S2 - Ch. 5 : Représentation d'un texte en machine

Cours

Une chaîne de caractères est une suite ordonnée de caractères. Ces caractères peuvent être :
- des lettres minuscules ou majuscules ;
- des symboles de ponctuation ou autres ;
- des chiffres.

En machine, les caractères, comme toute autre *information*, est codée de façon numérique par un nombre binaire.
Le problème qui se pose est que les lettres de l'alphabet, par exemple, ne sont pas les mêmes dans toutes les langues et des caractères. Il faut pouvoir coder des caractères aussi divers que : é, à, ô, ß, Δ,...
Différents systèmes d'**encodage** des caractères existent.

À retenir

Une représentation informatique des caractères nécessite d'associer à chaque caractère une unique séquence d'octets. Cette conversion s'effectue à l'aide d'une **table de codage** et d'un **encodage**.
- Une table de codage ou jeu de caractères ou charset, associe un entier nommé point de code à un caractère.
- Un encodage associe à un point de code une séquence d'octets.

Pour une table de codage, il peut exister plusieurs encodages : ce sera le cas pour la table Unicode, présentée ci-dessous, pour laquelle plusieurs encodages existent (UTF-8, UTF-16, ...).

1. Le code ASCII

Dans les premiers temps de l'informatique, de nombreux systèmes de codage, incompatibles entre eux, existaient.
En 1960, l'organisation internationale de normalisation (International Standard Office : ISO) a créé la norme ASCII (American Standard Code for Information Interchange) pour écrire des textes en anglais.
La table ASCII fournit la correspondance entre 128 caractères et leur représentation binaire. Les caractères sont numérotés de 0 à 127. Comme $2^7 = 128$, il suffit de 7 bits par caractère. Cependant, un octet est le plus souvent utilisé, le bit de poids fort, inutilisé, étant toujours égal à 0.
Dans le code ASCII, les codes des lettres minuscules et des lettres majuscules diffèrent d'un bit, le cinquième. Par exemple "G" est codé par $71_{10} = 0100\ 0111_2$ et "g" est codé par $103_{10} = 0101\ 0111_2$. Cela revient à ajouter $32 = 2^5$ pour passer du code de la majuscule au code de la minuscule.
Les chiffres sont codés par le nombre binaire $0011\ XXXX_2$ où $XXXX_2$ est la valeur du chiffre en binaire. Par exemple 5 est codé par $0011\ 0101_2$.
Le code ASCII d'un caractère étant composé de 8 bits, il peut aussi être représenté par un nombre hexadécimal à deux chiffres.

Voici la table ASCII complète :

Dec	Hex	Binary	HTML	Char
0	00	00000000	&#0;	NUL
1	01	00000001	&#1;	SOH
2	02	00000010	&#2;	STX
3	03	00000011	&#3;	ETX
4	04	00000100	&#4;	EOT
5	05	00000101	&#5;	ENQ
6	06	00000110	&#6;	ACK
7	07	00000111	&#7;	BEL
8	08	00001000	&#8;	BS
9	09	00001001		HT
10	0A	00001010
LF
11	0B	00001011	&#11;	VT
12	0C	00001100		FF
13	0D	00001101
CR
14	0E	00001110	&#14;	SO
15	0F	00001111	&#15;	SI
16	10	00010000	&#16;	DLE
17	11	00010001	&#17;	DC1
18	12	00010010	&#18;	DC2
19	13	00010011	&#19;	DC3
20	14	00010100	&#20;	DC4
21	15	00010101	&#21;	NAK
22	16	00010110	&#22;	SYN
23	17	00010111	&#23;	ETB
24	18	00011000	&#24;	CAN
25	19	00011001	&#25;	EM
26	1A	00011010	&#26;	SUB
27	1B	00011011	&#27;	ESC
28	1C	00011100	&#28;	FS
29	1D	00011101	&#29;	GS
30	1E	00011110	&#30;	RS
31	1F	00011111	&#31;	US

Dec	Hex	Binary	HTML	Char
32	20	00100000		space
33	21	00100001	!	!
34	22	00100010	"	"
35	23	00100011	#	#
36	24	00100100	$	$
37	25	00100101	%	%
38	26	00100110	&	&
39	27	00100111	'	'
40	28	00101000	(	(
41	29	00101001	)	)
42	2A	00101010	*	*
43	2B	00101011	+	+
44	2C	00101100	,	,
45	2D	00101101	-	-
46	2E	00101110	.	.
47	2F	00101111	/	/
48	30	00110000	0	0
49	31	00110001	1	1
50	32	00110010	2	2
51	33	00110011	3	3
52	34	00110100	4	4
53	35	00110101	5	5
54	36	00110110	6	6
55	37	00110111	7	7
56	38	00111000	8	8
57	39	00111001	9	9
58	3A	00111010	:	:
59	3B	00111011	;	;
60	3C	00111100	<	<
61	3D	00111101	=	=
62	3E	00111110	>	>
63	3F	00111111	?	?

