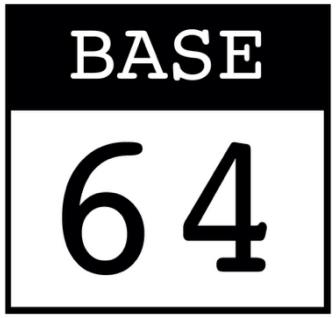

ENCODAGE BASE64

Certains **protocoles applicatifs** (comme **SMTP**, **HTTP** ou **FTP**) fonctionnent uniquement avec des **caractères imprimables** du jeu **ASCII**.

Or, l'encodage ASCII contient également des **caractères spéciaux** ou **de contrôle** (comme le **retour chariot**, la **tabulation** ou la **fin de ligne**) qui peuvent être **interprétés à tort** par ces protocoles comme des instructions ou des séparateurs de message.

Pour éviter toute ambiguïté, l'encodage **Base64** a été conçu.
Il repose sur un **alphabet réduit à 64 symboles sûrs** : **`A–Z, a–z, 0–9, +, /`**

Ces caractères sont tous **imprimables et non ambigus**, et **excluent les caractères de contrôle** du code ASCII.

Grâce à cette sélection, **Base64** permet de **transporter des données binaires sous forme de texte**, sans risque de corruption, d'altération ou de mauvaise interprétation par les protocoles réseau.

PRINCIPE DE L'ENCODAGE BASE64

Règles de conversion Base64 :

1. **Regroupement des bits**
   On commence par regrouper les données binaires en **blocs de 24 bits**, soit **3 octets (3 × 8 bits)**.
   Le nombre minimum de bits traité est **8 bits (1 octet)**.
   Si le dernier bloc contient moins de 24 bits, on **ajoute des zéros** à la fin pour le compléter — cette étape s'appelle le **padding binaire**.

2. **Découpage en groupes de 6 bits**
   Chaque bloc de 24 bits est ensuite découpé en **4 groupes de 6 bits**.
   Chaque groupe de 6 bits correspond à une **valeur entre 0 et 63**.

3. **Conversion en caractères ASCII**
   Chaque valeur (0 à 63) est remplacée par un **symbole** dans la **table Base64** (A–Z, a–z, 0–9, +, /).

4. **Ajout du remplissage (=)**
   Si les données d'origine ne formaient pas un bloc complet de 24 bits :
   - 1 octet manquant → on ajoute ==
   - 2 octets manquants → on ajoute =
     Ce **remplissage final** permet de conserver un nombre total de caractères multiple de 4**.**

BASE 64

01100110 01101111 01101111

On forme des groupes de 6 bits :

Index	Char	Binary	Index	Char	Binary	Index	Char	Binary	Index	Char	Binary
0	A	0	16	Q	10000	32	g	100000	48	w	110000
1	B	1	17	R	10001	33	h	100001	49	x	110001
2	C	10	18	S	10010	34	i	100010	50	y	110010
3	D	11	19	T	10011	35	j	100011	51	z	110011
4	E	100	20	U	10100	36	k	100100	52	0	110100
5	F	101	21	V	10101	37	l	100101	53	1	110101
6	G	110	22	W	10110	38	m	100110	54	2	110110
7	H	111	23	X	10111	39	n	100111	55	3	110111
8	I	1000	24	Y	11000	40	o	101000	56	4	111000
9	J	1001	25	Z	11001	41	p	101001	57	5	111001
10	K	1010	26	a	11010	42	q	101010	58	6	111010
11	L	1011	27	b	11011	43	r	101011	59	7	111011
12	M	1100	28	c	11100	44	s	101100	60	8	111100
13	N	1101	29	d	11101	45	t	101101	61	9	111101
14	O	1110	30	e	11110	46	u	101110	62	+	111110
15	P	1111	31	f	11111	47	v	101111	63	/	111111

011001 100110 111101 101111

Z m 9 v

BASE 64

01101100 11100000

On complète par des zéros, le derniers groupes de 000000 est encodé par le signe =

Index	Char	Binary	Index	Char	Binary	Index	Char	Binary	Index	Char	Binary
0	A	0	16	Q	10000	32	g	100000	48	w	110000
1	B	1	17	R	10001	33	h	100001	49	x	110001
2	C	10	18	S	10010	34	i	100010	50	y	110010
3	D	11	19	T	10011	35	j	100011	51	z	110011
4	E	100	20	U	10100	36	k	100100	52	0	110100
5	F	101	21	V	10101	37	l	100101	53	1	110101
6	G	110	22	W	10110	38	m	100110	54	2	110110
7	H	111	23	X	10111	39	n	100111	55	3	110111
8	I	1000	24	Y	11000	40	o	101000	56	4	111000
9	J	1001	25	Z	11001	41	p	101001	57	5	111001
10	K	1010	26	a	11010	42	q	101010	58	6	111010
11	L	1011	27	b	11011	43	r	101011	59	7	111011
12	M	1100	28	c	11100	44	s	101100	60	8	111100
13	N	1101	29	d	11101	45	t	101101	61	9	111101
14	O	1110	30	e	11110	46	u	101110	62	+	111110
15	P	1111	31	f	11111	47	v	101111	63	/	111111

011011 001110 000000 000000

b O A =