

Encodages



Karën Fort

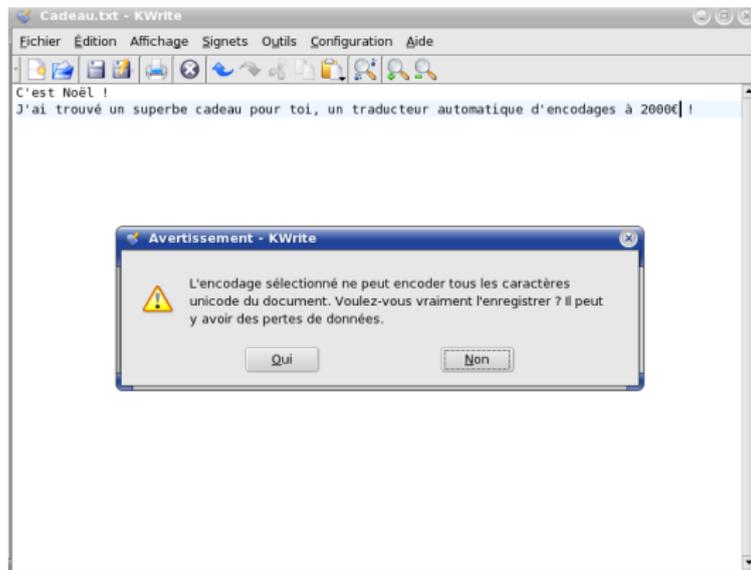
Karen.Fort@no-log.org

septembre 2012

Les encodages et vous

Les **encodages** existent,
vous les avez déjà rencontrés.

Encodages... à problèmes (1/3)



Impossible d'enregistrer...

Décomposons

1. **Création** d'un nouveau document texte vide dans un éditeur de texte.

Quelles difficultés peuvent se présenter à chaque étape ?

Décomposons

1. **Création** d'un nouveau document texte vide dans un éditeur de texte.
2. **Écriture** du texte.

Quelles difficultés peuvent se présenter à chaque étape ?

Décomposons

1. **Création** d'un nouveau document texte vide dans un éditeur de texte.
2. **Écriture** du texte.
3. **Enregistrement** du fichier.

Quelles difficultés peuvent se présenter à chaque étape ?

De mystérieux caractères

Un ou plusieurs caractère(s) que nous avons pu écrire sont apparemment [Unicode \(?\)](#) et ne sont pas enregistrables avec l'[encodage par défaut](#).

De mystérieux caractères

Un ou plusieurs caractère(s) que nous avons pu écrire sont apparemment [Unicode \(?\)](#) et ne sont pas enregistrables avec l'[encodage par défaut](#).

Solution ?



Encodages... à problèmes (2/3)

Contents of songs by Bulat Okudjava - Mozilla Firefox

http://bard-cafe.komkon.org/bae/okudjava.htm

Βουλγαρικά × ΙΑΙΕΙΟΑΑΑ, 1969 CIA

ΔΑΟΙΕ ΑΟΙΑΟΑ ΙΕΘΑΟΑ×Υ

Win Lat

	ΔΑΟΙΝ	Ι&ΑΙ xOAIN
ΙΑΕΘΟΙ× Ε ΘΡΟΝΙ		217k 1:48
Ι ΑΙΘΕΑ ΠΟΙΥΤΑ		215k 1:47
ΟΙΟΙΑΙΕΕ ΟΘΗΙΑΕΑΟ		325k 2:42
ΔΟΙΝ Ι ΙΟΟΑ×ΟΑ		321k 2:41
αΑΑΟ		328k 2:44
= Ο ΟΙΥΕΘΕΑ, ΧΟΙΕΙΠΟΟ ΟΑΔΙΣ...		255k 2:07
αΟΙΑΟΙΕ ΟΠΙΑΟΕΕ		139k 1:09
ΔΑΘΙΑΕ Ι ΑΟΘΑΕΕ		160k 1:20
+ ΔΙΕΙΑ ΙΑ ΡΟΘΑ ΟΟΑΔΙ ΟΙ&ΕΟΙΟΝ ΕΙΟΙΟ...		311k 2:36
ΙΟΥΕΡΑΙΟ ΙΑΕΦΟΥ×ΑΑΟ ×ΑΙΟ		331k 2:46
ΔΟΙΥΕ ΕΙΟ		164k 1:22
αΟΑΟΘΕΕΑ ΟΑΑΟΑ		269k 2:15
ΔΑΘΙΑΕ ΑΙΘΕΕΑΙΟΕΙΟΙ ΟΠΙΑΟΑ		184k 1:32
ΟΙΟΘΕΑ ΔΑΕΙΟΑ		276k 2:18
α xOΑ ΟΑΕΕ ΟΑΙΟ		328k 2:44
ΙΑΑΑΟΑΥ ΙΑΙΙΘΕΕΕ ΙΟ&ΟΟΘΕΕ		251k 2:05
ΙΑ ΙΑΟΙ		122k 1:01
ΠΠΕΟ×Α αΟΑΙΟΑ + ΕΠΙΑ		410k 3:25
ΙΑΑΟ		340k 2:50
αΓΙΑΥΕΑ		396k 3:18
ΟΑΟΟΥΕ ΧΟΘΑΕ		166k 1:23
ε ΡΑΙΟ ΙΑΙ ΑΥΘΟ ΙΑ ΟΥ?		291k 2:26
=ΟΑ ΠΡΘ ΕΦΕΑΙΕ ΔΑΘΕΕ...		170k 1:25
ΙΑΟΙ×ΟΘΕΕ ΟΙΑΣ		187k 1:33
ΟΑΟΟΥΕ ΔΕΛΟΑΕ		276k 2:18
- ΑΟΑΙΥΕ ΑΑΟΑΑΙΥΕΕ		244k 2:02
ΟΥ x ΡΑΙ xΠΠ×ΑΟΑ?		222k 1:51

Termine

Terminal | [Inbox for karen... | @mac@brussel.io... | file://homefortka... | cours_BST - Navig... | [un]dictionary_uf... | Contents of songs...

http://bard-cafe.komkon.org/bae/okudjava.htm

Une page mal fichue

Le code source de la page ne spécifie pas l'[encodage](#) du code, le navigateur utilise un [code par défaut](#) qui ne correspond pas.

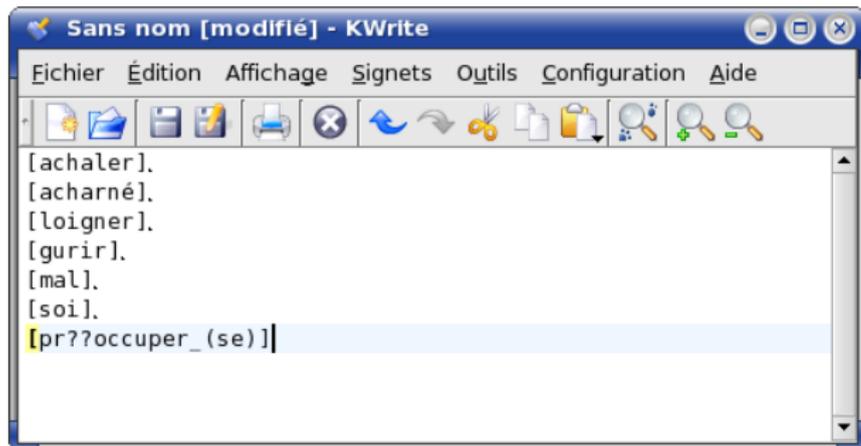
Une page mal fichue

Le code source de la page ne spécifie pas l'[encodage](#) du code, le navigateur utilise un [code par défaut](#) qui ne correspond pas.

