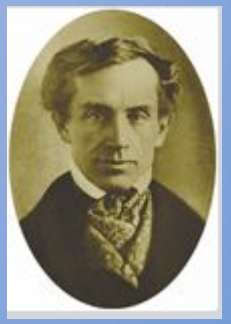
Introduction

Les lettres ? Toutes des nombres !

Dans ce chapitre, nous voyons comment sont représentés les caractères et les textes de toutes les langues du monde.

Nous expliquons pourquoi il existe plusieurs codes tels ASCII, latin-1, latin-2, UTF-32, UTF-8.
Nous présentons ensuite les formats enrichis qui permettent de décrire la forme des caractères et des textes, comme le font les logiciels de traitement de texte.
Un exemple de format enrichi est le langage HTML.
Nous nous intéressons, dans ce chapitre, à la représentation des textes, c'est-à-dire des suites de *caractères*, éventuellement enrichies d'informations typographiques.



Samuel Morse (1791 - 1872) est l'inventeur d'un code, dans lequel chaque lettre est exprimée par une alternance de sons brefs symbolisés par « . » et longs « - », utilisé pour télégraphier des textes. La lettre « a » y est exprimée par les sons « . - », la lettre « b » par les sons « - ... », etc.
Artiste peintre, Samuel Morse s'est intéressé aux télécommunications après qu'en 1925, un message lui annonçant que sa femme était malade ne lui est pas parvenu à temps.
Le code morse est à références de longueurs variables, mais ce n'est pas un code préfixe.

La représentation des caractères

- Un texte est une suite de caractères
- Les caractères sont composés entre autres de lettres minuscules et majuscules, de chiffres, de signes de ponctuation et de symboles mathématiques.
- Pour représenter ces caractères, on attribue un nombre à chacun.

La représentation des caractères

- *Code ASCII : American Standard Code for Information Interchange*

  codage de 127 caractères

Décimal	Octal	Hex	Binaire	Caractère	
000	000	00	00000000	NUL	(Null char.)
001	001	01	00000001	SOH	(Start of Header)
002	002	02	00000010	STX	(Start of Text)
003	003	03	00000011	ETX	(End of Text)
004	004	04	00000100	EOT	(End of Transmission)
005	005	05	00000101	ENQ	(Enquiry)
006	006	06	00000110	ACK	(Acknowledgment)
007	007	07	00000111	BEL	(Bell)
008	010	08	00001000	BS	(Backspace)
009	011	09	00001001	HT	(Horizontal Tab)
010	012	0A	00001010	LF	(Line Feed)
011	013	0B	00001011	VT	(Vertical Tab)
012	014	0C	00001100	FF	(Form Feed)
013	015	0D	00001101	CR	(Carriage Return)
014	016	0E	00001110	SO	(Shift Out)
015	017	0F	00001111	SI	(Shift In)
016	020	10	00010000	DLE	(Data Link Escape)
017	021	11	00010001	DC1	(XON)(Device Control 1)
018	022	12	00010010	DC2	(Device Control 2)
019	023	13	00010011	DC3	(XOFF)(Device Control 3)
020	024	14	00010100	DC4	(Device Control 4)

La représentation des caractères

Table ASCII

Décimal	Octal	Hex	Binaire	Caractère
021	025	15	00010101	NAK (Negative Acknowledgement)
022	026	16	00010110	SYN (Synchronous Idle)
023	027	17	00010111	ETB (End of Trans. Block)
024	030	18	00011000	CAN (Cancel)
025	031	19	00011001	EM (End of Medium)
026	032	1A	00011010	SUB (Substitute)
027	033	1B	00011011	ESC (Escape)
028	034	1C	00011100	FS (File Separator)
029	035	1D	00011101	GS (Group Separator)
030	036	1E	00011110	RS (Request to Send)(Record Separator)
031	037	1F	00011111	US (Unit Separator)
032	040	20	00100000	SP (Space)
033	041	21	00100001	! (exclamation mark)
034	042	22	00100010	" (double quote)
035	043	23	00100011	# (number sign)
036	044	24	00100100	$ (dollar sign)
037	045	25	00100101	% (percent)
038	046	26	00100110	& (ampersand)
039	047	27	00100111	' (single quote)
040	050	28	00101000	((left opening parenthesis)

