EPAI, Jérôme Frossard (2017)

CARACTÉRISER, METTRE EN FORME ET ANALYSER DES DONNÉES

Caractériser, mettre en forme et analyser des données

Historique

- Code de signaux maritimes
- Code morse pour le télégraphe dès 1836
  - initialement lettres et chiffres
  - quelques symboles ajoutés au fil du temps
- Code Baudot pour le Telex dès 1874
  - Premier code à utiliser un format binaire
  - 5 bits permettent de coder 32 caractères (2^5)

Principe du codage des caractères

- Pour représenter des caractères dans un ordinateur, on doit coder ces caractères sous forme de nombre.
- Pour cela, on procède en deux temps :
  1. On définit une table qui associe chaque caractère d'un alphabet à un numéro (point de code).
  2. On définit une façon de coder les numéros en binaire avec un nombre fixe ou variable de bits.
- Convertir les numéros en binaire et les représenter avec un nombre fixe de bits (p. ex. 8 bits) est une manière simple, mais ce n'est pas la seule.

Bit et byte

- Pour un processeur, un **mot** (*word*) est la taille d'un nombre que son unité de calcul peut traiter. Les premiers processeurs utilisaient souvent des mots de 36 bits.
- En raison de la capacité limitée de la mémoire, il n'était pas raisonnable de coder un seul caractère par mot.
- La solution a été de diviser chaque mot en six **morceaux** (*bite* [baIt]) de 6 bits, chacun pouvant contenir un caractère.
- Pour éviter la possible confusion entre « *bite* » et « *bit* » à l'écrit, *bite* a été changé en **byte** (même prononciation).
- Un byte est la plus petite subdivision d'un mot qui peut être traitée directement par un processeur.

Byte et octet

- Dès la fin des années 1960, par souci de standardisation des composants :
  - La mémoire est toujours organisée en cellules de 8 bits (**octet**).
  - La taille d'un mot pour un processeur est toujours un nombre entier d'octets (p. ex. 8 octets pour un processeur 64 bits).
  - La taille d'un byte est toujours un octet.
- Byte et octet sont devenus synonymes.
- On préfère octet en français et byte en anglais.

ASCII

- En 1968, l'ASCII (*American Standard Code for Information Interchange*) est proposé pour faciliter l'échange de données entre différentes machines (IBM, Bull, etc.).
- Le standard inclus les lettres majuscules et minuscules, les chiffres, des signes de ponctuation, des caractères spéciaux et des caractères de contrôles, 127 caractères au total.
- Le choix qui est fait de coder les caractère sur 7 bits permet de coder un caractère et un bit de parité pour la détection d'erreur de transmission dans un byte de 8 bits (octet).

Table de caractères ASCII

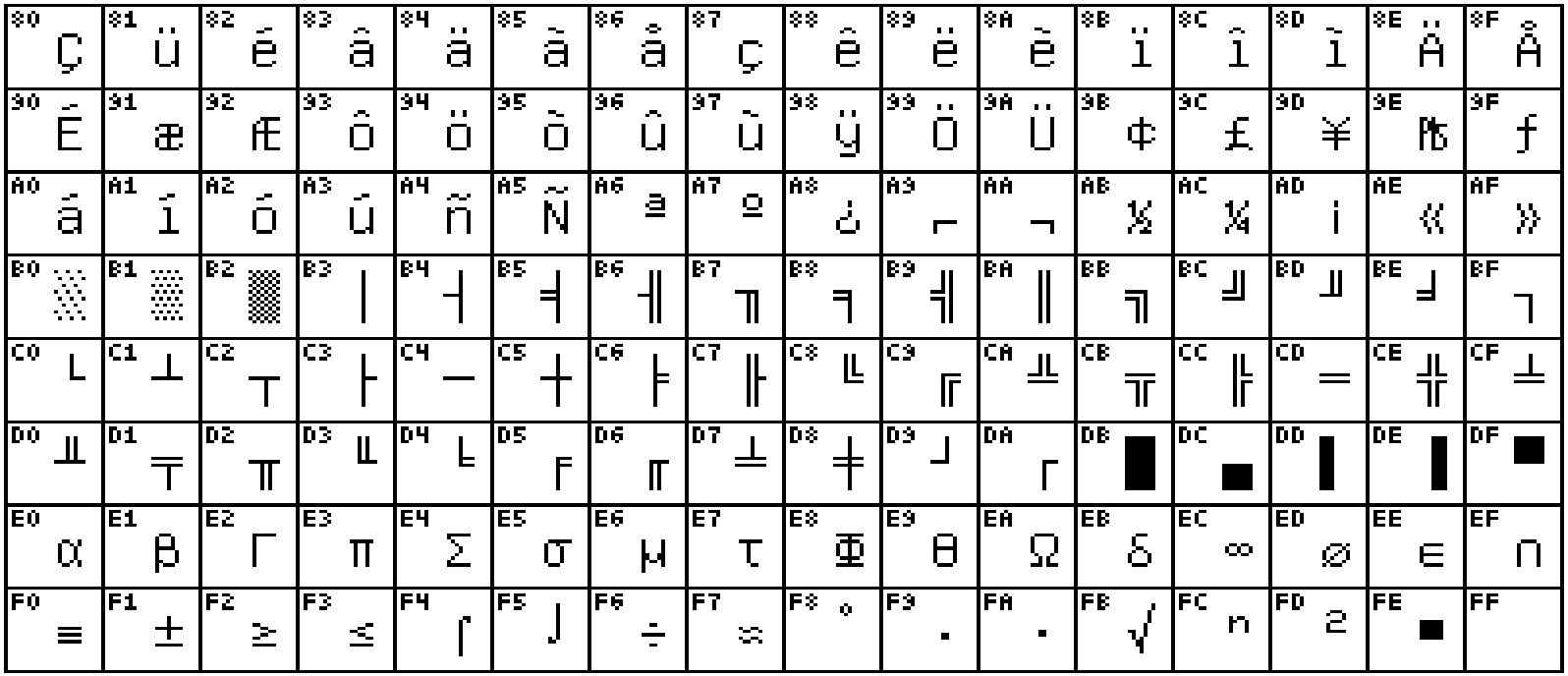
MSB \ LSB	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	SP	!	"	#	$	%	&	'	(	)	*	+	,	—	.	/
0011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	—
0110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	p	q	r	s	t	u	v	w	x	y	z	|	~	~	~	DEL

