Le codage numérique du texte

Sources : Wikipédia

Da ASCII code !

En informatique, chaque caractère est identifié par un code unique qui est un entier naturel. La correspondance entre le caractère et son code est appelé un **Charset**. Mais un code n'est pas utilisable tel quel par un ordinateur qui ne comprend que le binaire. Il faut donc encoder les codes en octets (**Encoding**).

Le code ASCII est l'un des plus anciens codes utilisés pour représenter du texte en informatique. ASCII signifie (American Standard Code for Information Interchange).

Il se base sur un tableau contenant les caractères les plus utilisés **en langue anglaise** : les lettres de l'alphabet en majuscule (de A à Z) et en minuscule (de a à z), les dix chiffres arabes (de 0 à 9), les signes de ponctuation (point, virgule, point-virgule, guillemet, parenthèses, etc.), quelques symboles et certains caractères spéciaux invisibles (espace, retour-chariot, tabulation, retour-arrière, etc.).

Les créateurs de ce code ont limité le nombre de ses caractères à 128, c'est-à-dire 2^7, pour qu'ils puissent être codés avec seulement 7 bits : les ordinateurs utilisaient des cases mémoire d'un octet, mais ils réservaient toujours le 8^e bit pour le contrôle de parité (c'est une sécurité pour éviter les erreurs, qui étaient très fréquentes dans les premières mémoires électroniques).

Exemple : Le caractère A est codé en ASCII par le nombre 65 (dans notre système décimal habituel), qui correspond en binaire au nombre 1000001.

Chaque caractère d'un texte codé en ASCII occupe alors un **octet (8 bits)**.
Un texte de **5000** caractères occupe donc **5 ko (kilo-octets)**.

***Activités* :** La phrase : « Enfin ! Je peux t'aider à comprendre ce qui s'est passé. » a une taille de 55 octets (il faut compter les espaces, l'apostrophe, le point final ...).

L'expérience peut être faite en utilisant un éditeur de texte quelconque comme le bloc-notes de Windows, TextEdit sous OSX ou encore kwrite, geany sous Linux. Il suffit d'écrire le texte, puis de l'enregistrer en tant que « texte brut » (le plus souvent avec une extension .txt) et ensuite de vérifier la taille en octets du fichier obtenu (ce qui peut se faire en cliquant d'abord avec le bouton droit sur l'icône du fichier puis sur "Propriétés"). On peut écrire la même chose dans un logiciel de traitement de texte (comme LibreOffice Writer ou Microsoft Word) et se rendre compte que la taille du fichier obtenu n'est pas du tout la même. Y a-t-il une explication ?

Word ajoute la mise en page, la couleur, et tout un tas de détails en plus du texte brut, tandis que les éditeurs de texte brut ne retiennent que le texte.

A l'aide de la table ASCII située à la fin de ce document, coder en binaire la phrase suivante :

« L'an qui vient ! »

01001100 00100111 01100001 01101110 00100000 01110001 01110101 01101001 001000000 01110110 01101001 01100101 01101110 01110100 00100000 00100001

Voici maintenant une exclamation codée en binaire : 01000010 01110010 01100001 01110110 01101111 00100001

Retrouver cette exclamation : Bravo!

Peut-on coder en binaire la phrase « Un âne est-il passé par là ? » à l'aide de la table ? *(Justifier la réponse)* :
Non, à cause des accents

Il va donc falloir étendre la table ASCII pour pouvoir coder les nouveaux caractères. Les mémoires devenant plus fiables et, de nouvelles méthodes plus sûres que le contrôle de parité ayant été inventées, le 8ième bit a pu être utilisé pour coder plus de caractères. Combien le fait d'avoir 8 bits amène-t-il de nouvelles possibilités ?

128

On élimine ainsi l'inconvénient très gênant de ne coder que les lettres non accentuées, ce qui peut suffire en anglais, mais pas dans les autres langues (comme le français et l'espagnol par exemple). On a pu aussi rajouter des caractères typographiques utiles comme des tirets de diverses tailles et sortes.