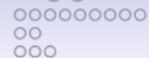
Solution pour le lecteur ?

Solution pour le créateur/traducteur de la page ?

Encodages... à problèmes (3/3)



Extrait (tronqué) d'un lexique du français... Imaginez, ça pourrait être un fichier client important, dans une langue que vous ne connaissez pas...



- Que voyez-vous ?
- Que s'est-il passé ?
- Que faire ?

Des problèmes différents, des solutions différentes

- **Enregistrer** dans le "bon" encodage (exemple 1).

Des problèmes différents, des solutions différentes

- **Enregistrer** dans le "bon" encodage (exemple 1).
- Fournir suffisamment d'**infos** au logiciel de visualisation pour que l'affichage de mon texte soit correct (exemple 2).

Des problèmes différents, des solutions différentes

- **Enregistrer** dans le "bon" encodage (exemple 1).
- Fournir suffisamment d'**infos** au logiciel de visualisation pour que l'affichage de mon texte soit correct (exemple 2).
- **Ne pas mélanger les encodages** (exemple 3).

Des problèmes différents, des solutions différentes

- **Enregistrer** dans le "bon" encodage (exemple 1).
- Fournir suffisamment d'**infos** au logiciel de visualisation pour que l'affichage de mon texte soit correct (exemple 2).
- **Ne pas mélanger les encodages** (exemple 3).
- Un point dont nous n'avons même pas parlé tellement il semble évident : s'assurer de la **capacité** du logiciel à gérer "proprement" la langue.

C'est toutes ces questions (ainsi que d'autres) que nous allons essayer d'approfondir dans ce cours.

La langue de l'ordinateur

- Le **texte** n'existe pas en informatique.

La langue de l'ordinateur

- Le **texte** n'existe pas en informatique.
- Quelle langue "parle" l'ordinateur ? Quels sont ses mots ?

La langue de l'ordinateur

- Le **texte** n'existe pas en informatique.
- Quelle langue "parle" l'ordinateur ? Quels sont ses mots ?
- Matériellement, un ordinateur ne comprend que le langage **binaire**, c'est-à-dire une suite de 0 et de 1.
Pour faire simple : du courant, pas de courant.

La langue de l'ordinateur

- Le **texte** n'existe pas en informatique.
- Quelle langue " parle " l'ordinateur ? Quels sont ses mots ?
- Matériellement, un ordinateur ne comprend que le langage **binaire**, c'est-à-dire une suite de 0 et de 1.
Pour faire simple : du courant, pas de courant.
- On appelle **bit** (BInary digiT) cette unité élémentaire d'information, qui peut prendre comme valeur 0 ou 1.

La langue de l'ordinateur

- Le **texte** n'existe pas en informatique.
- Quelle langue "parle" l'ordinateur ? Quels sont ses mots ?
- Matériellement, un ordinateur ne comprend que le langage **binaire**, c'est-à-dire une suite de 0 et de 1.
Pour faire simple : du courant, pas de courant.
- On appelle **bit** (Binary digiT) cette unité élémentaire d'information, qui peut prendre comme valeur 0 ou 1.
- Petit exercice : combien existe-t'il de possibilités de combiner 2 bits ?

La langue de l'ordinateur

- Le **texte** n'existe pas en informatique.
- Quelle langue "parle" l'ordinateur ? Quels sont ses mots ?
- Matériellement, un ordinateur ne comprend que le langage **binaire**, c'est-à-dire une suite de 0 et de 1.
Pour faire simple : du courant, pas de courant.
- On appelle **bit** (Binary digiT) cette unité élémentaire d'information, qui peut prendre comme valeur 0 ou 1.
- Petit exercice : combien existe-t'il de possibilités de combiner 2 bits ?
- 00 ou 11 ou 01 ou 10 = 4 possibilités.

Un peu plus loin dans les bases

- En fait, un **processeur** manipule plutôt des paquets de bits de taille fixe.

Un peu plus loin dans les bases

- En fait, un **processeur** manipule plutôt des paquets de bits de taille fixe.
- Les premiers processeurs fixèrent la taille de ces paquets à huit bits, soit un **octet**.

Un peu plus loin dans les bases

- En fait, un **processeur** manipule plutôt des paquets de bits de taille fixe.
- Les premiers processeurs fixèrent la taille de ces paquets à huit bits, soit un **octet**.

NOTE 1 octet = 8 bits = 1 **byte** (en anglais dans le texte).

Un peu plus loin dans les bases

- En fait, un **processeur** manipule plutôt des paquets de bits de taille fixe.
- Les premiers processeurs fixèrent la taille de ces paquets à huit bits, soit un **octet**.

NOTE 1 octet = 8 bits = 1 **byte** (en anglais dans le texte).

NOTE de huit bits (processeur 8086), les processeurs sont passés à 32 bits (Pentium 4), pour arriver aujourd'hui à 64. Ces paquets correspondent en fait à la capacité de transport (dans ses bus) et de traitement de la machine (taille des registres).

○○○○
○○○
○○○
○○○○○○●
○○○○
○○○○○○○○○
○○○○○
○○○○○○○○
○○○○○
○○○○○
○○○○○○○○○○○○○○
○○
○○
○○○○
○○○○○
○○

Dites "A" !

- "A" est en fait une **entité abstraite** dont le nom est "a majuscule" et dont le **glyphe** ressemble à un triangle dont on aurait raccourci et remonté le côté bas [dessin].



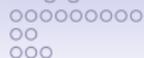
Dites "A" !

- "A" est en fait une **entité abstraite** dont le nom est "a majuscule" et dont le **glyphe** ressemble à un triangle dont on aurait raccourci et remonté le côté bas [dessin].
- Comment dit-on "A" à un ordinateur ?



Dites "A" !

- "A" est en fait une **entité abstraite** dont le nom est "a majuscule" et dont le **glyphe** ressemble à un triangle dont on aurait raccourci et remonté le côté bas [dessin].
- Comment dit-on "A" à un ordinateur ?
- 01000001



Dites "A" !

- "A" est en fait une **entité abstraite** dont le nom est "a majuscule" et dont le **glyphe** ressemble à un triangle dont on aurait raccourci et remonté le côté bas [dessin].
- Comment dit-on "A" à un ordinateur ?
- 01000001
- 1 caractère = 1 octet (attention, simplification).

Le byte et ses bits

- Combien de **valeurs** peut-on coder sur un **octet** ?
- 1 octet = 8 bits.
- $2 \times 2 = 2^8 = 256$

Quelques exemples

Caractère	Code Binaire	Code décimal	Description
A	01000001	065	Caractère "A"
a	01100001	097	Caractère "a"
T	01010100	084	Caractère "T"
t	01110100	116	Caractère "t"
3	00110011	051	Caractère "3"
\$	00100100	036	Caractère "\$"
BEL	00000111	007	Bip

ASTUCE : pour passer de la majuscule à la minuscule, il suffit de mettre le troisième bit à 1, futé, non ?

