



# Encodages et multilinguisme

Karën Fort

karen.fort@sorbonne-universite.fr / <https://members.loria.fr/KFort/>



# Quelques sources d'inspiration

- ▶ Cours de Gilles Serasset (LIG) pour la partie multilingue du début
- ▶ Pour les tables : <http://www.table-ascii.com/> ou <http://alis.isoc.org/index.html>
- ▶ Pour les définitions : <http://fr.wikipedia.org/>
- ▶ Pour la bidirectionalité : <http://www.i18nguy.com/markup/right-to-left.html>
- ▶ Pour une autre présentation du même sujet : <http://pagesperso-systeme.lip6.fr/Jean-Francois.Perrot/inalco/cours13/>

## Les langues des humains

Quelques chiffres

Systèmes d'écriture

La « langue » de l'ordinateur

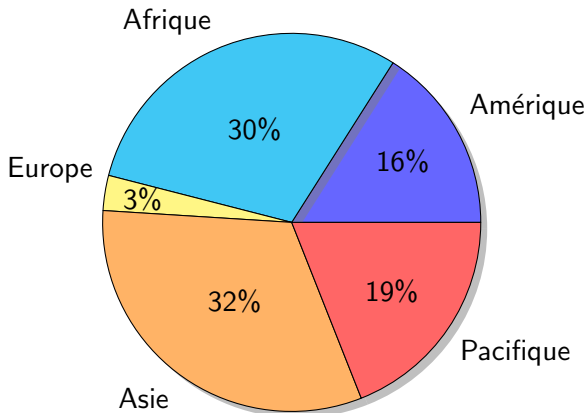
Les langues et l'informatique

Conclusion

Pour finir

# Langues dans le monde

L'édition 2013 de la base Ethnologue<sup>1</sup> recense plus de 7 000 langues vivantes dans le monde, réparties comme suit :

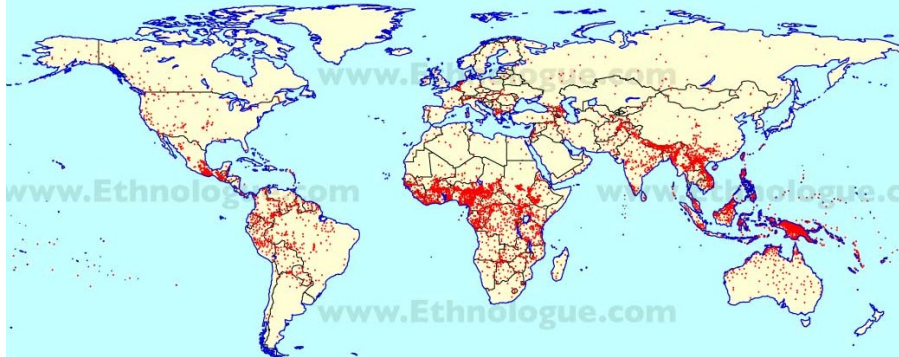
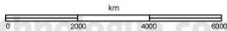


---

1. <http://www.ethnologue.com/>

## LANGUAGES OF THE WORLD

EACH DOT REPRESENTS THE PRIMARY LOCATION  
OF A LIVING LANGUAGE LISTED IN THE ETHNOLOGUE.



# Langues dans le monde (suite)

Mais

- ▶ hétérogénéité des sources
- ▶ délicate distinction entre langue et dialecte<sup>2</sup> (voir [Claude Hagège](#))

Suivant les sources, les estimations varient de 3 000 à 7 000 langues vivantes environ

**Max Weinreich**

"אַ שפּראַך איז אַ דיאַלעקט מיט אַן אַרמיי און פֿלאָט"

(a shprakh iz a dialekt mit an armey un flot)

une langue, c'est un dialecte avec une armée et une flotte

---

2. seules les langues font généralement l'objet de statistique, mais la définition d'une langue s'appuie sur des critères variables et subjectifs.

# Langues écrites, langues orales

Selon les sources, de 67 % à 90 % des langues sont orales. Cela donnerait en moyenne 1 500 langues écrites sur 6 912.

L'Alliance Biblique Universelle<sup>3</sup> indique quant à elle que la bible a été traduite en 2 426 langues différentes (au moins un livre de la bible a été traduit).