Par exemple, en français les caractères é, è, ç, à, ù, ô, æ, œ, sont fréquemment utilisés alors qu'ils ne figurent pas dans la table ASCII.

De la difficulté de convenir d'une norme...

Le fait d'utiliser un bit supplémentaire a bien entendu ouvert des possibilités mais malheureusement tous les caractères ne pouvaient être pris en charge. La norme ISO 8859–1 appelée aussi Latin-1 ou Europe occidentale est la première partie d'une norme plus complète appelée ISO 8859 (qui comprend 16 parties) et qui permet de coder tous les caractères des langues européennes. Cette norme ISO 8859–1 permet de coder 191 caractères de l'alphabet latin qui avaient à l'époque été jugés essentiels dans l'écriture. Par exemple la lettre œ n'y figure pas.

Dans les pays occidentaux, cette norme est utilisée par de nombreux systèmes d'exploitation, dont UNIX, Windows. Elle a donné lieu à quelques extensions et adaptations, dont Windows-1252 (appelée ANSI) et ISO 8859-15 (qui prend en compte le symbole € créé après la norme ISO 8859-1). C'est source de grande confusion pour les développeurs de programmes informatiques car un même caractère peut être codé différemment suivant la norme utilisée.

Voici deux tableaux :

ISO/CEI 8859-1

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	positions inutilisées															
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	positions inutilisées															
9x																
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Windows-1252 (CP1252)

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	€		‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ		Ž	
9x		‘	’	“	”	•	–	—	˜	™	š	›	œ		ž	Ÿ
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Un petit site bien utile : http://md5decrypt.net/Outils-conversion/

Le code binaire d'un texte : 01001010 01100101 00100000 01101110 01100101 00100000 01110000 01100101 01110101 01111000 00100000 01110000 01100001 01110011 00101100 00100000 01101010 00100111 01100001 01101001 00100000 01101100 01101001 01100011 01101111 01110010 01101110 01100101 00101110

A l'aide du site, retrouvez le texte contenu dans le code : Je ne peut pas, j'ai licorne

Peut-on dire si ce logiciel utilise la norme ISO 8859 ou la norme Windows 1252 ? *(Justifier la réponse)*

Non, car les caractères utilisés sont identiques dans les deux normes

Trouver une astuce pour savoir laquelle des deux est utilisée ! C'est la norme : Windows-1252

Quand le net s'affole...

Nous avons tous un jour reçu un courriel bizarre ou lu une page web telle que celle-ci :

> Prenons lâ€™exemple typique de la lumiÃ¨re Ã©mise par un phare maritime : elle est dâ€™abord indivisible, son coÃ»t de production Ã©tant alors indÃ©pendant du nombre d'utilisateurs ; elle possÃ¨de une propriÃ©tÃ© de non-rivalitÃ© (elle ne se dÃ©truit pas dans l'usage et peut donc Ãªtre adoptÃ©e par un nombre illimitÃ© d'utilisateurs) ; elle est Ã©galement non excluable car il est impossible dâ€™exclure de lâ€™usage un utilisateur, mÃªme si ce dernier ne contribue pas Ã son financement.

Bien que ceci soit de moins en moins fréquent (nous comprendrons pourquoi), on trouve parfois des phrases dans lesquelles certains caractères sont remplacés par d'autres qui n'ont rien à voir et qui empêchent la lecture et la compréhension du texte. Il s'agit ici d'un problème d'encodage et de décodage. La personne qui écrit le texte utilise une norme différente de celle utilisée par celui qui le lit ! Lorsque c'est un courriel on a la plupart du temps affaire à un spam venant de l'étranger, ce n'est pas sans raison...

Et l'Unicode vint...