NOTE : les **caractères chiffres** ont leur propre code.

NOTE 2 : il n'existe qu'un "A", qui s'affiche différemment selon les **polices** ou le **style** (voir Définitions).

○○○○
○○○
○○○○○○
○○●○
○○○○○○○○○
○○○○○
○○○○○○○○
○○○○○
○○○○○○○○○○○○○○
○○
○○○○
○○○○○
○○

Exercice : dites "Bonjour" au PC !

- Combien de **bits** dans "Bonjour" ?

Exercice : dites "Bonjour" au PC!

- Combien de **bits** dans "Bonjour" ?
- "Bonjour" contient 7 **caractères**
 - 1 caractère = 1 octet
 - 1 octet = 8 bits
 - Donc...

Exercice : dites "Bonjour" au PC !

- Combien de **bits** dans "Bonjour" ?
- "Bonjour" contient 7 **caractères**
 - 1 caractère = 1 octet
 - 1 octet = 8 bits
 - Donc...
- 7 caractères x 8 bits = **56 bits** !

Exercice : dites "Bonjour" au PC !

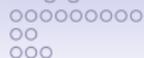
- Combien de **bits** dans "Bonjour" ?
- "Bonjour" contient 7 **caractères**
 - 1 caractère = 1 octet
 - 1 octet = 8 bits
 - Donc...
- 7 caractères x 8 bits = **56 bits** !
- Pour l'ordinateur : 11000010011100001001001110101100001101011100111101011011

Pas si simple

- Combien d'octets dans un kilo-octet (Ko en français, Kb en anglais) ?

Pas si simple

- Combien d'octets dans un kilo-octet (Ko en français, Kb en anglais) ?
- Pour les informaticiens : $1 \text{ Ko} = 2^{10} \text{ octets} = 1\,024 \text{ octets}$



Pas si simple

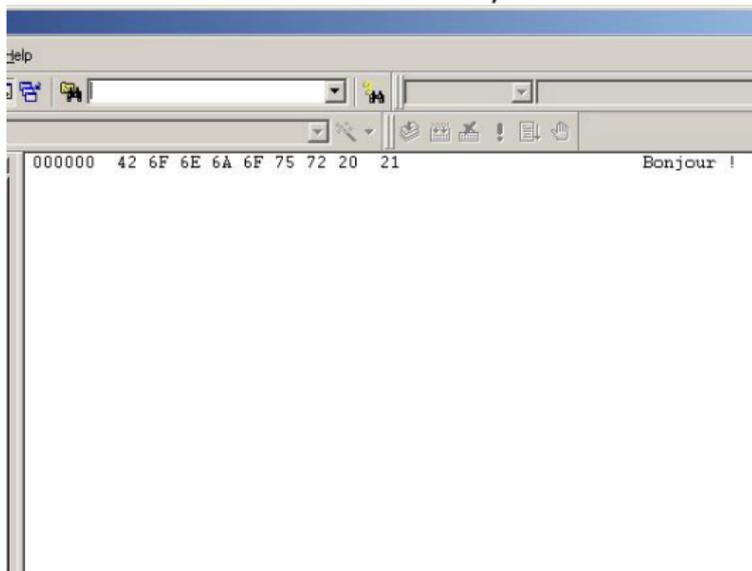
- Combien d'octets dans un kilo-octet (Ko en français, Kb en anglais) ?
- Pour les informaticiens : $1 \text{ Ko} = 2^{10} \text{ octets} = 1\,024 \text{ octets}$
- Pour les fabricants de matériel, depuis la normalisation de 1998, due à un procès : $1 \text{ Ko} = 10^3 \text{ octets} = 1\,000 \text{ octets}$

Pas si simple

- Combien d'octets dans un kilo-octet (Ko en français, Kb en anglais) ?
- Pour les informaticiens : $1 \text{ Ko} = 2^{10} \text{ octets} = 1\,024 \text{ octets}$
- Pour les fabricants de matériel, depuis la normalisation de 1998, due à un procès : $1 \text{ Ko} = 10^3 \text{ octets} = 1\,000 \text{ octets}$
- Pour plus d'infos sur ce sujet (on devrait maintenant parler de kibi et non de kilo pour les kilos binaires) :
<http://fr.wikipedia.org/wiki/Octet>

Sous le signe de l'hexa(décimal)

- Si vous ouvrez un fichier en **binaire**, vous ne verrez pas que des 0 et des 1. Vous verrez ça :



- Et ça, c'est de l'**hexa**. et c'est quand même plus facile à lire que du binaire.

Sous le signe de l'hexa(décimal)

- L'hexadécimal c'est de la **base 16**.

Sous le signe de l'hexa(décimal)

- L'hexadécimal c'est de la **base 16**.
- C'est pratique pour exprimer des octets, parce qu'avec un nombre de deux chiffres en base 16, on a 16 possibilités deux fois de suite, donc :
- $16 \times 16 = 16^2 = 256$

Sous le signe de l'hexa(décimal)

- L'hexadécimal c'est de la **base 16**.
- C'est pratique pour exprimer des octets, parce qu'avec un nombre de deux chiffres en base 16, on a 16 possibilités deux fois de suite, donc :
- $16 \times 16 = 16^2 = 256$
- Ce qui fait pile 2^8 . Ca vous dit quelque chose ?

Sous le signe de l'hexa(décimal)

- L'hexadécimal c'est de la **base 16**.
- C'est pratique pour exprimer des octets, parce qu'avec un nombre de deux chiffres en base 16, on a 16 possibilités deux fois de suite, donc :
- $16 \times 16 = 16^2 = 256$
- Ce qui fait pile 2^8 . Ca vous dit quelque chose ?
- Il a exactement autant de nombres binaires à 8 "chiffres" (des octets) que de nombres hexadécimaux à 2 "chiffres" !

Sous le signe de l'hexa(décimal)

- L'hexadécimal c'est de la [base 16](#).
- C'est pratique pour exprimer des octets, parce qu'avec un nombre de deux chiffres en base 16, on a 16 possibilités deux fois de suite, donc :
- $16 \times 16 = 16^2 = 256$
- Ce qui fait pile 2^8 . Ca vous dit quelque chose ?
- Il a exactement autant de nombres binaires à 8 "chiffres" (des octets) que de nombres hexadécimaux à 2 "chiffres" !
- Les informaticiens adorent ! Du coup, les éditeurs qui proposent d'ouvrir des fichiers en binaires les ouvrent en fait en hexa.

○○○○
○○○
○○○○○○
○○○○
○○●○○○○○○
○○○○○
○○○○○○○○
○○○○○○
○○○○○○○○○○○○○
○○
○○○○
○○○○○
○○

J'apprends à compter

Pour compter en **base 16** il faut **inventer** de nouveaux "chiffres" :

0 1 2 3 4 5 6 7 8 9 A B C D E F

Donc, l'ordinateur cause octet, so what ?

Il faut un [traducteur](#) pour que notre texte soit "codé" et "décodé" proprement, de manière **standardisée**.

C'est là qu'interviennent les tables de conversion, ou les [encodages](#).