Dec	Hex	Binary	HTML	Char
64	40	01000000	@	@
65	41	01000001	A	A
66	42	01000010	B	B
67	43	01000011	C	C
68	44	01000100	D	D
69	45	01000101	E	E
70	46	01000110	F	F
71	47	01000111	G	G
72	48	01001000	H	H
73	49	01001001	I	I
74	4A	01001010	J	J
75	4B	01001011	K	K
76	4C	01001100	L	L
77	4D	01001101	M	M
78	4E	01001110	N	N
79	4F	01001111	O	O
80	50	01010000	P	P
81	51	01010001	Q	Q
82	52	01010010	R	R
83	53	01010011	S	S
84	54	01010100	T	T
85	55	01010101	U	U
86	56	01010110	V	V
87	57	01010111	W	W
88	58	01011000	X	X
89	59	01011001	Y	Y
90	5A	01011010	Z	Z
91	5B	01011011	[	[
92	5C	01011100	\	\
93	5D	01011101	]	]
94	5E	01011110	^	^
95	5F	01011111	_	_

Dec	Hex	Binary	HTML	Char
96	60	01100000	`	`
97	61	01100001	a	a
98	62	01100010	b	b
99	63	01100011	c	c
100	64	01100100	d	d
101	65	01100101	e	e
102	66	01100110	f	f
103	67	01100111	g	g
104	68	01101000	h	h
105	69	01101001	i	i
106	6A	01101010	j	j
107	6B	01101011	k	k
108	6C	01101100	l	l
109	6D	01101101	m	m
110	6E	01101110	n	n
111	6F	01101111	o	o
112	70	01110000	p	p
113	71	01110001	q	q
114	72	01110010	r	r
115	73	01110011	s	s
116	74	01110100	t	t
117	75	01110101	u	u
118	76	01110110	v	v
119	77	01110111	w	w
120	78	01111000	x	x
121	79	01111001	y	y
122	7A	01111010	z	z
123	7B	01111011	{	{
124	7C	01111100			|
125	7D	01111101	}	}
126	7E	01111110	~	~
127	7F	01111111	&#127;	DEL

ASCII-Tables.com

Le code ASCII est suffisant pour écrire un texte en anglais ou pour écrire un programme informatique et il est encore très largement utilisé de nos jours, car il a l'avantage d'être léger.
Cependant, il est insuffisant pour représenter d'autres langues que l'anglais : pas de caractères accentués, de c-cédille, pas de caractères grecs, hébreux, arabes, chinois, ...

2. Le code ISO-8859-1

Pour encoder les langues européennes occidentales, plusieurs extensions du code ASCII ont été définies par l'ISO, comme notamment la norme ISO-8859-1 (appelée aussi ISO-Latin-1). Avec cette norme :

- le codage des caractères présents dans la table ASCII est conservé ;
- chaque caractère est toujours codé sur un octet.

On exploite le bit de poids fort inutilisé par le codage ASCII : cela permet de coder $2^8 = 256$ caractères, soir deux fois plus qu'avec le code ASCII.

3. Le code Unicode

Ces extensions du code ASCII ne suffisent évidemment pas à encoder les caractères des langues non latines. Il a donc fallu créer une autre norme internationale : la norme **Unicode**, apparue au début des années 90. Dans sa version 14.0 publiée en septembre 2021, la table Unicode compte 144 697 caractères couvrant plus de 150 écritures.
Unicode est une table qui regroupe tous les caractères existant au monde, mais ne s'occupe pas de la façon dont les caractères sont codés dans la machine. Il existe pour cela plusieurs formats différents, le plus répandu étant l'encodage **UTF-8**.
UTF-8 est un code à taille variable dans lequel les caractères sont représentés sur 1, 2, 3 ou 4 octets.
Les 128 premiers caractères de la table UTF-8 sont compatibles avec le codage ASCII. Ainsi le codage UTF-8 d'un texte ne comportant que des caractères présents dans la table ASCII sera le même que le codage ASCII de ce texte.
Ce ne sera pas vrai pour un texte ISO-8859-1.

Il importe donc, quand on veut décoder un texte, de savoir quel est le codage utilisé sous peine de décoder improprement les caractères, ce qui arrive par exemple lorsque l'encodage d'une page web n'est pas bien reconnu par le navigateur :

Voici une page encod�e en iso-8859-1 et affich�e en UTF-8.