Bien entendu, il n'y a pas que les langues européennes qui existent, et la généralisation de l'utilisation d'Internet dans le monde et des échanges interculturels a nécessité une prise en compte d'un nombre beaucoup plus importants de caractères d'écriture (par exemple le mandarin possède plus de 5000 caractères !). De plus certaines confusions étaient possibles comme par exemple lorsqu'était prévu un caractère « signe monétaire », le même texte autorisant aux États-Unis une dépense en dollars pouvait une fois transmis par courrier électronique au Royaume-Uni autoriser la même dépense en livres sterling, sans que quoi que ce soit ait été modifié au texte... La norme Unicode a été créée pour permettre le codage de textes écrits quelque soit le système d'écriture utilisé. On attribue à chaque caractère un nom, une position normative et un bref descriptif qui seront les mêmes quelque soit la plate-forme informatique ou le logiciel utilisés. Un consortium composé d'informaticiens, de chercheurs, de linguistes, de personnalités du monde de la politique... s'occupe donc d'unifier toutes les pratiques en un seul et même système : l'Unicode.

L'Unicode est une table de correspondance Caractère-Code (Charset), et l'UTF-8 est l'encodage correspondant (Encoding) le plus répandu. Maintenant, par défaut, les navigateurs Internet utilisent le codage UTF-8 et les concepteurs de sites pensent de plus en plus à créer leurs pages web en prenant en compte cette même norme. Voilà pourquoi il y a de moins en moins de problèmes de compatibilité.

Une petite expérience :

Ouvrez un navigateur Internet comme Firefox, Internet Explorer, Safari ou Opéra. Dans la barre d'outils du premier on peut voir à « Affichage », « Encodage des caractères » que c'est l'UTF-8 qui est sélectionné par défaut. Changeons ceci et sélectionnons Europe Occidentale (Windows). Les petits caractères désagréables apparaissent. Que s'est-il passé ? En allant dans « Outils », « Informations sur la page », on voit que cette page est encodée en UTF-8. Lorsque le lecteur est lui aussi en UTF-8 tout va bien. Dès qu'on change le paramètre du lecteur (ici, le navigateur), des incompatibilités apparaissent.
En utilisant Internet Explorer, et en allant dans « Affichage », « Source », on obtient ceci :