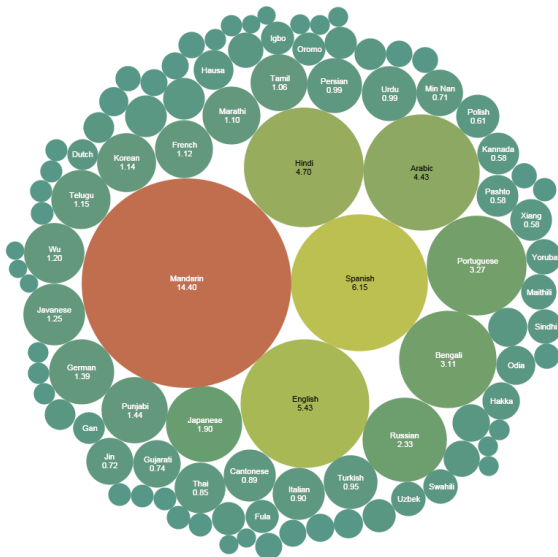
Certaines langues, considérées comme orale peuvent s'écrire en utilisant une transcription utilisant un alphabet existant.

---

3. [www.biblesociety.org](http://www.biblesociety.org)

# 100 langues les plus parlées en % de la population mondiale

*Nationalencyklopedin 2007 (2010) - CC-BY-SA-4.0 (Jroehl)*





# Systèmes d'écriture

L'écriture est apparue sous des formes différentes dans différentes civilisations humaines : Égypte, Mésopotamie, Chine, Inde et, plus tardivement, Mésoamérique.

La plus ancienne serait la Mésopotamie, vers 3 500-3 400 av. J.C.

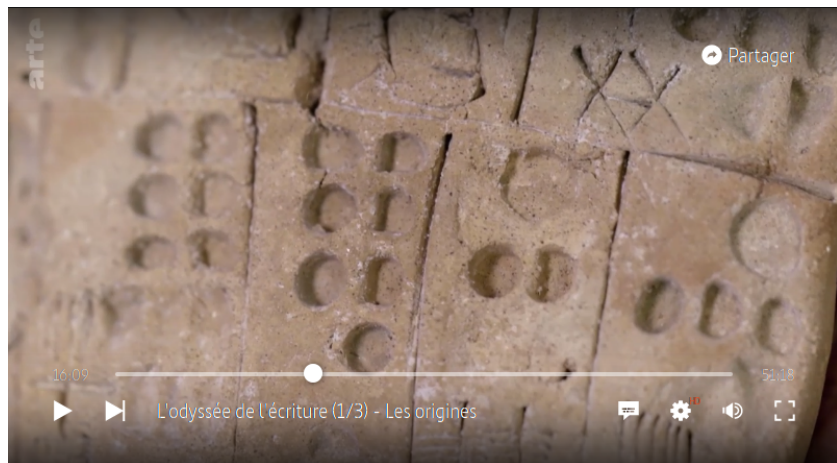
On parle de Systèmes d'écriture (writing systems)

Deux grandes classes :

- ▶ Les écritures de type phonologique où on écrit les phonèmes (alphabets + syllabaires)
- ▶ Les écritures sémantiques où on écrit les morphèmes (écritures chinoises, hiéroglyphiques ou cunéiformes)

# L'odyssée de l'écriture (1/3) : documentaire d'Arte

Les origines





## Exemple : le système d'écriture japonais

UNICODE漢字コード分かりません。

- ▶ « kanji », idéogrammes chinois : 漢字 (caractères)
- ▶ « hiragana », syllabaire pour les mots et la morphologie japonaise : 分かりません : wakarimasen (ne comprends pas)
- ▶ « katakana », syllabaire pour les emprunts aux langues étrangères : コード : cōdo (code)
- ▶ latin, pour les chiffres et les acronymes : UNICODE

Note : il n'y a pas de séparateur entre les mots...

### Problèmes informatiques en vrac

Prononciation des kanjis, découpage d'un texte en mots, saisie (plus de caractères que de touches sur le clavier...), typographie selon le sens d'écriture (horizontal/vertical), etc.