Les caract�res accentu�s sont mal retranscrits !

On représente en général un code UTF-8 sous la forme `U+XXXX` où XXXX est un nombre écrit en hexadécimal. En voici quelques exemples :

Code UTF-8	Caractère	Écriture
U+FEBE	ﺾ	Arabe
U+05E6	צ	Hébreu
U+30CD	ネ	Japonais katakana
U+4E7B	箸	Chinois

Le site `unicode-table.com/fr` liste tous les caractères de la table Unicode.

Remarque

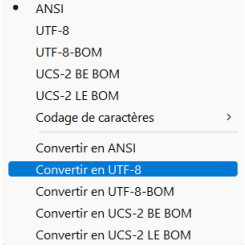
Remarque Depuis sa version 3, Python utilise l'encodage UTF-8 pour les chaînes de caractères.
Nous disposons en Python de deux fonctions liées à ce codage :

- `chr(x)` : retourne le caractère codé par l'entier x écrit en base 10 ;
- `ord(c)` : retourne l'entier correspondant au caractère c (de type str).

```
>>> chr(244)
'ô'
>>> ord("")
8364
```

4. Convertir un fichier d'un encodage à un autre

Pour convertir un fichier d'un encodage à un autre, on peut utiliser l'éditeur Notepad++ dans lequel on trouve un menu "Encodage" comprenant par exemple une commande "convertir en UTF-8".



5. Ouverture d'un fichier texte en Python

Une bonne compréhension du problème de l'encodage des caractères permet d'éviter certaines erreurs lors de l'utilisation de fichiers textes en Python.
Considérons par exemple deux fichiers textes `texte_ascii.txt` et `texte_utf-8.txt` contenant le même texte, l'un étant encodé en ASCII étendu (c'est-à-dire en ISO-8859-1) et l'autre en UTF-8.
Le programme ci-dessous ouvre ces fichiers l'un après l'autre et affiche leur contenu dans la console interactive.

```
from io import open

f = open("texte_ascii.txt")
for ligne in f.readlines():
    print(ligne)
f.close()

print("-----")

f = open("texte_utf-8.txt")
for ligne in f.readlines():
    print(ligne)
f.close()
```

Tout d'abord, le programme est exécuté sous Windows :

```
Ceci est un fichier texte

encodé au format ASCII étendu

Voici quelques mots avec des caractères spéciaux

comme par exemple forçat ou encore Niño ...
-----
Ceci est un fichier texte

encodÃĭ au format UTF-8 Ãĭtendu

Voici quelques mots avec des caractÃĺres spÃĭciaux

comme par exemple forÃğat ou encore NiÃśo ...
```

Et maintenant sous Linux :

```
Traceback (most recent call last):
  File "/usr/lib/python3.10/idlelib/run.py", line 578, in runcode
    exec(code, self.locals)
  File "/home/fabrice/windows/lecture_fichier.py", line 4, in <module>
    for ligne in f.readlines():
  File "/usr/lib/python3.10/codecs.py", line 322, in decode
    (result, consumed) = self._buffer_decode(data, self.errors, final)
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xe9 in position 32: invalid ...
```

Pourquoi cette différence ? La fonction `open` de Python essaie de lire le fichier texte en utilisant l'encodage du système d'exploitation dans lequel Python est utilisé.

— Sous Windows, l'encodage du système est un dérivé de ISO-8859-1. Le fichier ASCII est donc bien lu et affiché, le fichier UTF-8, lui, est bien lu, mais les caractères ne sont pas affichés correctement.
— Sous Linux, l'encodage du système est UTF-8. Le fichier ASCII provoque une erreur en lecture car Python s'attend par défaut à lire des caractères encodés en UTF-8.

Pour éviter ce type de problème et assurer la portabilité d'un programme d'un système d'exploitation à un autre, il est préférable de toujours indiquer l'encodage à la fonction `open`. Le programme suivant fonctionnera sous les deux systèmes et provoquera l'affichage attendu.

```
from io import open

f = open("texte_ascii.txt", encoding="ISO-8859-1")
for ligne in f.readlines():
    print(ligne)
f.close()

print("-----")

f = open("texte_utf-8.txt", encoding="UTF-8")
for ligne in f.readlines():
    print(ligne)
f.close()
```

```
Ceci est un fichier texte

encodé au format ASCII étendu

Voici quelques mots avec des caractères spéciaux

comme par exemple forçat ou encore Niño ...
-----
Ceci est un fichier texte

encodé au format UTF-8 étendu

Voici quelques mots avec des caractères spéciaux

comme par exemple forçat ou encore Niño ...
```