Annexe C

Jeux de caractères et encodage

Position du problème

Nous avons vu que l'ordinateur code toutes les informations qu'il manipule en *binaire*. Pour coder les nombres entiers un changement de base suffit, pour les flottants, on utilise une norme (IEEE 754), mais la situation est plus complexe pour représenter les caractères.

Tous les caractères que l'on peut écrire à l'aide d'un ordinateur sont représentés en mémoire par des nombres. On parle d'*encodage*. Le « a » minuscule par exemple est représenté, ou encodé, par le nombre 97. Pour pouvoir afficher ou imprimer un caractère lisible, leurs dessins, appelés *glyphes*, sont stockés dans des catalogues appelés *polices de caractères*. Les logiciels informatiques parcourent ces catalogues pour rechercher le glyphe qui correspond à un nombre. Suivant la police de caractère, on peut ainsi afficher différents aspects du même « a » (97).

Les 128 premiers caractères comprennent les caractères de l'alphabet latin (non altérés [1]), les majuscules et les minuscules, les chiffres arabes, et quelques signes de ponctuation, c'est la fameuse table ASCII [2] (☞ Fig. C.1). Chaque pays a ensuite complété ce jeu initial suivant les besoins de sa propre langue, créant ainsi son propre système d'encodage.

Cette méthode a un fâcheux inconvénient : le caractère « à » français peut alors être représenté par le même nombre que le caractère « å » scandinave dans les deux encodages, ce qui rend impossible l'écriture d'un texte bilingue avec ces deux caractères !

Pour écrire un document en plusieurs langues, le standard nommé Unicode a été développé et maintenu par un consortium [3]. Il permet d'unifier une grande table de correspondance internationale, sans chevauchement entre les caractères. Les catalogues de police se chargent ensuite de fournir des glyphes correspondants.

n°	char	n°	char	n°	char	n°	char
32		56	8	80	P	104	h
33	!	57	9	81	Q	105	i
34	"	58	:	82	R	106	j
35	#	59	;	83	S	107	k
36	$	60	<	84	T	108	l
37	%	61	=	85	U	109	m
38	&	62	>	86	V	110	n
39	'	63	?	87	W	111	o
40	(	64	@	88	X	112	p
41	)	65	A	89	Y	113	q
42	*	66	B	90	Z	114	r
43	+	67	C	91	[	115	s
44	,	68	D	92	\	116	t
45	-	69	E	93	]	117	u
46	.	70	F	94	^	118	v
47	/	71	G	95	_	119	w
48	0	72	H	96	`	120	x
49	1	73	I	97	a	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	{
52	4	76	L	100	d	124	|
53	5	77	M	101	e	125	}
54	6	78	N	102	f	126	~
55	7	79	O	103	g	127	△

Figure C.1 – Table ASCII.

1. C'est-à-dire sans signe diacritique, par exemple les accents, le tréma, la cédille…
2. American Standard Code for Information Interchange
3. Le *Consortium Unicode*.

L'encodage UTF-8

Comme il s'agit de différencier plusieurs centaines de milliers de caractères (on compte plus de 6000 langues dans le monde) il n'est évidemment pas possible de les encoder sur un seul octet.

En fait, la norme Unicode ne fixe aucune règle concernant le nombre d'octets ou de bits à réserver pour l'encodage, mais spécifie seulement la valeur numérique de l'identifiant associé à chaque caractère (☞ Fig. C.2).

Comme la plupart des textes produits en occident utilisent essentiellement la table ASCII qui correspond justement à la partie basse de la table Unicode, l'encodage le plus économique est l'UTF-8 [1] :

- pour les codes 0 à 127 (cas les plus fréquents), l'UTF-8 utilise l'octet de la table ASCII ;
- pour les caractères spéciaux (codes 128 à 2047), quasiment tous nos signes diacritiques, l'UTF-8 utilise 2 octets ;
- pour les caractères spéciaux encore moins courants (codes 2048 à 65535), l'UTF-8 utilise 3 octets ;
- enfin pour les autres (cas rares), l'UTF-8 en utilise 4.

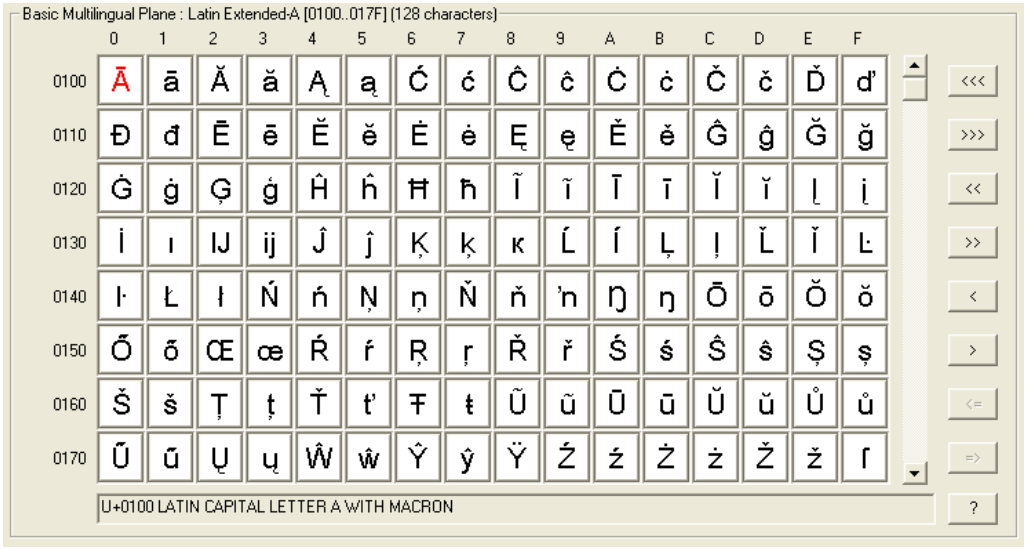

FIGURE C.2 – Extrait de la table Unicode.

Exemple de l'encodage UTF-8 du caractère Unicode « é » :

Symbole	Code décimal	Code hexadécimal	Encodage UTF-8
é	233	e9	C3 A9

Voici trois exemples de caractères spéciaux codés en notation hexadécimale et séparés par le caractère d'échappement de la tabulation :

```
>>> print("\u00e9 \t \u0110 \t \u0152")
é	Đ	Œ
```

Applications aux scripts Python

En Python 3, les chaînes de caractères (le type `str()`) sont des chaînes Unicode. Par ailleurs, puisque les scripts Python que l'on produit avec un éditeur sont eux-mêmes des textes, ils sont suceptibles d'être encodés suivant différents formats. Afin que Python utilise le bon, il est important d'indiquer l'encodage de caractères utilisé. On le précise obligatoirement en 1re ou 2e ligne des sources.

Les encodages les plus fréquents sont [2] :

```
# -*- coding : utf8 -*-
```

ou :

```
# -*- coding : latin1 -*-
```

1. Retour chap. 2, p. 5
2. Notons que `utf8` et `latin1` sont des alias de `utf-8` et `latin-1`.