## Exemple : le système d'écriture arabe

- ▶ est un abjad : un système d'écriture ne notant que les consonnes (ou presque),
- ▶ s'écrit de droite à gauche (sauf les nombres, écrits de gauche à droite...)
- ▶ pas de distinction majuscule/minuscule, les lettres s'attachent entre elles,
- ▶ la forme des lettres diffère suivant leur contexte (seules, en début de mot, en fin de mot, entre deux lettres)
- ▶ une espace sépare les mots, mais on en trouve aussi (un plus petit) à l'intérieur de mots qui contiennent des lettres ne s'attachant pas

### Problèmes informatiques en vrac

Quelles sont les voyelles (qui portent la morphologie), typographie fine (on ne justifie pas un texte en agrandissant les espaces, mais en allongeant les lettres), Sélection discontinue (si il y a un nombre, notamment), etc.

# L'odyssée de l'écriture (2/3) : documentaire d'Arte

Le système d'écriture de l'arabe



Et vous ?

Décrivez brièvement les particularités de votre système d'écriture maternel...

## Les langues des humains

### La « langue » de l'ordinateur

- Traces

- Bits, octets, caractères

- Normes

- Unicode

- Limitations d'Unicode

## Les langues et l'informatique

### Conclusion

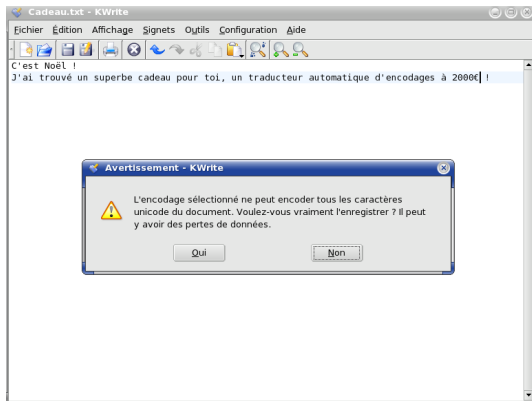
### Pour finir



# 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

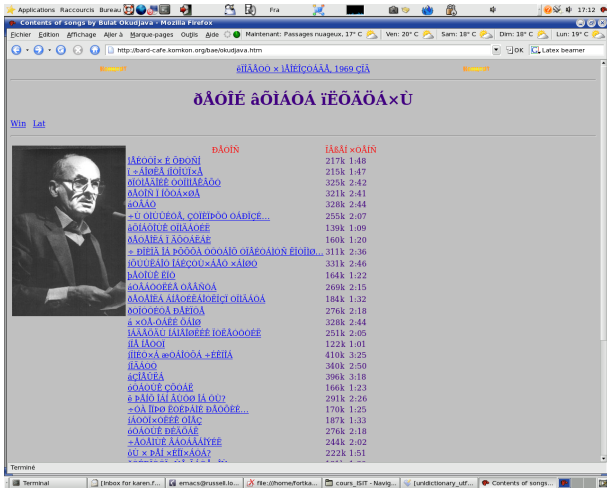
Un ou plusieurs caractère(s) que nous avons pu écrire sont apparemment **Unicode (?)** et ne sont pas enregistrables avec l'**encodage par défaut**

# 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)



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



## Code source de la page

```

view-source: - Source de : http://bard-cafe.komkon.org/bae/okudjava.htm - Mozilla Firefox
Fichier Edition Affichage

<html>
<head>
<title>
Contents of songs by Bulat Okudjava
</title>
</head>

<BODY BACKGROUND="" BGCOLOR="#c0c0c0" TEXT="#400080" LINK="#0080ff" VLINK="#8080c0" ALINK="#00ffff">

<center>
<TABLE width=70% border=0>
<TR>
<TD width=20%><p align=left></p></TD>
<TD width=50%><center><a href="okudjava/koncert.ram">БІЛІАДОО × ІАІЕІ[ОААА, 1969 (ІА</a></center></TD>
<TD width=20%><p align=right></p></TD>
</TR>
</TABLE>
</center>
<br>

<h1><center><b>ОДІЕ ОІАОА ІЕОАДАУ</b></center></h1><p>

<a href="okudjav.htm">Win</a><nbsp;<b><a href="okudjavt.htm">Lat</a><b>
<br>
<p>

<table border=0>
<tr valign=bottom>
<td rowspan=15>

<td><center><font color=red>ОДІН</font></center>
<td><center><font color=red>ІАБІН</font></center>
<td><center><font color=red>ОАІН</font></center>
<tr><td><a href="okudjava/1.ra"> ІАЕОІ× Е ОЕОІН</a><td>217k<td>1:48
<tr><td><a href="okudjava/2.ra"> І АІЕА ІІОІІ×</a><td>215k<td>1:47
<tr><td><a href="okudjava/3.ra"> ОІОІАІЕЕ ООІІІАЕАОО</a><td>325k<td>2:42
<tr><td><a href="okudjava/4.ra"> ОАОІ І ІОА×ОА</a><td>321k<td>2:41
<tr><td><a href="okudjava/5.ra"> ОАААО</a><td>328k<td>2:44
<tr><td><a href="okudjava/6.ra"> ОУ ОІОІЕОА, ОІЕІРД ОАІЕ...</a><td>255k<td>2:97
<tr><td><a href="okudjava/7.ra"> ОІОАІІЕ ОІААЕЕ</a><td>139k<td>1:09
<tr><td><a href="okudjava/8.ra"> ОААІЕА І АООАЕА</a><td>160k<td>1:20
<tr><td><a href="okudjava/9.ra"> О ІЕІА ІА РООА ООАІО ОІАЕОАІО ОІОІО...</a><td>311k<td>2:36

