

Cas 4 : point de code de longueur supérieur à 16 bits :

- encodage sur 4 x 8 bits
- utilisation des préfixes : 11110 puis 10, 10 et 10

point de code en binaire	000u unuu zzzz yyxx xxxx 111 0unu 10uu zzzz 10yy yyy 10xx xxxx
valeur en UTF-8	

2. Exemples

	point de code en binaire	valeur en UTF-8	remarque
A	0100 0001	0100 0001	(cas 1) 1x8 bits
Q	0011 1010 1001	1100 1110 1010 1001	(cas 2) 2x8 bits
X	0110 0101 1000 0111	1110 0110 1001 0110 1000 0111	(cas 3) 3x8 bits
U	0001 0000 0011 1000 0100	1111 0000 1001 0000 1000 1110 1000 0100	(cas 4) 4x8 bits

Remarques UTF-8 est optimisé pour les caractères occidentaux (utilisation de 1 octet) mais pas pour les caractères asiatiques (utilisation de 3 octets).

3. Exercice d'application

ACTIVITÉ 2

Compléter le tableau ci-dessous :

caractère	point de code	UTF-8
☐	0000 0001 0000 0010 0100	
N	0000 0000 0000 0100 1110	
\$		1100 0101 1001 1111
☐	0001 1101 0011 1101	

1. Encodage de textes

Comment la lettre 'Z' est-elle stockée dans la mémoire de l'ordinateur ? Comment la différencier de la lettre 'A' ?

Pour répondre à ces questions, il a fallu créer une **table de conversion** entre un caractère et sa représentation en binaire.

Première table utilisée (début 1960) : **ASCII**

Decimal Hex Char	Decimal Hex Char	Decimal Hex Char
0 (NULL)	32 (SPACE)	64 @
1 (START OF HEADING)	33 !	65 A
2 (START OF TEXT)	34 "	66 B
3 (END OF TEXT)	35 #	67 C
4 (END OF TRANSMISSION)	36 \$	68 D
5 (ENQUIRY)	37 %	69 E
6 (ACKNOWLEDGE)	38 &	70 F
7 (BELL)	39 '	71 G
8 (BACKSPACE)	40 (72 H
9 (HORIZONTAL TAB)	41)	73 I
10 (LINE FEED)	42 *	74 J
11 (VERTICAL TAB)	43 +	75 K
12 (FORM FEED)	44 ,	76 L
13 (CARRIAGE RETURN)	45 -	77 M
14 (SHIFT OUT)	46 .	78 N
15 (SHIFT IN)	47 /	79 O
16 (DATA LINK ESCAPE)	48 0	80 P
17 (DEVICE CONTROL 1)	49 1	81 Q
18 (DEVICE CONTROL 2)	50 2	82 R
19 (DEVICE CONTROL 3)	51 3	83 S
20 (DEVICE CONTROL 4)	52 4	84 T
21 (NEGATIVE ACKNOWLEDGE)	53 5	85 U
22 (SYNCHRONOUS IDLE)	54 6	86 V
23 (END OF TRANS. BLOCK)	55 7	87 W
24 (CANCEL)	56 8	88 X
25 (END OF MEDIUM)	57 9	89 Y
26 (SUBSTITUTE)	58 :	90 Z
27 (ESCAPE)	59 ;	91 [
28 (FILE SEPARATOR)	60 <	92 \
29 (GROUP SEPARATOR)	61 =	93]
30 (RECORD SEPARATOR)	62 >	94 ^
31 (UNIT SEPARATOR)	63 ?	95 _

60	96	127
a	7F	7E
b	77	7D
c	76	7C
d	6F	7B
e	6E	7A
f	6D	79
g	6C	78
h	6B	77
i	6A	76
j	69	75
k	68	74
l	67	73
m	66	72
n	65	71
o	64	70
p	63	6F
q	62	6E
r	61	6D
s	5F	6C
t	5E	6B
u	5D	6A
v	5C	69
w	5B	68
x	5A	67
y	59	66
z	58	65
{	57	64
	56	63
}	55	62
~	54	61
[DEL]	53	60

ACTIVITÉ 1

Quelle est la représentation en **décimal** du caractère Z ? en **binair** ?

Quelle sont celles de la caractère z ? Comment représenter le caractère é ?

CORRECTION

Z est représenté par le nombre décimal $(90)_{10}$. Ce qui donne en binaire $(1011010)_2$ car $90 = 64 + 16 + 8 + 2 = 2^6 + 2^4 + 2^3 + 2^1$

z est représenté par le nombre décimal $(122)_{10}$. Ce qui donne en binaire $(1111010)_2$ car $122 = 64 + 32 + 16 + 8 + 2 = 2^6 + 2^5 + 2^4 + 2^3 + 2^1$

Impossible avec la table ASCII. C'est pour cela qu'à la fin des années 1960 ont été introduites d'autres tables (par exemple ISO-8859).

Conversions texte ↔ binaire

La norme Unicode (début 1990) permet une correspondance entre chaque caractère de l'*humanité* utilisé dans le monde (passé et présent) et un nombre entier appelé **point de code**.

Le point de code d'une taille maximale de 32 bits peut être représenté par des nombres binaires de tailles différentes :

- 1 à 4 paquets de taille 8 bits : UTF-8
- 1 à 2 paquets de taille 16 bits : UTF-16
- 1 paquet de taille 32 bits : UTF-32.

Dans la suite de ce document, nous allons étudier *uniquement* l'encodage en UTF-8 des caractères suivants :

caractère	origine	point de code décimal	point de code binaire
A		65	0100 0001
Ω	lettre grecque	937	0011 1010 1001
文	idéogramme chinois	25 991	0110 0101 1000 0111
	ougaritique (Syrie) des années -1400 à -1100	66 436	0001 0000 0011 1000 0100

Méthode UTF-8 4 cas en fonction de la longueur du point de code :

Cas 1 : point de code de longueur **7 bits ou moins** :

- encodage sur 7 bits identique au point de code
- complété par des 0 à gauche pour atteindre une taille de 8 bits (*padding*)

point de code en binaire	valeur en UTF-8
0xxx xxxx	0xxx xxxx

Cas 2 : point de code de longueur **8 bits à 11 bits** :

- encodage sur 2×8 bits
- utilisation des préfixes : 110 puis 10

point de code en binaire	valeur en UTF-8
0yyy yyxx xxxx	110y yyyy 10xx xxxx

Cas 3 : point de code de longueur **12 bits à 16 bits** :

- encodage sur 3×8 bits
- utilisation des préfixes : 1110 puis 10 et 10

point de code en binaire	valeur en UTF-8
zzzz yyyy yyxx xxxx	1110 zzzz 10yy yyyy 10xx xxxx