BINARY — HEX — ASCII

Et les accents ?

- Problème : L'ASCII (ISO/IEC 646) inclut l'alphabet latin de base, mais pas de lettres accentuées (é, ü, etc.)
- Solution : étendre l'ASCII à 8 bits et permettre de représenter 128 caractères supplémentaires.
- Problème : 128 caractères de plus ne suffisent pas pour représenter toutes les variations de l'alphabet latin (accent aigu, grave et circonflexe, tréma, caron, rond en chef, etc.).
- Solution : les 128 premiers caractères sont toujours les mêmes, les 128 derniers sont spécifiques à une langue ou à un groupe de langues (pages de codes).

Page de code 437 (DOS LatinUS)



Page de code 850 (DOS Latin1)

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Ç	ü	é	â	ä	ů	ć	ç	ł	ë	Ő	ő	î	Ź	Ä	Ć
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
É	Ĺ	ĺ	ô	ö	Ľ	ľ	Ś	ś	Ö	Ü	Ť	ť	Ł	×	č
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
á	í	ó	ú	Ą	ą	Ž	ž	Ę	ę	¬	ź	Č	ş	«	»
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
░	▒	▓	│	┤	Á	Â	Ě	Ş	╣	║	╗	╝	Ż	ż	┐
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
└	┴	┬	├	─	┼	Ă	ă	╚	╔	╩	╦	╠	═	╬	¤
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
đ	Đ	Ď	Ë	ď	Ň	Í	Î	ě	┘	┌	█	▄	Ţ	Ů	▀
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
Ó	ß	Ô	Ń	ń	ň	Š	š	Ŕ	Ú	ŕ	Ű	ý	Ý	ţ	´
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
	˝	˛	ˇ	˘	§	÷	¸	°	¨	˙	ű	ř	˚	■	

Page de code 855 (DOS Cyrilic)

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
ђ	Ђ	ѓ	Ѓ	ё	Ё	є	Є	ѕ	Ѕ	і	І	ї	Ї	ј	Ј
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
љ	Љ	њ	Њ	ћ	Ћ	ќ	Ќ	ў	Ў	џ	Џ	ю	Ю	ъ	Ъ
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
а	А	б	Б	ц	Ц	д	Д	е	Е	ф	Ф	г	Г	«	»
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
░	▒	▓	│	┤	х	Х	и	И	╣	║	╗	╝	й	Й	┐
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
└	┴	т	├	─	┼	к	К	╚	╔	╩	╦	╠	═	╬	¤
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
л	Л	м	М	н	Н	о	О	п	┘	Г	▉	▄	П	я	▀
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
Я	р	Р	с	С	т	Т	у	У	ж	Ж	в	В	ь	Ь	№
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
	ы	Ы	з	З	ш	Ш	э	Э	щ	Щ	ч	Ч	§	■	

Windows et l'ISO

- L'idée des pages de code a été reprise et standardisée (ISO 8859) dans la seconde moitié des années 80.
- Les 128 premiers caractères sont ceux de l'ASCII.
- Suppression des caractères graphiques dans la partie étendue au profit du plus grand nombre de caractères accentués.
- En 2017, Windows utilise toujours des pages de code et certaines sont basées sur le standard ISO :
  - ISO 8859-1 / Windows 1252 (latin 1)
  - ISO 8859-2 / Windows 1250 (latin 2)
  - ISO 8859-7 / Windows 1253 (greek)