```

<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

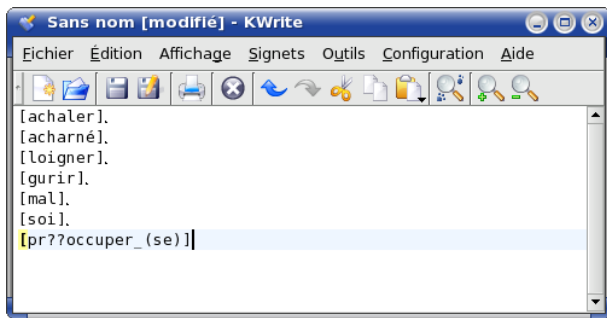
# Une page mal fichue

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)
- ▶ S'assurer de la capacité du logiciel à gérer « proprement » la langue

# 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

## 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

## Exercice

Combien existe-t'il de possibilités de combiner 2 bits ?

→ 00 ou 11 ou 01 ou 10 = 4 ( $2^2$ ) possibilités

Petite parenthèse : ça ne vous rappelle rien ?



(c) DVD Les shadoks, édition intégrale

Compter



## 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**
- ▶ 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)

### À noter

1 octet = 8 bits = 1 **byte** (en anglais)

# 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

## À noter

1 caractère = 1 octet (attention, simplification !)

# Le byte et ses bits

## Exercice

Combien de valeurs peut-on coder sur un octet ?

# Le byte et ses bits

## Exercice

Combien de valeurs peut-on coder sur un octet ?

→ 1 octet = 8 bits

→  $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8 = 256$

## Quelques exemples

Caractère	Code Binaire	Description
A	01000001	Caractère « A »
a	01100001	Caractère « a »
T	01010100	Caractère « T »
t	01110100	Caractère « t »
3	00110011	Caractère « 3 »
\$	00100100	Caractère « \$ »
BEL	00000111	Bip

### À noter

- ▶ pour passer de la majuscule à la minuscule, il suffit de mettre le troisième bit à 1
- ▶ les caractères chiffres ont leur propre code
- ▶ il n'existe qu'un « A », qui s'affiche différemment selon les polices ou le style (voir Définitions)



# Combien de caractères ?

o

A

FBA6

---

*A*

~

FBA7

---

A

l

a

FBA8

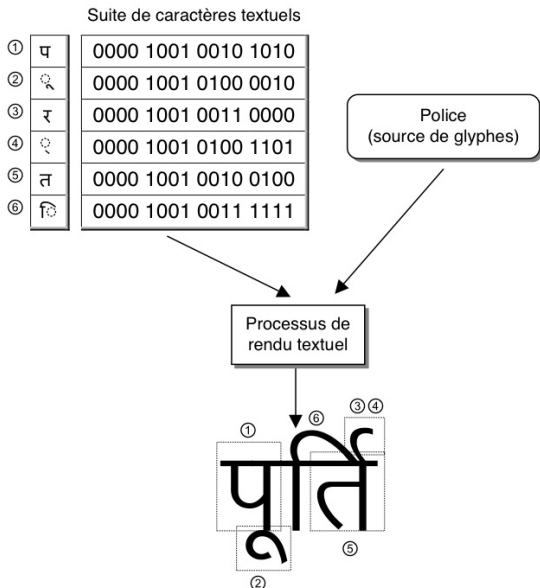
---

*a*

l

FBA9

# Caractères vs glyphs



# Quelques définitions

- Caractère** Entité abstraite, considérée comme atomique, servant à l'écriture d'une langue,
- Glyphe** Forme graphique abstraite prise par un caractère dans un certain contexte,
- Fonte** Ensemble de formes graphiques concrètes représentant un ensemble de glyphes,