Récapitulons

- un ordinateur ne comprend que le langage **binaire**, c'est-à-dire une suite de 0 et de 1.
- l'objet qui prend comme valeur 0 ou 1 est appelé **bit**.
- en simplifiant : 1 **caractère** = 1 **octet** = 8 bits.

Da ASCII code

- Au début de l'informatique, on estimait que coder les caractères sur **7 bits**, **ça suffisait bien**, puisque ça permet de représenter $2^7=128$ caractères différents. "A" est donc codé 1000001.
- C'est ainsi que fut créée la table **ASCII** (American Standard Code for Information Interchange), publiée en 1968.
- Très longtemps ce fut LA table de référence en informatique, à tel point qu'elle devint une norme : ISO-646.



Table ASCII : ça suffit, non ?

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

Non, ça ne suffit pas.

Big Blue invente l'ASCII étendu

- 1981, IBM sort son premier PC et **ajoute un bit à l'ASCII**.
- L'**ASCII étendu** OEM ((Original Equipment Manufacturer) permet donc de coder 256 caractères (2^8) et certains sont contents de pouvoir fêter Noël.
- Ce code ASCII étendu n'est **pas unique** et dépend fortement de la plateforme utilisée.

Table ASCII étendu OEM

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Ç	ü	é	â	ä	à	¸	ç	ê	ë	è	ï	î	ì	ñ	
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	ö	Ü	¢	£	¥	℞	ƒ
A	á	í	ó	ú	ñ	ñ			¿			½	¾	¡	«	»
B	▩	▩	▩													
C	⌞	⌞	⌞	⌞	-	⌞		⌞	⌞	⌞	⌞	⌞	⌞	⌞	=	⌞
D	μ	τ	π	μ	ε	ƒ	π		⌞	⌞	⌞	■	■	■	■	■
E	α	β	Γ	Π	Σ	σ	μ	τ	ϙ	θ	Ω	δ	∞	∞	€	∅
F	≡	±	≥	≤	ƒ	J	÷	∞	°	·	·	√	n	z		

On imagine la tête de ceux qui ont besoin d'écrire водка...

La famille ISO

... d'où l'idée de créer **différents jeux de caractères** au gré des besoins de chaque langue.

- Á partir de 1987 la table ASCII étendue fut déclinée en de multiples variations (toujours codées sur **un octet**).
- Les **128 premiers caractères** de tous les jeux ISO 8859 correspondent aux caractères **ASCII**.
- La norme ISO 8859 contient aujourd'hui **16 tables**, numérotées de 1 à 16 :1 pour les langues dites occidentales, 2 pour les langues d'Europe centrale/de l'Est, 5 pour le cyrillique, 6 pour l'arabe, 7 pour le grec, 8 pour l'hébreu, etc.

Exemple : ISO-8859-1

La table [ISO-8859-1](#) définit ce qu'elle appelle l'alphabet latin numéro 1 ou [latin-1](#) : 191 caractères de l'alphabet latin.

ISO/CEI 8859-1																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	inutilisés															
1x	inutilisés															
2x		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
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	inutilisés															
9x	inutilisés															
Ax		ı	ç	é	ı	ş	ı	§	ı	©	¸	«	¬		®	ı
Bx	°	±	²	³	´	µ	¶	·	,	ı	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



Exemple : ISO-8859-5

Remarquez les 128 premiers caractères...

ISO/IEC 8859-5																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	unused															
1x	unused															
2x	<u>SP</u>	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
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	unused															
9x	unused															
Ax	NBSP	Ё	Ђ	Ѓ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	SHY	Ў	Ѓ
Bx	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Cx	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Dx	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
Ex	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
Fx	Ѕ	ё	ђ	ѓ	є	ѕ	і	ї	ј	љ	њ	ћ	ќ	ѕ	ў	џ

Exemple : ISO-8859-15

Aussi connue sous le nom de Latin-9 (?), c'est une extension directe d'ISO 1 (mais plus tardive), à l'exception de 8 caractères.

Différences ISO 8859-1 / ISO 8859-15 :

Position	0xA4	0xA6	0xA8	0xB4	0xB8	0xBC	0xBD	0xBE
8859-1	¤	¦	¨	´	¸	¼	½	¾
8859-15	€	Š	š	Ž	ž	Œ	œ	Ÿ

○○○○
○○○
○○○
○○○○○○○
○○○○
○○○○○○○○○
○○○○●
○○○○○○○○
○○○○○○
○○○○○○○○○○○○○○
○○
○○○○
○○○○○
○○

Vous en avez assez ?

Evidemment, parmi les encodages les plus courants se trouve des encodages " maison " :

- Ceux de [Microsoft](#) : Windows1252 et al.
- Ceux d'[Apple](#) : MacRoman.

Limitations

- Problèmes d'**incomplétude** ou d'affichage pour certaines langues.
- Impossible d'écrire du russe **et** du français (hors ASCII) dans un seul et même fichier.
- Problèmes d'**erreurs** dues à la quasi-superposition de certains encodages (\$ se transformant en £, par exemple)
- Et le milliard de **Chinois** ?

Parenthèse sur le chinois

- Il existe deux "jeux" d'écriture du (des) chinois : le **simplifié**, utilisé dans la République Populaire de Chine et à Singapour et le **traditionnel**, plus répandu dans la diaspora, à Taïwan Hong Kong, Macao.

Parenthèse sur le chinois

- Il existe deux "jeux" d'écriture du (des) chinois : le **simplifié**, utilisé dans la République Populaire de Chine et à Singapour et le **traditionnel**, plus répandu dans la diaspora, à Taïwan Hong Kong, Macao.
- A chaque type d'écriture son encodage, en particulier : **GB 2312-80** (ou Guobiao) pour le chinois simplifié, avec 6763 caractères seulement (!!), et **Big5** pour le traditionnel, avec 13053 caractères.

Parenthèse sur le chinois

- Il existe deux "jeux" d'écriture du (des) chinois : le **simplifié**, utilisé dans la République Populaire de Chine et à Singapour et le **traditionnel**, plus répandu dans la diaspora, à Taïwan Hong Kong, Macao.
- A chaque type d'écriture son encodage, en particulier : **GB 2312-80** (ou Guobiao) pour le chinois simplifié, avec 6763 caractères seulement (!!), et **Big5** pour le traditionnel, avec 13053 caractères.
- **13053 caractères**? ! Mais ils les mettent où ?

Parenthèse sur le chinois

- Il existe deux "jeux" d'écriture du (des) chinois : le **simplifié**, utilisé dans la République Populaire de Chine et à Singapour et le **traditionnel**, plus répandu dans la diaspora, à Taïwan Hong Kong, Macao.
- A chaque type d'écriture son encodage, en particulier : **GB 2312-80** (ou Guobiao) pour le chinois simplifié, avec 6763 caractères seulement (!!), et **Big5** pour le traditionnel, avec 13053 caractères.
- **13053 caractères**? ! Mais ils les mettent où?
- Les Chinois ont tout simplement **plus de bits** pour coder leurs jeux de caractères : **16** exactement (soit 2 octets).

○○○○
○○○
○○○
○○○○○○○
○○○○
○○○○○○○○○
○○○○○
○○●○○○○○
○○○○○
○○○○○○○○○○○○○○
○○
○○○○
○○○○○
○○

Exercice : les éléphants d'Asie

- Combien de valeurs peut-on représenter sur 16 bits ?

