

# NSI 1ère - Données

Représentation d'un texte en machine

---

QK

# Représentation d'un texte en machine

---

# Un caractère ?

Comment enregistrer, de manière optimale, du texte en mémoire ?

De combien de symboles a-t-on besoin ?

- 26 lettres dans l'alphabet, 52 avec les majuscules.
- 10 chiffres 0123456789
- Un peu de ponctuation : , ; : ! ? . / \* \$ - + = ( ) [ ] { } " ' etc.
- Quelques caractères techniques (retour à la ligne, espace etc.)

On dépasse  $2^6 = 64$  mais en se contentant du minimum, on reste en dessous de  $2^7 = 128$ . On peut encoder une table assez vaste avec 7 bits.

Idée d'ASCII (1961) : uniformiser les nombreux encodages incompatibles entre eux.

## ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Figure 1: La table ASCII

## Remarques sur la table précédente

- Tout élément de la table est codé sur 7 bits, 1 octet par caractère suffit ( $2^8 = 256$ )
- Les chiffres commencent à  $30_{16}$ , les majuscules à  $41_{16}$  et les minuscules à  $61_{16}$
- Pour obtenir la notation binaire, on part de l'hexa.  
Premier chiffre : 3 bits, second chiffre 4 bits

$A \rightarrow 41_{16} \rightarrow 4 \times 16 + 1 \rightarrow 0100\ 0001$

$s \rightarrow 73_{16} \rightarrow 7 \times 16 + 3 \rightarrow 0111\ 0011$

- Seulement 95 caractères imprimables, pas de caractère accentués :

!"#\$%&'()\*+,-./0123456789:;<=>?

@ABCDEFGHIJKLMN OPQRSTUVWXYZ[\]^\_

`abcdefghijklmnopqrstu vwxyz{|}~

Les fonctions `chr` et `ord` permettent d'accéder à la table

```
>>> chr(65) # caractère 65 (décimal)
```

```
'A'
```

```
>>> ord('A') # numéro décimal du caractère
```

```
65
```

Comment compléter la table ASCII ?

L'encodage iso-8859-1, dit iso-Latin-1 est apparu en 1986 et correspond à l'Europe de l'ouest. D'autres versions pour les caractères iso-Latin-2 de l'Europe de l'est etc.

- Reprend la table ascii et ajoute les accents au coût d'un octet supplémentaire.
- Encore incomplet : œ et Œ n'y sont pas !  
*Ce qui a contribué à leur disparition de nombreux documents écrits dans les années 90. . .*
- Windows (Windows-1252) et Mac (MacRoman) ont leurs versions  
Échange de documents et développement de logiciels **plus que pénibles.**

Bref, c'est de la merde imparfait.

L'unicode et en particulier **UTF8** vise à résoudre TOUS les problèmes dans UNE norme.

- minimiser l'espace occupé par un caractère
- proposer un encodage adaptable à tous les caractères employés sur terre
- conserver l'ordre de la table ascii de départ

Unicode remonte à 1991, est encore en développement, comporte déjà 137 374 caractères d'une centaine d'écritures dont les idéogrammes, l'alphabet grec etc.

UTF8 est utilisé par 90,5% des sites web en 2017 et dans la majorité des systèmes UNIX (comprenez les serveurs)

## Motivation d'unicode : \$ et £

Les machines des années 1980 étant fournies avec leur propre encodage, une somme d'argent en dollars se voyait attribuer le symbole monétaire \$ aux USA et le symbole £ au royaume uni (symbole monétaire de la livre sterling).

Mais  $1\$ \neq 1\text{£}$  et les confusions étaient fréquentes.

On a ensuite, peu à peu, étendu ce projet à tous les symboles existant.

## Principe simplifié d'UTF8

- Chaque caractère est codé avec une séquence de 1 à 4 octets.
- Un texte encodé en ASCII est encodé de la même manière en UTF8 (sauf exception)
- Les premiers bits indiquent la taille de la séquence :
  - 0xxxxxxx : 1 octet
  - 110xxxxx 10xxxxxxxx : 2 octets
  - 1110xxxx 10xxxxxx 10xxxxxx : 3 octets
  - 11110xxx 1001xxxx 10xxxxxx 10xxxxxx : 4 octets
- On note U+XXXX un caractère encodé en UTF8
- La taille est variable (gênant pour les développeurs novices), l'espace en mémoire est parfois important
- Un caractère peut avoir plusieurs représentations → problèmes de sécurité informatique : certaines opérations interdites sont filtrées en reconnaissant des caractères. Ce problème est globalement résolu.

## Python 2 et l'encodage des caractères

**Python 2** supporte bien UTF8 à *condition de lui demander*.

**Sans quoi le premier accent va faire planter python 2.**

On trouve souvent dans l'entête d'un fichier .py :

```
#!/usr/bin/env python  
# -*- coding: utf-8 -*-
```

qui signifient :

- Exécute ce fichier avec python, situé dans le dossier /usr/bin/env
- **l'encodage du fichier est en utf8**

Python 3 supporte nativement utf8, on peut se passer de cette précision

On utilise les fonctions `chr` et `ord`

- `chr(entier)` retourne le caractère encodé par cet entier en utf-8
- `ord(caractère)` retourne l'encodage utf-8 de ce caractère.

**les fonctions `chr` et `ord` supportent unicode :)**

```
>>> chr(82)
```

```
'R'
```

```
>>> ord('R')
```

```
82
```

# Martine écrit en UTF-8



## WHAT ?

- La lettre **é** a été *encodée* en **UTF-8** (parce que 2 caractères sont affichés)  
En mémoire elle occupe 2 octets (elle n'est pas dans la table ascii)
- Ces deux octets ont été *décodés* en **iso-latin1** (1 octet par caractère).