# Combien de caractères ?

o

A

FBA6

---

*A*

~

FBA7

---

A

l

a

FBA8

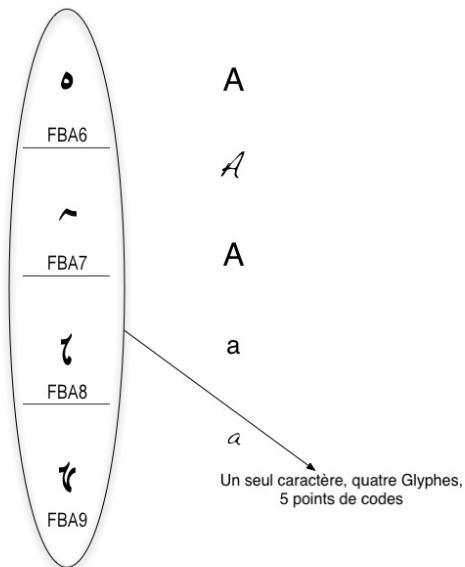
---

*a*

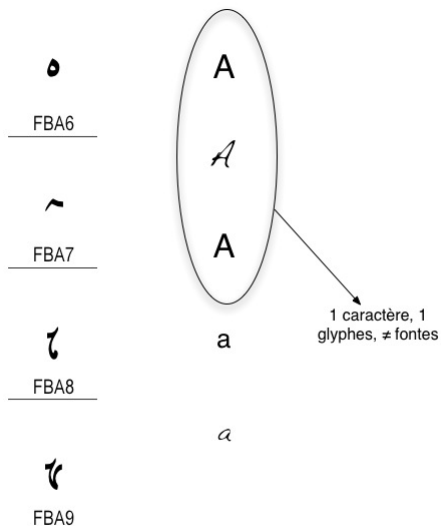
l

FBA9

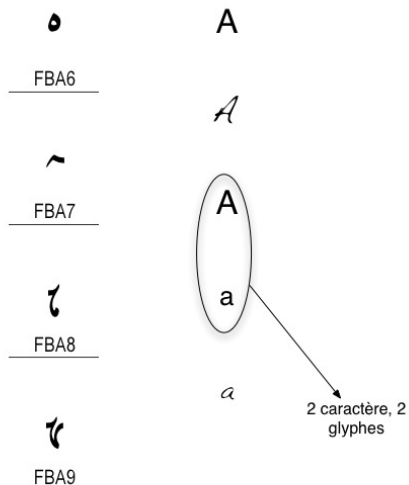
# Combien de caractères ?



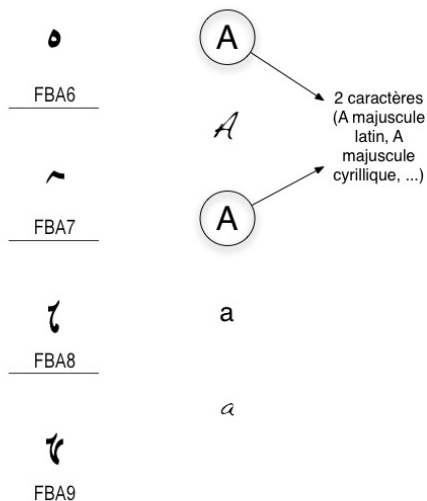
# Combien de caractères ?



# Combien de caractères ?



# Combien de caractères ?





Dites « Bonjour » à la dame

### Exercice

Combien de bits dans « Bonjour » ?