La représentation des caractères

Table ASCII

Décimal	Octal	Hex	Binaire	Caractère
041	051	29	00101001	) (right closing parenthesis)
042	052	2A	00101010	* (asterisk)
043	053	2B	00101011	+ (plus)
044	054	2C	00101100	, (comma)
045	055	2D	00101101	- (minus or dash)
046	056	2E	00101110	. (dot)
047	057	2F	00101111	/ (forward slash)
048	060	30	00110000	0
049	061	31	00110001	1
050	062	32	00110010	2
051	063	33	00110011	3
052	064	34	00110100	4
053	065	35	00110101	5
054	066	36	00110110	6
055	067	37	00110111	7
056	070	38	00111000	8
057	071	39	00111001	9
058	072	3A	00111010	: (colon)
059	073	3B	00111011	; (semi-colon)
060	074	3C	00111100	< (less than sign)

La représentation des caractères

Table ASCII

Décimal	Octal	Hex	Binaire	Caractère
061	075	3D	00111101	= (equal sign)
062	076	3E	00111110	> (greater than sign)
063	077	3F	00111111	? (question mark)
064	100	40	01000000	@ (AT symbol)
065	101	41	01000001	A
066	102	42	01000010	B
067	103	43	01000011	C
068	104	44	01000100	D
069	105	45	01000101	E
070	106	46	01000110	F
071	107	47	01000111	G
072	110	48	01001000	H
073	111	49	01001001	I
074	112	4A	01001010	J
075	113	4B	01001011	K
076	114	4C	01001100	L
077	115	4D	01001101	M
078	116	4E	01001110	N
079	117	4F	01001111	O
080	120	50	01010000	P

La représentation des caractères

Table ASCII

Décimal	Octal	Hex	Binaire	Caractère	
081	121	51	01010001	Q	
082	122	52	01010010	R	
083	123	53	01010011	S	
084	124	54	01010100	T	
085	125	55	01010101	U	
086	126	56	01010110	V	
087	127	57	01010111	W	
088	130	58	01011000	X	
089	131	59	01011001	Y	
090	132	5A	01011010	Z	
091	133	5B	01011011	[	(left opening bracket)
092	134	5C	01011100	\	(back slash)
093	135	5D	01011101	]	(right closing bracket)
094	136	5E	01011110	^	(caret cirumflex)
095	137	5F	01011111	_	(underscore)
096	140	60	01100000	`	
097	141	61	01100001	a	
098	142	62	01100010	b	
099	143	63	01100011	c	
100	144	64	01100100	d	

La représentation des caractères

Table ASCII

Décimal	Octal	Hex	Binaire	Caractère
101	145	65	01100101	e
102	146	66	01100110	f
103	147	67	01100111	g
104	150	68	01101000	h
105	151	69	01101001	i
106	152	6A	01101010	j
107	153	6B	01101011	k
108	154	6C	01101100	l
109	155	6D	01101101	m
110	156	6E	01101110	n
111	157	6F	01101111	o
112	160	70	01110000	p
113	161	71	01110001	q
114	162	72	01110010	r
115	163	73	01110011	s
116	164	74	01110100	t
117	165	75	01110101	u
118	166	76	01110110	v
119	167	77	01110111	w
120	170	78	01111000	x
121	171	79	01111001	y
122	172	7A	01111010	z
123	173	7B	01111011	{ (left opening brace)
124	174	7C	01111100	| (vertical bar)
125	175	7D	01111101	} (right closing brace)
126	176	7E	01111110	~ (tilde)
127	177	7F	01111111	DEL (delete)

La représentation des caractères

- Code ASCII est prévu initialement pour l'anglais
- Peu adapté pour d'autres langues dont le français qui utilisent des caractères latins (accents, cédille, ….)