○○○○
○○○
○○○
○○○○○○○
○○○○
○○○○○○○○○
○○○○○
○○●○○○○○
○○○○○
○○○○○○○○○○○○○○
○○
○○○○
○○○○○
○○

Exercice : les éléphants d'Asie

- Combien de valeurs peut-on représenter sur 16 bits ?
- $2 \times 2 = 2^{16} = 65536$

○○○○
○○○
○○○

○○○
○○○○
○○○○○

○○○○
○○○○○
○○●○

○○○○
○○○○○
○○○○○

○○○○○○○○○
○○
○○○

○
○

○○○○
○○

Exercice : les éléphants d'Asie

- Combien de valeurs peut-on représenter sur 16 bits ?
- $2 \times 2 = 2^{16} = 65536$
- Ca suffit donc pour le chinois, même traditionnel. D'ailleurs, même 14 bits auraient suffi... ($2^{14} = 16384$)
Alors pourquoi 16 ?

○○○○
○○○
○○○○○○
○○○○
○○○○○○○○○
○○○○○
○○●○○○○○
○○○○○
○○○○○○○○○○○○○○
○○
○○○○
○○○○○
○○

Exercice : les éléphants d'Asie

- Combien de valeurs peut-on représenter sur 16 bits ?
- $2 \times 2 = 2^{16} = 65536$
- Ca suffit donc pour le chinois, même traditionnel. D'ailleurs, même 14 bits auraient suffi... ($2^{14} = 16384$)
Alors pourquoi 16 ?
- Parce que c'est incomparablement plus pratique, étant donné que l'ordinateur gère des [octets](#)...

○○○○
○○○
○○○○○○
○○○○
○○○○○○○○○
○○○○○
○○○●○○○○
○○○○○
○○○○○○○○○○○○○○
○○
○○○○
○○○○○
○○

Récapitulons

○○○○
○○○
○○○
○○○○○○○
○○○○
○○○○○
○○○○○○○○○
○○○○○
○○○○○
○○○●○○○○
○○○○○○
○○○○○○
○○○○○○○○○○○○○○
○○
○○
○○○○
○○○○○
○○

Récapitulons

- 1968 - [ASCII](#) (7 bits) : c'est pas Noël !

○○○○
○○○
○○○
○○○○○○○
○○○○
○○○○○
○○○○○○○○○
○○○○○
○○○○○
○○○●○○○○
○○○○○
○○○○○
○○○○○○○○○○○○○○
○○
○○○○
○○○○○
○○

Récapitulons

- 1968 - [ASCII](#) (7 bits) : c'est pas Noël !
- 1981 - [ASCII étendu](#) (8 bits) : c'est Noël sans la водка.

○○○○
○○○
○○○
○○○○○○○
○○○○
○○○○○○○○○
○○○○○
○○○●○○○○
○○○○○
○○○○○○○○○○○○○○
○○
○○○○
○○○○○
○○

Récapitulons

- 1968 - [ASCII](#) (7 bits) : c'est pas Noël !
- 1981 - [ASCII étendu](#) (8 bits) : c'est Noël sans la водка.
- 1987 - Les [ISO](#) (8 bits) : Noël avec водка à part, mais toujours pas d'€.

Récapitulons

- 1968 - [ASCII](#) (7 bits) : c'est pas Noël !
- 1981 - [ASCII étendu](#) (8 bits) : c'est Noël sans la водка.
- 1987 - Les [ISO](#) (8 bits) : Noël avec водка à part, mais toujours pas d'€.
- 1997 - [ISO-8859-15](#) (8 bits) : mise à jour d'ISO-8859-1, on passe à l'€ !

Récapitulons

- 1968 - [ASCII](#) (7 bits) : c'est pas Noël !
- 1981 - [ASCII étendu](#) (8 bits) : c'est Noël sans la водка.
- 1987 - Les [ISO](#) (8 bits) : Noël avec водка à part, mais toujours pas d'€.
- 1997 - [ISO-8859-15](#) (8 bits) : mise à jour d'ISO-8859-1, on passe à l'€ !
- En parallèle, des [encodages "maison"](#) (8 bits) : mêmes défauts qu'ISO. Ne sont pas reconnus comme [normes](#).

Récapitulons

- 1968 - [ASCII](#) (7 bits) : c'est pas Noël !
- 1981 - [ASCII étendu](#) (8 bits) : c'est Noël sans la водка.
- 1987 - Les [ISO](#) (8 bits) : Noël avec водка à part, mais toujours pas d'€.
- 1997 - [ISO-8859-15](#) (8 bits) : mise à jour d'ISO-8859-1, on passe à l'€ !
- En parallèle, des [encodages "maison"](#) (8 bits) : mêmes défauts qu'ISO. Ne sont pas reconnus comme [normes](#).
- On en reste à 8 bits = [256 caractères possibles](#), or le chinois en compte beaucoup plus !

Le projet Unicode

En 1988 (?) est élaboré un projet un peu fou : recenser **tous** les caractères de **toutes** les langues écrites existantes ou ayant existé et mettre au point une **table de référence universelle** capable de coder tout ça.

L'entreprise colossale est aussitôt baptisée projet **Unicode**.

Unicode se veut :

- **universel** : toutes les langues doivent être couvertes (même les plus rares)
- **efficace** : simple à analyser
- **uniforme** : nombre fixe de bits
- **non-ambigu** : une valeur = un seul caractère (une fois codé, éternel !)

La couverture d'Unicode

- Les **255 premiers caractères** de la table Unicode sont ceux de la table **ISO-5589-1**.

La couverture d'Unicode

- Les **255 premiers caractères** de la table Unicode sont ceux de la table **ISO-5589-1**.
- Première étape : les **63 586 caractères** les plus utilisés sont réunis dans le **Plan Multilingue de Base (PMB ou BMP en anglais)**.

La couverture d'Unicode

- Les **255 premiers caractères** de la table Unicode sont ceux de la table **ISO-5589-1**.
- Première étape : les 63 586 caractères les plus utilisés sont réunis dans le **Plan Multilingue de Base (PMB ou BMP en anglais)**.
- Première publication de la norme Unicode, en **1991** : 65 536 caractères sont recensés et encodés.

La couverture d'Unicode

- Les **255 premiers caractères** de la table Unicode sont ceux de la table **ISO-5589-1**.
- Première étape : les 63 586 caractères les plus utilisés sont réunis dans le **Plan Multilingue de Base (PMB ou BMP en anglais)**.
- Première publication de la norme Unicode, en **1991** : 65 536 caractères sont recensés et encodés.
- Aujourd'hui, **Unicode version 6.1.0** : plus de **110 116** lettres ou syllabes, chiffres ou nombres, symboles divers, signes diacritiques et signes de ponctuation.

La couverture d'Unicode

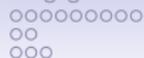
- Les **255 premiers caractères** de la table Unicode sont ceux de la table **ISO-5589-1**.
- Première étape : les 63 586 caractères les plus utilisés sont réunis dans le **Plan Multilingue de Base (PMB ou BMP en anglais)**.
- Première publication de la norme Unicode, en **1991** : 65 536 caractères sont recensés et encodés.
- Aujourd'hui, **Unicode version 6.1.0** : plus de **110 116** lettres ou syllabes, chiffres ou nombres, symboles divers, signes diacritiques et signes de ponctuation.
- Mais aussi : plus de 137 468 caractères à usage privé, 8 258 points de codes réservés de façon permanente et plusieurs centaines de caractères de contrôle ou modificateurs.