# Dites « Bonjour » à la dame

## Exercice

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

## Exercice

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

# Pas si simple

## Exercice

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

→ 1 Ko =  $2^{10}$  octets = 1 024 octets

# Pas si simple

## Exercice

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

→ 1 Ko =  $2^{10}$  octets = 1 024 octets

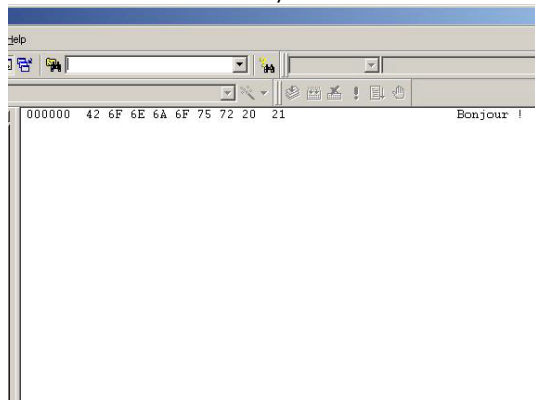
## À noter

Pour les fabricants de matériel, depuis la normalisation de 1998, due à un procès : 1 Ko =  $10^3$  octets = 1 000 octets

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

## Exercice

Comment écrire 10 en base 16 ?

# J'apprends à compter

## Exercice

Comment écrire 10 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 plate-forme 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		ê	ë	è		ï	î	ì	ñ	
9	é	æ	æ	ô	ö	ò	û	ù	ÿ	ö	ü	ç	£	¥	¤	
A	á	í	ó	ú	ñ	ñ						½	¾	¿	«	»
B								n								
C					—	+								=		
D																
E	α	β	Γ	Π	Σ	σ	μ	τ	ϑ	θ	Ω	δ	ω	ø	€	
F	≡	±	≥	≤		J	÷									

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

- ▶ À 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																
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																
Ax		ı	ç	£	¤	¥	¦	§	¨	©	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

# 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																
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																
Ax	NBSP	Ё	Ђ	Ѓ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	SHY	Ў	а
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	ѐ	ђ	ѓ	є	ѕ	і	ї	ј	љ	њ	ћ	ќ	џ	ѡ	Ѣ	ѣ

## 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 aviez assez ?

Évidemment, parmi les encodages les plus courants se trouve des encodages « maison » :

- ▶ [Microsoft](#) : Windows1252 et al.
- ▶ [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 6 763 caractères seulement (!!), et **Big5** pour le traditionnel, avec 13 053 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 6 763 caractères seulement (!!), et **Big5** pour le traditionnel, avec 13 053 caractères.
- ▶ **13 053 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 6 763 caractères seulement (!!), et **Big5** pour le traditionnel, avec 13 053 caractères.
- ▶ **13 053 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)

# Les éléphants d'Asie

## Exercice

Combien de valeurs peut-on représenter sur 16 bits ?

# Les éléphants d'Asie

## Exercice

Combien de valeurs peut-on représenter sur 16 bits ?

→  $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^{16} = 65\,536$

→ Ça suffit donc pour le chinois, même traditionnel. D'ailleurs, même 14 bits auraient suffi. . . ( $2^{14} = 16\,384$ )

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 12.1 (mai 2019) contient 137 994 caractères (dont 137 766 graphiques)

# 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**
  - ▶ 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 : Internet, Table des caractères (Windows), KCharSelect (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 sud-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 nécessitent 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 : □ + □ + □ + □

# Qui c'est le meilleur ?

- ▶ Avantages et inconvénients de chaque UTF ?
- ▶ **UTF-32** est de longueur fixe, donc 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 une 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 et Mac) 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 et Mac) 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])
- ▶ 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 appelé **BOM (Byte Order Mark)** et codé U+FEFF (big-endian) ou FFFE (little-endian)

## À noter

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»¿**

## Limitations d'Unicode (1/2)

- ▶ Le Consortium Unicode est agité de nombreux débats, ce qui est plutôt bon signe !
- ▶ Exemple 1 : le Consortium Unicode et l'ISO considèrent que les caractères chinois, coréens et japonais sont les mêmes, que seuls les glyphes diffèrent... ce qui fait l'objet d'un débat houleux.
- ▶ Exemple 2 : certaines écritures africaines représentant des populations importantes (le tifinagh par exemple) ne sont pas aussi bien représentées que des écritures américaines bien moins utilisées (le déséret, par exemple).

## Limitations d'Unicode (2/2)

- ▶ Exemple 3 : certains Bretons se plaignent de l'absence du K barré et de caractères uniques pour représenter CH et C'H.
- ▶ Exemple 4 : d'aucuns reprochent à Unicode d'avoir apparemment favorisé certaines écritures en les codant d'une façon plus simple ou plus adéquate. Le gothique, par exemple, n'est pas codé en tant que tel et son affichage correct nécessite l'ajout d'un protocole de niveau supérieur ou de caractères de commande à sa translittération latine.