ISO 8859-1

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x																
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																
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

positions inutilisées

positions inutilisées

Windows 1252

	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

Inconvénients des pages de code

- Il n'est pas possible d'avoir en même temps des caractères latins, cyrilliques et arabes par exemple.
- Les programmes sont généralement écrits pour une page de code spécifique. Si le système est configuré avec une autre page de code, le programme ne s'affiche pas correctement.
- Il n'est pas possible d'écrire un programme Java en mode texte qui fonctionne dans tous les cas.
  - L'output de NetBeans utilise les pages de code Windows.
  - La console de Windows utilise les pages de codes DOS.
  - Le terminal de Linux utilise Unicode.

Les alphabets non latins

- Le chinois simplifié comprend largement plus de 2000 caractères, le japonais un peu moins de 2000 (beaucoup sont communs aux deux langues)
- Nécessite plus de 8 bits : codes multibytes
- Exemple : Shift JIS
  - Les caractères ASCII (codes de 00_{16} – $7F_{16}$) et les Kana qui (codes $A0_{16}$ – DF_{16}), sont codés sur un seul octet.
  - Les caractères Kanji sont représenté par deux octets : lorsqu'un octet contient une valeur comprise entre 81_{16} et $9F_{16}$ ou $E0_{16}$ et FC_{16}, il doit être associé à l'octet suivant pour former le code du caractère.
- Problème : pas de consensus, versions incompatibles

Une table pour tous les caractères

- Les codes sur 8 bits avec des pages de codes et les codes multibytes permettent l'encodage de n'importe quel caractère de n'importe quel alphabet, mais l'échange de données et la représentation simultanée de plusieurs alphabets sont toujours des problèmes.
- Solution : créer un seul code qui comprend tous les caractères de tous les alphabets connus

Unicode

- L'Unicode est né de la collaboration d'un consortium privé et de l'ISO, l'Unicode / ISO 10646 définit dès 1991 un jeu de caractères universel.
- Unicode associe à chaque caractère un numéro appelé **point de code**.
- L'idée initiale est d'encoder les points de code sur 16bits, mais très vite, le nombre de caractères dépasse la limite de 65535 et une autre manière de représenter les points de code devient nécessaire.
- En 2017, près de 245'000 points de code sont assignés sur les 1'114'112 points de code disponibles.

Encoder les points de code

- Même si des mots de 16 bits ne suffisent plus, il n'est pas question, dans le courant des années 90, d'utiliser un mots de 32 bits pour chaque caractère (la mémoire est encore très coûteuse).
- Un solution est de coder les points de code avec un nombre de bits variable :
  - UTF-8 : encodage sur 1, 2, 3 ou 4 octets.
  - UTF-16 : encodage sur 1 ou 2 mots de 16 bits.
  - UTF-32 : encodage sur 1 mot de 32 bits.

UTF-8

- Unicode Transformation Format sur 8 bits
- UTF encode les points de code sur 1, 2, 3 ou 4 octets
- On utilise les bits de poids fort pour déterminer s'il s'agit ou non du premier octet d'un code, et les bits restants pour coder la valeur du point de code en binaire :
  - 0xxx'xxxx
  - 110x'xxxx 10xx'xxxx
  - 1110'xxxx 10xx'xxxx 10xx'xxxx
  - 1111'0xxx 10xx'xxxx 10xx'xxxx 10xx'xxxxx

Exemple

- Comment encoder le caractère « é » ?
  - À l'aide d'une table, on trouve le point de code : $E9_{16}$
  - $E9_{16}$ est plus grand que $7F_{16}$mais plus petit 800_{16}, on peut donc le coder sur deux octets (11 bits).
  - En UTF-8, pour écrire un code sur deux octets, on utilise le format suivant : 110x'xxxx 10xx'xxxx
  - On convertit $E9_{16}$ en binaire : $1110'1001_2$
  - En remplaçant les x par la valeur du point de code sur 11 bits, on obtient : 1100'0011 1010'1001

Pourquoi est-ce important ?

- L'encodage de texte est utilisé dans tous les aspects de l'informatique :
  - Encodage de textes brut (fichier de texte).
  - Encodage d'un programme (code source) avec langage de programmation (p. ex. Java).
  - Encodage de la structure d'un texte avec un langage de description (p. ex. Open XML).
  - Encodage de l'information à travers les réseaux de communication (p. ex. HTTP, SMTP, etc.).
- Dans tous ces domaines il n'est pas rare de devoir prendre une décision relative à l'encodage des caractères.