La couverture d'Unicode

- Les **255 premiers caractères** de la table Unicode sont ceux de la table **ISO-5589-1**.
- Première étape : les 63 586 caractères les plus utilisés sont réunis dans le **Plan Multilingue de Base (PMB ou BMP en anglais)**.
- Première publication de la norme Unicode, en **1991** : 65 536 caractères sont recensés et encodés.
- Aujourd'hui, **Unicode version 6.1.0** : plus de **110 116** lettres ou syllabes, chiffres ou nombres, symboles divers, signes diacritiques et signes de ponctuation.
- Mais aussi : plus de 137 468 caractères à usage privé, 8 258 points de codes réservés de façon permanente et plusieurs centaines de caractères de contrôle ou modificateurs.
- Unicode est donc encore très "vide".

Le point de code Unicode

- En **Unicode**, une lettre correspond à quelque chose appelé un **point de code** qui n'est qu'un **concept théorique**. Comment ce point de code est représenté en mémoire ou sur disque est une tout autre histoire...
- Toute les lettres de tous les alphabets se sont vues attribuer un **point de code** par le consortium Unicode. Ce point de code s'écrit : **U+XXXX**. Le U+ signifie "Unicode", et les X derrière sont en hexadécimal. U+FEC9 est la lettre arabe Ain. La lettre **A** correspond à **U+0041**.
- Exemple : Bonjour
U+0042 U+006F U+006E U+006A U+006F U+0075 U+0072
- Outils : charmap sous Windows 2000/XP, KCharSelect sous Linux.



C'est bien beau tout ça, mais l'ordinateur il en fait quoi ?

- Le consortium Unicode a prévu trois principaux formats "transformés" pour l'encodage des point de code en binaire. Ces **formats de transformation** sont baptisés **UTF(Unicode Transformation Format)**.
- UTF-32, UTF-16, UTF-8.
- Le principe intangible derrière ces transformations étant que tout point de code Unicode doit pouvoir être retrouvé **sans ambiguïté** à partir de sa version transformée.

Le gros simplet : UTF-32

- En **UTF-32** (ou UCS-4), tout point de code Unicode est directement codé en sa valeur binaire, sur un **nombre fixe de bits** (32) l'encodage revenant alors à une **simple conversion vers le binaire**.
- En contrepartie, ce format se révèle également le plus **gourmand** en ressources puisque chaque caractère, qu'il soit un "e" universel ou un idéogramme sub-indonésien, nécessitera **quatre octets** pour être codé.
- UTF-32 est donc assez **peu utilisé**.

L'indécis : UTF-16

- UTF-16 décompose les caractères Unicode en **seizets** binaires, soit **2 octets**.
- Ces 2 octets suffisent largement pour coder les caractères du PMB (partie correspondant à UCS-2).
- On utilise 2 seizets pour coder les autres. Ces seizets sont appelés **seizets d'indirection** (**surrogate pair** en anglais).
- UTF-16 est donc un encodage de **longueur variable**, en **16 ou 32 bits**.

L'économe : UTF-8

- Le format **UTF-8** est, encore plus qu'UTF-16, un procédé d'encodage à "taille variable".

L'économe : UTF-8

- Le format **UTF-8** est, encore plus qu'UTF-16, un procédé d'encodage à "taille variable".
- En UTF-8, chaque point de code de 0 à 127 (ASCII) est stocké **sur un seul octet**.

L'économe : UTF-8

- Le format **UTF-8** est, encore plus qu'UTF-16, un procédé d'encodage à "taille variable".
- En UTF-8, chaque point de code de 0 à 127 (ASCII) est stocké **sur un seul octet**.
- Les points de code à partir de 128 et au delà sont stockés en utilisant 2, 3, et **jusqu'à 4 octets** (6 en principe)!

Conclusion : machin-byte

- UTF-32 code toujours sur 4 octets.
- UTF-16 est double-byte, puisqu'il utilise 2 octets (sauf pour les caractères qui nécessite des "surrogate pairs").
- UTF-8 est lui multibyte, car il utilise entre 1 et 4 octets pour coder un caractère.

Résumé

- UTF-32 : □ □ □ □

Résumé

- UTF-32 : □ □ □ □
- UTF-16 : □ □ + □ □

Résumé

- UTF-32 : □ □ □ □
- UTF-16 : □ □ + □ □
- UTF-8 : □ + □ + □ + □

○○○○
○○○
○○○○○○
○○○○
○○○○○○○○○
○○○○○
○○○○○○○○
○○○○○●
○○○○○○○○○○○○○○
○○
○○○
○○○○○
○○

Exercice : qui c'est le meilleur ?

- Avantages et inconvénients de chaque UTF ?
- **UTF-32** est de longueur fixe, dont facile à traiter, mais il prend de la place.
- **UTF-16** est parfois plus délicat à manipuler, mais il prend moins de place et permet un bonne efficacité, même sur les caractères plus rares pour nous. C'est l'encodage du langage de programmation Java.
- **UTF-8** est le plus optimal pour un usage occidental, mais il prend beaucoup de place pour coder certains caractères. Il est en outre délicat à manipuler.

Parenthèse sur les œufs durs

- Pour transmettre une séquence d'octets dans un flux de données, par exemple la lettre "A" (U+0041), on a le choix : on peut commencer par l'**octet de poids faible** puis l'**octet de poids fort** ([00] puis [41]) ou le contraire ([41] puis [00]), tout dépend du processeur.

Parenthèse sur les œufs durs

- Pour transmettre une séquence d'octets dans un flux de données, par exemple la lettre "A" (U+0041), on a le choix : on peut commencer par l'**octet de poids faible** puis l'**octet de poids fort** ([00] puis [41]) ou le contraire ([41] puis [00]), tout dépend du processeur.
- Par exemple, les SPARC (Solaris) sont en **big-endian** alors que les Intel (PC) sont en **little-endian**.

Parenthèse sur les œufs durs

- Pour transmettre une séquence d'octets dans un flux de données, par exemple la lettre "A" (U+0041), on a le choix : on peut commencer par l'**octet de poids faible** puis l'**octet de poids fort** ([00] puis [41]) ou le contraire ([41] puis [00]), tout dépend du processeur.
- Par exemple, les SPARC (Solaris) sont en **big-endian** alors que les Intel (PC) sont en **little-endian**.
- Comment gérer ça en Unicode ?

Les petits et les grands Boutistes

- Solution BE (pour **Big Endian**) : les octets sont transmis dans un **ordre décroissant** de poids (le point de code U+0041 est donc décomposé en la séquence [00-41]),

Les petits et les grands Boutistes

- Solution BE (pour [Big Endian](#)) : les octets sont transmis dans un [ordre décroissant](#) de poids (le point de code U+0041 est donc décomposé en la séquence [00-41]),
- solution LE (pour [Little Endian](#)) : les octets sont transmis dans un [ordre croissant](#) de poids (le point de code U+0041 est alors décomposé en la séquence [41-00]),

Les indéterminés

- solution **indéterminée** : on peut choisir entre l'une ou l'autre méthode à condition d'indiquer en tout début de transmission la solution retenue, grâce à un point de code particulier joyeusement appelé **BOM (Byte Order Mark)** et codé U+FEFF (big-endian) ou FFFE (little-endian).