- extension du code ASCII, le code *latin-1*, qui contient 191 caractères.

La représentation des caractères

Pour représenter les textes écrits en grec, russe, chinois, japonais, coréen, etc., il a fallu proposer un format universel : ***Unicode***.
Unicode recense près de 110 000 caractères et associe un nom et un numéro à chacun. A priori, ce numéro se code sur 32 bits.

Unicode existe en plusieurs déclinaisons, parmi lesquelles UTF-32, dans laquelle chaque caractère est ainsi exprimé sur 32 bits, et UTF-8, dans laquelle les caractères les plus courants sont exprimés sur 8 bits et les moins courants sur 16, 32 ou 64 bits, utilisant la notion de compression.

La représentation des caractères

Le format UTF-8 a vocation à devenir le standard, mais il ne l'est pas encore : malgré les efforts des comités de normalisation, l'humanité n'a pas encore réussi à se doter d'un format universellement accepté, si bien qu'il est parfois nécessaire de traduire un texte d'UTF-8 en latin-1 ou de latin-2 en UTF-8.

Quand cette traduction n'est pas bien faite, les caractères accentués sont remplacés par des caractères bizarres.

La représentation des caractères

SAVOIR-FAIRE **Trouver la représentation en ASCII binaire d'un texte**

En utilisant une table, on cherche le code ASCII de chaque caractère. Puis on traduit chacun de ces nombres en représentation binaire.

Exercice 1

Trouver la représentation binaire en ASCII du texte « Je pense, donc je suis. »

La représentation des caractères

Correction exercice 1

On traduit le texte caractère par caractère à partir de la table ASCII : 74, 101, 32, 112, 101, 110, 115, 101, 44, 32, 100, 111, 110, 99, 32, 106, 101, 32, 115, 117, 105, 115, 46.

On exprime ensuite chacun de ces nombres en binaire sur huit bits :

01001010 01100101 00100000 01110000 01100101 01101110 01110011 01100101 00101100 00100000 01100100 01101111 01101110 01100011 00100000 01101010 01100101 00100000 01110011 01110101 01101001 01110011 00101110.

Remarques :

- On peut utiliser un éditeur hexadécimal comme Frhed pour obtenir la traduction du texte en ASCII hexadécimal
- Puis utiliser par exemple la calculatrice Windows en mode programmeur pour obtenir la traduction des octets en binaire.

La représentation des caractères

SAVOIR-FAIRE Décoder un texte représenté en ASCII binaire

On découpe la suite de bits en octets, on traduit chaque octet en décimal, puis on cherche en utilisant une table, le caractère exprimé par chacun de ces nombres.

Exercice 2

Trouver le texte représenté en ASCII binaire par la suite de bits 011000110110111101100100011000010110011101100101001000000100000101010011010000011010010010100100.

La représentation des caractères

Correction exercice 2

On commence par découper la suite de bits en octets :

01100011 01101111 01100100 01100001 01100111 01100101 00100000 01000001

01010011 01000011 01001001 01001001

Chaque octet représente un nombre entier (en base 10) :

99 111 100 97 103 101 32 65 83 67 73 73

On cherche ensuite dans la table des codes ASCII la traduction de chacun de ces nombres en une lettre :

On obtient alors la phrase : « codage ASCII ».

.

La représentation des textes enrichis

Les textes en ASCII ou en Unicode sont simplement des suites de caractères.

Les éditeurs de texte sont les logiciels qui manipulent ces suites de caractères.

Toutefois, quand on écrit un texte, on peut souhaiter lui donner une forme spéciale, plus jolie, plus lisible, comme le fait un imprimeur.