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
    transitional.dtd">
2  <html lang="fr" xml:lang="fr" dir="ltr" xmlns="http://www.w3.org/1999/xhtml" >
3  <head>
4    <base href="http://interstices.info/"  />
5
6
7    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
8    <meta http-equiv="Cache-Control" content="no-cache" />
9    <meta http-equiv="Pragma" content="no-cache" />
10   <meta http-equiv="Expires" content="-1" />
11   <meta http-equiv="X-UA-Compatible" content="IE=7" />
12   <meta http-equiv="Content-Style-Type" content="text/css" />
13   <meta http-equiv="Content-Script-Type" content="text/javascript" />
14
15
16   <meta name="Generator" content="Jalios JCMS - Copyright (C) Jalios S.A. 2001 - 2011 -
    http://www.jalios.com/" />
17   <meta name="keywords" content="Musique,Creative Commons,Ordinateur et
    société,Autre,Art,Débattre,Informatique" />
18   <meta name="description" content="Découvrir la recherche en informatique Avec le développement
    d'Internet, la numérisation d'œuvres écrites, d'images ou de sons s'est rapidement généralisée. Ce
    phénomène a des conséquences économiques et sociales importantes." />
19
20   <title>Interstices - Œuvres numérisées sur Internet : les enjeux de leur diffusion</title>
```

On peut lire l'entête de la page html visitée avec notamment à la ligne 7 la précision de l'encodage de cette page qui est en UTF-8.
On peut aussi dans « Affichage », « Codage », sélectionner Grec (ISO) et se rendre compte en lisant le texte, que le « à » a été remplacé par un « L » à l'envers dit Gamma.

Une autre petite expérience :
Sous Windows aller dans « Démarrer », « Exécuter », taper « charmap ». Cocher « Affichage avancé », sélectionner « Windows Occidental » (c'est à peu de choses près l'ISO 8859-1) dans « Jeu de caractères » regarder le nombre de caractères proposés, puis sélectionner « Unicode ». Il y a maintenant un très grand nombre de caractères disponibles, beaucoup de langues sont représentées. Sous Linux on pourra utiliser « gucharmap », et pour OSX (Apple) il faut chercher le « palette de caractères ».

Quelques précisions sur l'UTF-8

L'encodage UTF-8 utilise 1, 2, 3 ou 4 octets en respectant certaines règles :

- Un texte en ASCII de base (appelé aussi US-ASCII) est codé de manière identique en UTF-8. On utilise un octet commençant par 0.

Caractère	Point de code (hexadécimal)	Valeur scalaire		Codage UTF-8
		décimal	binaire	binaire
A	U+0041	65	1000001	01000001

- Les octets ne sont pas remplis entièrement. Les bits de poids fort du premier octet forment une suite de 1 indiquant le nombre d'octets utilisés pour coder le caractère. Les octets suivants commencent tous par 10.

Définition du nombre d'octets utilisés

Représentation binaire UTF-8	Signification
`0xxxxxxx`	1 octet codant 1 à 7 bits
`110xxxxx 10xxxxxx`	2 octets codant 8 à 11 bits
`1110xxxx 10xxxxxx 10xxxxxx`	3 octets codant 12 à 16 bits
`11110xxx 10xxxxxx 10xxxxxx 10xxxxxx`	4 octets codant 17 à 21 bits

- Dans la norme ISO 8859-1 le « é » est codé 1110 1001, en UTF-8 on le codera **110***000*11 **10**101001. On pourra remarquer deux choses : le codage ISO s'inscrit dans le codage UTF-8, on a comblé le premier octet avec des zéros (en italique).

Exercice :

Le symbole **€** correspond à la valeur décimale 8364.

- Convertir cette valeur en binaire : 0010 0000 1010 1100
- Combien d'octets doit-on utiliser en UTF-8 pour coder ce nombre convenablement (les moitiés d'octet sont interdites) ? 3 octets codant à 14 bits (donc 16)
- Donner le codage UTF-8 correspondant : **1110**0010 **10**000010 **10**101100

Quelques remarques :

- Ce codage permet de coder tous les caractères de la norme Unicode.
- Les caractères Unicode sont la plupart du temps représentés en hexadécimal. C'est un moyen de simplifier l'écriture en binaire qui devient lourde lorsqu'on manipule plusieurs octets. Il s'agit de la base 16 ce qui signifie que l'on a besoin de 16 symboles : 0 1 2 3 4 5 6 7 8 9 A B C D E F.
  A représentant 10, B, 11... Pour passer du binaire à l'hexadécimal, rien de plus simple, on fait des paquets de quatre bits que l'on représente par un des 16 symboles.