Les indéterminés

- solution **indéterminée** : on peut choisir entre l'une ou l'autre méthode à condition d'indiquer en tout début de transmission la solution retenue, grâce à un point de code particulier joyeusement appelé **BOM (Byte Order Mark)** et codé U+FEFF (big-endian) ou FFFE (little-endian).
- NOTE : la **BOM** n'est utile qu'en **UTF-16** et **UTF-32**, mais elle est souvent ajoutée pour marquer l'UTF-8 (mais n'est pas obligatoire). Elle est alors encodée EF BB BF.

En quoi ça nous concerne ?

- Avez-vous déjà rencontré un `ï»¿` en début de fichier, dans Notepad, par exemple ?

En quoi ça nous concerne ?

- Avez-vous déjà rencontré un `ï»¿` en début de fichier, dans Notepad, par exemple ?
- C'est la représentation en ISO 1 d'une **BOM** et ça prouve que votre fichier est en UTF-8.

En quoi ça nous concerne ?

- Avez-vous déjà rencontré un `ï»¿` en début de fichier, dans Notepad, par exemple ?
- C'est la représentation en ISO 1 d'une **BOM** et ça prouve que votre fichier est en UTF-8.
- Ça prouve aussi que votre éditeur est une **truffe** !

En quoi ça nous concerne ?

- Avez-vous déjà rencontré un `ï»¿` en début de fichier, dans Notepad, par exemple ?
- C'est la représentation en ISO 1 d'une **BOM** et ça prouve que votre fichier est en UTF-8.
- Ça prouve aussi que votre éditeur est une **truffe** !
- La BOM peut aussi poser des problèmes à certains programmes.

Récapitulons

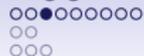
- **Unicode** : norme permettant de coder tous les caractères de toutes les langues du monde de manière non ambiguë.
- Distinction **points de code** et **formats de transformation**
- Les **UTF** (formats de transformation) : UTF-32 (4 octets), UTF-16 (2+2 octets), UTF-8 (de 1 à 4 octets)
- La **BOM** (Byte Order Mark) : sert d'indicateur de sens pour la lecture des octets, peut apparaître au début de certains fichiers utf-8 sous la forme **ï»¿**

HTML et les encodages

- Que savez-vous d'**HTML** ?
- Que savez-vous de la **gestion des encodages** par HTML ?

Survol (très) rapide de HTML

- HTML = **HyperText Markup Language**
- soit : un langage informatique de **balisage** (comme XML ou SGML)
- ... **textuel, structuré, non extensible**
- son “langage” ne peut pas être redéfini
- **syntaxe** assez souple : une balise ouverte n'a pas forcément à être fermée (sauf **XHTML**)



Exemple

Karën FORT

Doctorante - Equipe Représentation des Connaissances et Langage Naturel ([RCLN](#))



Sujet de thèse

- Les ressources annotées, un enjeu pour l'analyse de contenu : vers une méthodologie de l'annotation manuelle de corpus (dir. [Adeline Nazarenko](#))

Intérêts

Je m'intéresse aux ressources linguistiques (lexiques ou corpus), et plus particulièrement aux méthodologies qui président à leur construction.

Actualités

- Ethique et Big Data : création d'une charte auto-administrée permettant la traçabilité des ressources langagières ([wiki des travaux en cours](#)).
- Collaboration avec [G. Pesterev](#) (IRCAM) sur la définition d'une méthodologie pour l'annotation d'audio musical.
- Collaboration avec [M. Lafourcade](#) (LIRMM), [J. Chamberlain](#), U. Kruschwitz, M. Poesio (University of Essex) sur la création

Les dessous

Fichier source

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1
/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"><head>

  <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type" />

  <meta name="keywords" content="karen fort, Kar6eu1;n FORT, Karen FORT, NLP, TAL, annotation
manuelle, corpus linguistics, manual annotation, traitement automatique des langues, ressources
linguistiques, INIST, LIPN" /><title>Kar6eu1;n FORT</title>

|

  <link rel="stylesheet" href="._perso.css" type="text/css" /></head><body>
<div class="gauche"><a href="http://www.lipn.univ-paris13.fr/"></a>
<ul class="menu">
<li> <a href="._index.php">Accueil</a> </li>
<li> <a href="._Publications.php">Publications</a></li>
<li> <a href="._Projets.php">Projets</a></li>
<li> <a href="._Enseignement.php">Enseignements</a> </li>
<li> <a href="._ReferencesAnnot.php">R6eacute;f6eacute;rences</a> </li>
<li> <a href="._Liens.php">Liens</a> </li>
<li> <a href="._CV.php">CV</a> </li>
<br />

```

Les entités nommées

- En **HTML**, il existe un jeu standard de 252 **entités nommées** qui sont soit absentes de certains encodages de caractères, soit sensibles au balisage dans certains contextes (par exemple les signes inférieur et guillemets droits).
- Ces entités nommées peuvent être incluses dans un document HTML via l'utilisation de références, qui prennent la forme **&EntitéNommée;**, où EntitéNommée est le nom de l'entité.
- Exemple : "é" correspond à l'entité nommée é

Les entités numériques

- En **HTML**, il est possible d'écrire n'importe quel caractère **Unicode** à l'aide d'**entités numériques** référant le **code point** du caractère.
- Ces entités numériques prennent deux formes :
&#CodePoint;, où CodePoint est en **décimal**, et
**odePoint;**, où CodePoint est en **hexadécimal**.
- Exemple : "é" peut être encodé **é** ou **é**
- NOTE : le support de l'hexa est plus récent et peut être problématique pour les navigateurs les plus anciens.

○○○○
○○○
○○○○○○○
○○○○
○○○○○○○○○
○○○○○
○○○○○○○○
○○○○○
○○○○○○○○○○○●○○
○○
○○○○
○○○○○
○○

Exemple

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1
/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"><head>

  <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type" />

  <meta name="keywords" content="karen fort, Kar&euml;n FORT, Karen FORT, NLP, TAL, annotation
manuelle, corpus linguistics, manual annotation, traitement automatique des langues, ressources
linguistiques, INIST, LIPN" /><title>Kar&euml;n FORT</title>

|

  <link rel="stylesheet" href="/perso.css" type="text/css" /></head><body>
<div class="gauche"><a href="http://www.lipn.univ-parisl3.fr/"></a>
<ul class="menu">

  <li> <a href="/index.php">Accueil</a> </li>
  <li> <a href="/Publications.php">Publications</a></li>
  <li> <a href="/Projets.php">Projets</a></li>
  <li> <a href="/Enseignement.php">Enseignements</a> </li>
  <li> <a href="/ReferencesAnnot.php">R&eacute;f&eacute;rences</a> </li>
  <li> <a href="/Liens.php">Liens</a> </li>
  <li> <a href="/CV.php">CV</a> </li>
<br />

```

Avantages et inconvénients des entités

- Les **entités** sont lisibles par **n'importe quel navigateur**, en particulier les entités nommées.
- Les **entités** sont **indépendantes de l'encodage** du fichier source (pas de problème à l'enregistrement ou à la lecture).
- Par contre, le **code source** n'est pas facile à écrire et à éditer et il prend un peu plus de place.