WATCHING THE UNICODE PEOPLE TRY TO GOVERN THE INFINITE CHAOS OF HUMAN LANGUAGE WITH CONSISTENT TECHNICAL STANDARDS IS LIKE WATCHING HIGHWAY ENGINEERS TRY TO STEER A RIVER USING TRAFFIC SIGNS.

Les langues des humains

La « langue » de l'ordinateur

Les langues et l'informatique

Désigner une langue

Conclusion

Pour finir

## Comment désigner une langue ?

Donner un nom à une langue est délicat. En effet, le nom des langues désignées est défini par la langue d'usage.

ex. : allemand, deutsch, German, nemško, tedesco, ... désigne la langue allemande (en français, allemand, anglais, slovène, italien, ...)

Il est délicat de choisir une langue particulière pour faire cette désignation (ex : tout désigner en anglais...).

On ne peut pas non plus choisir de désigner une langue sous son nom "propre" (ex : English pour anglais, français pour français, Ting Vit pour le vietnamien,...). Cela serait délicat à écrire pour les langues orales, et difficiles à prononcer pour les langues à système d'écriture inconnu du lecteur...

# Comment désigner une langue ?

Il faut donc trouver un système pour nommer une langue qui soit :

- ▶ simple à mettre en œuvre,
- ▶ facilement transportable (utilisation uniquement de caractères ASCII),
- ▶ compréhensible par tous (via une norme).

Il s'agit de la norme ISO-639.

## la norme ISO-639

ISO 639 fournit trois<sup>4</sup> ensembles de codes de langues :

- ▶ ISO 639-1 (alpha-2) utilise des codes sur 2 caractères, et les associe avec les noms en français, en anglais et dans la langue elle-même ;
- ▶ ISO 639-2 (alpha-3) utilise des codes sur 3 caractères et a deux codages possibles : ISO 639-2/B (bibliographiques) et ISO 639-2/T (terminologiques) ; les codes sont associés avec des noms en français et en anglais.
- ▶ ISO 639-3 complète l'ISO 639-2 avec de très nombreuses autres langues encore manquantes ; SIL<sup>5</sup> en est l'auteur principal et est en charge des enregistrements dans ce volet de la norme ; seuls les noms en anglais sont associés aux codes ajoutés.

---

4. nouveau en 2007

5. [www.sil.org](http://www.sil.org)



## la norme ISO-639 (suite)

Ces trois ensembles ont des motivations différentes :

- ▶ ISO 639-1 (alpha-2) a été construit principalement pour des utilisation en terminologie, lexicographie et linguistique ; il contient des langues répandues et pour lesquelles il existe de nombreuses ressources terminologiques ;
- ▶ ISO 639-2 (alpha-3) a été construit pour des utilisations en bibliographie et terminologie ; il contient les langages de la partie 1, ainsi que de nombreux autres ayant une littérature abondante.
- ▶ ISO 639-3 vise à fournir une énumération de langues la plus complète possible, y compris les langues vivantes, les langues mortes, les langues anciennes et les langues construites artificiellement<sup>6</sup>, qu'elles soient majeures ou mineures, écrites ou orales.

---

6. comme l'espéranto, le volapük, le klingon, etc. il ne s'agit pas de langages artificiels

# Présentation de la norme ISO 639-3

L'ensemble des codes :

```
CREATE TABLE ISO_639-3 (  
  Id          char(3) NOT NULL,  
             -- The three-letter 639-3 identifier  
  Part2B      char(3) NULL,  
             -- Equivalent 639-2 identifier of the bibliographic applica  
             -- code set, if there is one  
  Part2T      char(3) NULL,  
             -- Equivalent 639-2 identifier of the terminology applicati  
             -- code set, if there is one  
  Part1       char(2) NULL,  
             -- Equivalent 639-1 identifier, if there is one  
  Scope       char(1) NOT NULL,  
             -- I(ndividual), M(acrolanguage), S(pecial)  
  Type        char(1) NOT NULL,  
             -- A(ncient), C(onstructed),  
             -- E(xtinct), H(istorical), L(iving), S(pecial)  
  Ref_Name    varchar(150) NOT NULL)  
             -- Reference language name
```