  Par exemple : 1110 1001 qui est le code binaire du é en ISO 8859-1 devient $E9_{16}$ (l'indice signifie que l'on est en base 16) ce que l'on peut retrouver dans le tableau donné plus haut. Il faut noter que la notation en binaire est très peu utilisée sauf pas les électroniciens et par ceux qui travaillent en langage machine.
- Le système de codage UTF-8 permet d'encoder un même caractère de plusieurs manières. Ceci peut poser un problème de sécurité car un programme détectant certaines chaînes de caractères (pour contrer des injections dans les bases de données par exemple), s'il est mal écrit, pourrait alors accepter des séquences nuisibles. En 2001 un virus a attaqué des serveurs http du web.
  Par exemple, le symbole € pourrait être codé sur 4 octets (forme super longue) de la manière suivante : **11110**000 **10**000010 **10**000010 **10**101100. Si elle n'est pas rejetée ou remise sous forme standard ce codage ouvrira une brèche potentielle de sécurité par laquelle on pourra faire passer un virus.

Quel est le nombre maximal de caractères que l'on peut encoder grâce à l'UTF-8 lorsqu'on utilise les quatre octets? 2^21 = 2 097 152

Voici un site très utile à consulter sans modération en complément de ceux proposés en liens :

Arcanapercipio [1] [2]

Décimale	Binaire	Valeur	Explication
000	00000000	NUL	NULL Character
001	00000001	SOH	Start of Header
002	00000010	STX	Start of Text
003	00000011	ETX	End of Text
004	00000100	EOT	End of Transmission
005	00000101	ENQ	Enquiry
006	00000110	ACK	Acknowledgement
007	00000111	BEL	Bell
008	00001000	BS	Backspace
009	00001001	HT	Horizontal Tab
010	00001010	LF	Line Feed
011	00001011	VT	Vertical Tab
012	00001100	FF	Form Feed
013	00001101	CR	Carriage Return
014	00001110	SO	Shift Out
015	00001111	SI	Shift In
016	00010000	DLE	Data Link Escape
017	00010001	DC1	Device Control 1 (XON)
018	00010010	DC2	Device Control 2
019	00010011	DC3	Device Control 3 (XOFF)
020	00010100	DC4	Device Control 4
021	00010101	NAK	Negative Acknowledgement
022	00010110	SYN	Synchronous Idle
023	00010111	ETB	End of Transmission Block
024	00011000	CAN	Cancel
025	00011001	EM	End of Medium
026	00011010	SUB	Substitude
027	00011011	ESC	Escape
028	00011100	FS	File Separator
029	00011101	GS	Group Separator
030	00011110	RS	Record Separator / Request to Send
031	00011111	US	Unit Separator
032	00100000	SP	Space
033	00100001	!	exclamation mark
034	00100010	"	Double quote
035	00100011	#	Number sign / hash sign
036	00100100	$	Dollar sign
037	00100101	%	Pourcent
038	00100110	&	Ampersand
039	00100111	'	Simple quote
040	00101000	(	Left parenthesis / Opening parenthesis
041	00101001	)	Right parenthesis / Closing parenthesis
042	00101010	*	Asterisk
043	00101011	+	Plus
044	00101100	,	Comma
045	00101101	-	Minus / Dash
046	00101110	.	Dot
047	00101111	/	Forward slash
048	00110000	0	
049	00110001	1	
050	00110010	2	
051	00110011	3	
052	00110100	4	
053	00110101	5	
054	00110110	6	
055	00110111	7	
056	00111000	8	
057	00111001	9	
058	00111010	:	Colon
059	00111011	;	Semi-colon
060	00111100	<	Less than
061	00111101	=	Equal sign
062	00111110	>	Greater than
063	00111111	?	Question mark
064	01000000	@	AT symbol
065	01000001	A	
066	01000010	B	
067	01000011	C	
068	01000100	D	
069	01000101	E	
070	01000110	F	
071	01000111	G	
072	01001000	H	
073	01001001	I	
074	01001010	J	
075	01001011	K	
076	01001100	L	
077	01001101	M	
078	01001110	N	
079	01001111	O	
080	01010000	P	
081	01010001	Q	
082	01010010	R	
083	01010011	S	
084	01010100	T	
085	01010101	U	
086	01010110	V	
087	01010111	W	
088	01011000	X	
089	01011001	Y	
090	01011010	Z	
091	01011011	[	Left bracket / Opening bracket
092	01011100	\	Back slash
093	01011101	]	Right bracket / Closing bracket
094	01011110	^	Caret / Circumflex
095	01011111	_	Underscore
096	01100000	`	Back quote
097	01100001	a	
098	01100010	b	
099	01100011	c	
100	01100100	d	
101	01100101	e	
102	01100110	f	
103	01100111	g	
104	01101000	h	
105	01101001	i	
106	01101010	j	
107	01101011	k	
108	01101100	l	
109	01101101	m	
110	01101110	n	
111	01101111	o	
112	01110000	p	
113	01110001	q	
114	01110010	r	
115	01110011	s	
116	01110100	t	
117	01110101	u	
118	01110110	v	
119	01110111	w	
120	01111000	x	
121	01111001	y	
122	01111010	z	
123	01111011	{	Left brace / Opening brace
124	01111100	|	Vertical bar
125	01111101	}	Right brace / Closing brace
126	01111110	~	Tilde
127	01111111	DEL	Delete

Code ASCII

(Wikipédia)

Connaissez-vous l'art ASCII ?