Le tag meta et son charset

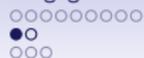
- Comment **spécifier un encodage** dans du **HTML** ? Comment **lire** le fichier HTML tant qu'on ne sait pas quel encodage est à l'intérieur ?

Le tag meta et son charset

- Comment **spécifier un encodage** dans du **HTML** ? Comment **lire** le fichier HTML tant qu'on ne sait pas quel encodage est à l'intérieur ?
- Il faut spécifier dans le **header** du code HTML une balise (tag) **meta** contenant des informations sur le fichier source.

Le tag meta et son charset

- Comment **spécifier un encodage** dans du **HTML** ? Comment **lire** le fichier HTML tant qu'on ne sait pas quel encodage est à l'intérieur ?
- Il faut spécifier dans le **header** du code HTML une balise (tag) **meta** contenant des informations sur le fichier source.
- Ce meta tag doit vraiment être la **première chose** dans le header parce que dès que le navigateur voit ce tag, il arrête de parser la page et recommence son interprétation depuis le début en utilisant l'encodage que vous avez spécifié.



Le tag meta et son charset

- Comment **spécifier un encodage** dans du **HTML** ? Comment **lire** le fichier HTML tant qu'on ne sait pas quel encodage est à l'intérieur ?
- Il faut spécifier dans le **header** du code HTML une balise (tag) **meta** contenant des informations sur le fichier source.
- Ce meta tag doit vraiment être la **première chose** dans le header parce que dès que le navigateur voit ce tag, il arrête de parser la page et recommence son interprétation depuis le début en utilisant l'encodage que vous avez spécifié.
- Exemple :

```
<meta http-equiv="Content-Type" content="text/html ; charset=utf-8" >
```

HTML : surprise, surprise !

Vérifiez toujours de quoi ça a l'air :

- sur plusieurs [navigateurs](#)
- sur plusieurs [systèmes d'exploitation](#) (non, le monde entier n'est pas sous Windows)

Survol (très) rapide de XML

- XML = **Extensible Markup Language**
- soit : un langage informatique de **balisage** (comme HTML ou SGML)
- ... **textuel**, **structuré**, et **extensible** car
- son “langage” (vocabulaire et grammaire) peut être redéfini (par exemple, *mabalise* peut être un nom de balise)
- **syntaxe** stricte, peut être validé par des outils automatiques.

XML : déclaration d'encodage

- Comme en HTML, on peut utiliser les [entités](#).
- Mais la [déclaration d'encodage](#) est obligatoire et intervient dès la première ligne du fichier :
- `<?xml version="1.0" encoding="utf-8" ?>`

○○○○
○○○
○○○○○○○
○○○○
○○○○○○○○○
○○○○○
○○○○○○○○
○○○○○
○○○○○○○○○○○○○○
○○
○○●○
○○○○○
○○

Récapitulons

Récapitulons

- **HTML** : langage non extensible

Récapitulons

- **HTML** : langage **non extensible**
- Encodages en **HTML** : utilisation d'**entités** (nommées ou numériques) ou de la **balise meta** pour déclarer l'encodage du document.

Récapitulons

- **HTML** : langage **non extensible**
- Encodages en **HTML** : utilisation d'**entités** (nommées ou numériques) ou de la **balise meta** pour déclarer l'encodage du document.
- **XML** : langage **extensible**

Récapitulons

- **HTML** : langage **non extensible**
- Encodages en **HTML** : utilisation d'**entités** (nommées ou numériques) ou de la **balise meta** pour déclarer l'encodage du document.
- **XML** : langage **extensible**
- Encodages en **XML** : utilisation d'**entités** (nommées ou numériques) possible mais **déclaration d'encodage** obligatoire.

Autres problèmes liés au multilinguisme

- **Right-to-left** (arabe, hébreu) : ou comment insérer un système d'écriture de droite à gauche dans un système d'écriture de gauche à droite (Texte bi-directionnel) ? Facile, des marques de contrôle sont prévues afin de pouvoir changer le sens d'écriture (en HTML, attribut DIR=" rtl" ou " ltr").
- **Agglutinations** (arabe).

Les retours à la ligne, ne quittez pas !

- Selon les plate-formes, le caractère représentant le **saut de ligne** n'est pas le même.
- **Windows** : CR+LF (Carriage Return Line Feed, héritage de la machine à écrire!)
- **Monde Unix** : LF
- **Mac** : CR

Les outils qui sauvent

- Bon **éditeur de texte** brut : EmEditor (Windows), Yudit (Unix). Pour Mac ?

Les outils qui sauvent

- Bon **éditeur de texte** brut : EmEditor (Windows), Yudit (Unix). Pour Mac ?
- Editeur qui permette l'ouverture en **binaire** : Visual, Eclipse (outils de dev.)

○○○○
○○○
○○○○○○
○○○○
○○○○○○○○○
○○○○○
○○○○○○○○
○○○○○
○○○○○○○○○○○○○○
○○
○○○○
○○●○○
○○

Définitions (Wikipedia) 1/3

- **glyphe** : représentation graphique (parmi une infinité possible) d'un signe typographique.
- **bit** : unité de mesure désignant la quantité élémentaire d'information représentée par un chiffre du système binaire.
- **octet** : une unité de mesure en informatique mesurant la quantité de données. Un octet est lui-même composé de 8 bits, soit 8 chiffres binaires. Le byte, soit la plus petite unité adressable d'un ordinateur, a presque toujours une taille d'un octet et les deux mots sont généralement, mais abusivement, considérés comme synonymes.

Définitions 2/3

- **charset / jeu de caractères** : ensemble de caractères.
Exemple : ISO Latin-1, Unicode.
- **encodage** : façon dont les caractères, dans un alphabet donné, sont convertis en octet. L'encodage n'indique absolument pas comment ces caractères seront affichés à l'écran ou après impression.
Exemple : iso-8859-1, UTF-8.

Définitions (Wikipedia) 3/3

- **police** et **fonte** : une fonte de caractères, en typographie, est un ensemble de glyphes, c'est-à-dire de représentations visuelles de caractères, d'une même famille, de même style, corps et graisse. Elle se distingue de la police d'écriture qui regroupe tous les corps et graisses d'une même famille, dont le style est coordonné.

Je leur ai tout piqué !

- Pour les bases : <http://www.arcanapercipio.com/>
- Pour les tables : <http://www.table-ascii.com/> ou <http://alis.isoc.org/index.html>
- Pour les définitions et tout le reste : <http://fr.wikipedia.org/>
- Pour Unicode, indispensable et drôle : <http://french.joelonsoftware.com/Articles/Unicode.html>
- Pour la bidirectionnalité : <http://www.i18nguy.com/markup/right-to-left.html>
- Mais je n'ai compris tout ça que grâce à [Bruno Freudensprung](#).

Copyright et al

- Ce cours a été réalisé en [LaTeX Beamer](#).
- Il est disponible sous licence [Creative Commons](#).