4 nombres de seulement 6 bits (qui en binaire ne permettent que 64 combinaisons). Chacune des 4 valeurs est enfin représentée (codée) par un caractère simple et prédéfini de l'alphabet retenu. (Table ci-dessous.)

Ainsi 3 octets quelconques sont remplacés par 4 caractères simples, choisis pour être compatibles avec tous les systèmes existants.

Chaque **valeur** (chaque groupe de 6 bits) est utilisée comme index dans la table ci-dessous.
Le caractère correspondant est indiqué dans la colonne **codage**.

Valeur	Codage	Valeur	Codage	Valeur	Codage	Valeur	Codage
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(complément)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Un traitement spécial est effectué si moins de 24 bits sont disponibles à la fin de la séquence de données à coder (elle n'a pas forcément une taille multiple de 24 bits). Dans un tel cas, des zéros sont ajoutés à la droite des données initiales pour aller vers le multiple de 6 bits le plus proche. Chaque paquet de 6 bits est converti dans l'alphabet. Puis on ajoute des caractères '=' complémentaires pour former quand même 4 caractères.

Puisque les données d'entrée doivent être constituées d'un nombre entier d'octets, seuls trois cas sont possibles en fin de séquence :

- il reste exactement 3 octets à coder (24 bits), alors on obtient directement 4 caractères sans traitement complémentaire ;
- il reste seulement 2 octets (16 bits) à coder, alors on ajoute 2 zéros à droite pour former 3 caractères de l'alphabet (3×6 = 16+2 = 18 bits) suivis d'un quatrième caractère '=' en complément;
- il reste un seul octet (8 bits) à coder, alors on ajoute 4 zéros à droite pour former 2 caractères de l'alphabet (2×6 = 8+4 = 12 bits) suivis de deux caractères '=' en complément.

Reproches

On reproche souvent à ce codage une augmentation de la taille de données. Car les caractères de codage (ceux de l'alphabet) sont généralement stockés ensuite selon la table ASCII habituelle. Ce qui immobilise un octet pour chaque caractère. Bien sûr toutes les possibilités de la table ASCII ne sont pas utilisées. C'était l'objectif de se restreindre à certains caractères !

Il faut néanmoins 32 bits (4×8) pour en stocker 24 (3×8) dans les systèmes habituels. C'est alors une augmentation d'au moins 33% de la taille.

On reproche aussi souvent à ce codage d'être illisible au premier regard humain. Contre ces défauts, si la majorité des caractères d'un texte initial sont déjà "simples" (c'est-à-dire compatibles avec tous les systèmes), on peut envisager de ne coder que les caractères problématiques.

Intérêt

L’intérêt de l'encodage base64 ne se trouve donc pas dans la représentation de données textuelles, mais surtout dans la représentation de données binaires.

Lorsque l’on veut représenter des données binaires (une image, un exécutable) dans un document textuel, tel qu’un courriel, la simple transcription des 0 et des 1 utilise en ASCII un octet pour chaque bit : soit une augmentation de la taille d’un facteur 8. L'encodage en base64 devient donc très compétitif.

Par ailleurs, le reproche fait sur la lisibilité des données tombe de lui-même dans ces conditions : les données binaires n’ont pas vocation à être compréhensibles sans interprétation par un logiciel dédié (cas d’une image, par exemple).

Exemple

Encodage

HEX : 00 1A 2B 00 FF E7
BIN 8 octets : 00000000 00011010 00101011 00000000 11111111 11100111
BIN 24 octets : 000000000001101000101011 000000001111111111100111
BIN 6 octets : 000000 000001 101000 101011 000000 001111 111111 100111

DEC :	00	01	40	43	00	15	63	39
Base64 :	A	B	o	r	A	P	/	n

Decodage

Base64 : A B C D E F G =
DEC : 00 01 02 03 04 05 06 x
BIN 6 : 000000 000001 000010 000011 000100 000101 000110 x
BIN 24 : 000000000001000010000011 000100000101000110000000
BIN 8 : 00000000 00010000 10000011 00010000 01010001 10000000
HEX : 00 10 83 10 51 80