On peut jouer sur :

- la police de caractères – Times, Courier, etc. –,
- la taille des caractères – 11 points, 12 points, etc. –,
- leur forme – romain, italique, etc. –,
- leur graisse – maigre, gras, etc.

La représentation des textes enrichis

On peut aussi souhaiter découper un texte en chapitres et mettre en valeur les titres des chapitres, etc. Or, les seules caractéristiques que l'on puisse exprimer avec un code comme l'ASCII, par exemple, sont la casse d'une lettre – minuscule ou majuscule – et le découpage en paragraphes, grâce au symbole retour chariot.

Les traitements de texte sont les logiciels qui permettent ces mises en pages plus élaborées.

La représentation des textes enrichis

Ceci a amené à enrichir ces formats, de manière à :

1. *qualifier* certaines parties du texte, par exemple en mettant certaines parties en gras ou en italique,
2. structurer le texte en *divisions* : un texte n'est pas uniquement une suite de paragraphes, mais est hiérarchisé en parties, chapitres, sec- tions, sous-sections…,
3. présenter certaines informations sous forme de listes et de tables,
4. permettre de faire *référence* à d'autres textes,
5. donner des informations sur le texte : son titre, son ou ses auteur(s), sa date de création, sa langue, des mots-clés utilisés pour le recher- cher parmi plusieurs textes, etc.

Ces informations sur le texte, et non du texte, sont appelées des ***méta-données***.

L'un de ces formats enrichis, qui est utilisé en particulier pour écrire des pages web est appelé le format **HTML**. (**H**yper **T**ext **M**arkup **L**angage).

La représentation des textes enrichis

Exercice 3

On considère trois fichiers contenant le même texte :

- cigale1.txt
- cigale2.txt
- cigale.odt

1. Ouvrir ces fichiers à l'aide de l'éditeur de textes NOTEPAD. Voyez-vous une différence entre ces fichiers ?
2. Ouvrir ces fichiers à l'aide de l'éditeur de textes WORDPAD. Voyez-vous une différence entre ces fichiers ?
3. Ouvrir ces trois fichiers à l'aide du logiciel OPENOFFICE . Voyez-vous une différence ?
4. Quelle est la taille en octets de ces fichiers ?
5. Ouvrir les trois fichiers avec l'éditeur hexadécimal FRHED. Reconnaissez-vous le texte contenu dans chacun des trois fichiers ?
6. Rechercher la signification des extensions .txt et .odt.
7. Quelle différence y a-t-il entre les fichiers cigale1.txt et cigale2.txt ?

La représentation des textes enrichis

Correction exercice 3

1.On ne voit pas de différence avec NOTEPAD pour les fichiers cigale1,txt et cigale2,txt Par contre, le contenu du fichier cigale.odt contient des caractères non alphanumériques.

2.On ne voit pas de différence avec WORDPAD pour les 3 fichiers.

3.On ne voit pas de différence non plus avec OPENOFFICE.

4.cigale1.txt et cigale2,txt : 1 Ko cigale.odt : 16 Ko.

Plus exactement : cigale1.txt : 624 octets cigale2.txt : 648 octets

cigale.odt : 15 433 octets.

5.On identifie bien le texte pour les fichiers cigale1,txt et cigale2.txt mais pas pour le fichier cigale.odt

6..odt est une extension de fichier pour le format « texte formaté » issu du format OpenDocument qui est un format ouvert de données pour les applications bureautiques.

.txt est une extension pour les fichiers textes non formatés.

La représentation des textes enrichis

Correction exercice 3

7.La différence entre les fichiers cigale1.txt et cigale2,txt se situent au niveau des fins de ligne :

- Le fichier cigale1.txt contient en fin de ligne 0x0A (LF : Line Feed) utilisé par le système d'exploitation Unix pour générer un saut de ligne.
- Le fichier cigale2.txt contient en fin de ligne 0D0A (CRLF : Carriage Return Line Feed) utilisé par le système d'exploitation Windows pour générer un saut de ligne.