## Extrait de la table des codes

<b>Id</b>	<b>Part2B</b>	<b>Part2T</b>	<b>Part1</b>	<b>Scope</b>	<b>Type</b>	<b>Ref_Name</b>
aao				I	L	Algerian Saharan Arabic
abh				I	L	Tajiki Arabic
ara	ara	ara	ar	<b>M</b>	L	Arabic
czh				I	L	Huizhou Chinese
deu	ger	deu	de	I	L	German
epo	epo	epo	eo	I	<b>C</b>	Esperanto
fra	fre	fra	fr	I	L	French
frm	frm	frm		I	<b>H</b>	Middle French (ca. 1400-1600)
fro	fro	fro		I	H	Old French (842-ca. 1400)
mis	mis	mis		S	S	Uncoded languages
mjy				I	<b>E</b>	Mahican
mul	mul	mul		<b>S</b>	<b>S</b>	Multiple languages
nut				I	L	Nung (Viet Nam)
und	und	und		<b>S</b>	<b>S</b>	Undetermined
vie	vie	vie	vi	I	L	Vietnamese
zho	chi	zho	zh	M	L	Chinese
zxx	zxx	zxx		S	S	No linguistic content

# Présentation de la norme ISO 639-3

Le mapping des "macrolangues" :

```
CREATE TABLE ISO_639-3_Macrolanguages (  
  M_Id  char(3) NOT NULL,  -- The identifier for a macrolanguage  
  I_Id  char(3) NOT NULL)  -- The identifier for an individual language  
                           -- that is a member of the macrolanguage
```

## Extrait de la table des Macrolangues

M_Id	I_Id	Explication
ara	aao	<u>Algerian Saharan Arabic</u> fait partie de <u>Arabic</u>
ara	abh	<u>Tajiki Arabic</u> fait partie de <u>Arabic</u>
ara	abv	
zho	cjy	
zho	cmn	
zho	cpx	
zho	czh	<u>Huizhou Chinese</u> fait partie de <u>Chinese</u>

## Autres informations en vrac

- ▶ il existe une table décrivant les code ayant été retirés (si vous ne trouvez pas le votre...)
- ▶ les codes qaa à qtz sont des codes réservés à un usage local ; ils ne doivent être transmis qu'après accord entre les parties
- ▶ le ISO 639-3 Registration Authority est SIL International, [www.sil.org](http://www.sil.org) (Dallas, Texas)
- ▶ la table des codes contient 7642 entrées (430 langues mortes, 4 spéciales, 63 historiques, 114 anciennes, 17 construites)

## 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"`).
- ▶ **Agglutations** (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 raccourcis ? Même pas peur !

- ▶ Il existe un UTF-7, utilisé pour les mails, par exemple.

# Les raccourcis ? Même pas peur !

- ▶ Il existe un UTF-7, utilisé pour les mails, par exemple.
- ▶ Des vrais chiffres peuvent être déclarés pour que l'ordinateur fasse des opérations dessus.

## Les raccourcis ? Même pas peur !

- ▶ Il existe un **UTF-7**, utilisé pour les mails, par exemple.
- ▶ Des **vrais chiffres** peuvent être déclarés pour que l'ordinateur fasse des opérations dessus.
- ▶ Il existe un troisième jeu de caractères en chinois, le **nushu**, utilisé uniquement par des femmes...

# Les outils qui sauvent

- ▶ Bon éditeur de texte brut : Notepad++ (Windows), Sublime Text (Windows, Mac, Linux), Gedit (Linux)
- ▶ Editeur qui permette l'ouverture en binaire (hexa) : Visual, Eclipse (outils de dev.)

## Définitions (Wikipedia) 1/3

- ▶ **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.

Les langues des humains

La « langue » de l'ordinateur

Les langues et l'informatique

Conclusion

**Pour finir**

CQFR : Ce Qu'il Faut Retenir

TD





- ▶ bit, octet
- ▶ ASCII, tables ISO, Unicode
- ▶ ISO 639

## Exercice non noté

La fonction *ord* de python3 renvoie le code unicode d'un caractère.  
*chr* est la fonction inverse.

La liste [233, 112, 97, 116, 97, 110, 116, 32, 33] correspond à la liste des codes unicodes d'une chaîne.

Laquelle ?