

# Persistante des données I

## DON2

Denis Boigelot, Geneviève Cuvelier, Selim Rexhep,  
Yannick Voglaire



Haute École Bruxelles-Brabant  
École Supérieure d'Informatique

Année académique 2020 / 2021

## Plan du cours

- 0 – Présentation**
- 1 – Introduction**
- 2 – Dépendance fonctionnelle**
- 3 – Schéma conceptuel**
- 4 – Projection et sélection**
- 5 – Jointure**
- 6 – Agrégat**
- 7 – Sous-requête**
- 8 – Fichiers**

# Table des matières

- 1 **Introduction**
- 2 **Les fichiers binaires**
- 3 **Les fichiers textes**
- 4 **Les formats de données textuelles**
- 5 **Manipulation avec Java**

# Table des matières

- 1 **Introduction**
- 2 **Les fichiers binaires**
- 3 **Les fichiers textes**
- 4 **Les formats de données textuelles**
- 5 **Manipulation avec Java**

# Introduction

## Fichier

Un **fichier** informatique est une unité informationnelle physiquement stockée sur un support de mémoire de masse, non volatile.

Partageons les fichiers en deux grandes catégories :

- ◊ les fichiers binaires,
- ◊ les fichiers textes.

## Introduction

## les fichiers binaires

Un **fichier binaire** est fichier dont les données sont stockées sous forme d'octets qui n'ont donc de sens que pour le logiciel qui les utilise.

**Figure** – Fichier .odt lu avec un éditeur de texte

# Introduction

## les fichiers textes

Un **fichier texte** (ou fichier plat) est un fichier dont le contenu représente uniquement une suite de caractères. Il peut être lu avec un éditeur de texte.



**Figure –** Fichier .txt lu avec un éditeur de texte

# Introduction

Si par définition tout fichier est binaire, l'usage veut que l'on qualifie un fichier de binaire pour indiquer que les octets du fichier ne représentent pas des caractères.

# Introduction

Les fichiers binaires peuvent être ouverts par des éditeurs de texte (déconseillé aux utilisateurs non expérimentés), mais les données y seront mal représentées et surtout, incompréhensibles.

Pourquoi ? Tout simplement car on demande à l'éditeur en question d'interpréter les octets du fichier comme des caractères, alors que le fichier n'a pas été conçu pour ça !

# Introduction

Pour pouvoir exploiter les données d'un fichier, il n'est pas suffisant de savoir que ce fichier est un fichier texte ou binaire. En effet, rien n'est encore dit sur la façon dont les données sont codées au moyen d'octets !

Nous avons besoin de la notion de **format**, qui nous fournira les détails du codage des données.

# Introduction

Exemple : Si on sait que le fichier XXX est binaire, comment savoir si les octets qu'il contient représentent une image ? Du son ? Une vidéo ? Et si on sait que les octets représentent une image, comment reconstituer l'image et l'afficher à l'écran à partir des octets ? C'est précisément la notion de format qui nous fournira cette information.

# Introduction

**Format de données** Façon de représenter (coder) les données sous forme d'une suite de bits. Un format de donnée suit une convention (éventuellement normalisé) pour représenter les données (texte, image, son...).

**Codification** Action de représenter l'information sous un format de données donné.

# Introduction

Le format d'un fichier est parfois (mais pas toujours !) représenté par son **extension**, c'est-à-dire le suffixe .XYZ qui se trouve à la fin de son nom.

Exemple de formats de fichiers images (binaires) : .bmp (bitmap), .jpeg (Joint Photographic Experts Group), .png (Portable Network Graphics).

# Introduction

Il existe donc différentes façons d'encoder une image.

Remarque : un fichier représentant une image est-il forcément un fichier binaire ? Réponse : non ! Il est parfaitement possible de représenter certaines images au moyen de fichiers textes (nous donnerons un exemple dans les slides qui suivent).

# Introduction

Pour encoder les caractères d'un fichier texte, il existe aussi différents formats (qui seront étudiés à la fin de ce chapitre).

Exemples : les fichiers texte brut ont pour extension .txt, mais le format peut malgré tout varier ! Les fichiers .java contiennent du texte représentant du code Java, les fichiers .c contiennent du texte représentant du code écrit en langage C, ...

# Introduction

Dans la suite du chapitre, nous passerons en revue la façon d'encoder certaines données couramment rencontrées en informatique et dans la vie de tous les jours (les couleurs, le son, les caractères ... ).

# Fichiers structurés

Et pour une base de données relationnelle ? Au bout du compte, les données des différentes tables seront aussi stockées dans des fichiers ! Mais de quelle façon ?

Cela dépend du SGBD utilisé, mais nous pouvons donner quelques indications générales.

# Fichiers structurés

Chaque ligne (on parle aussi d'**enregistrement**) d'une table donnée de la DB doit être stockée dans un fichier :

ligne	DDJZ	MUN11	112, r. Neuve	Genève	DZ	U
	B512	GILLET	14, r. de l'Eté	Toulouse	B1	-8700
	C003	AVRON	8, r. de la Cure	Toulouse	B1	-1700

Cette ligne est constituée d'une suite de champs contenant des valeurs. Le SGBD enregistre alors, pour chaque enregistrement, un chaîne d'octets constituée de la suite des valeurs des différents champs ainsi que d'autres informations techniques éventuelles (longueur totale de l'enregistrement, type de chaque donnée, etc.)

# Fichiers structurés

La notion d'enregistrement constitue alors l'**unité de lecture/écriture** du fichier. Le SGBD ne peut lire une partie d'un enregistrement, il le lit en entier ou pas du tout

Remarque : Est-ce qu'il suffit de ranger les enregistrements les uns à la suite des autres dans un fichier ?

# Fichiers structurés

Imaginons une table contenant 10 millions d'enregistrement (un tel nombre est courant dans les bases de données professionnelles) rangées séquentiellement les uns à la suite des autres dans un ou plusieurs fichiers.

Que se passe-t-il si le SBGD doit régulièrement lire des lignes au hasard dans la table ?

# Fichiers structurés

Nous pourrions décider de ranger les enregistrements selon un certain ordre de façon à pouvoir utiliser un algorithme de recherche performant<sup>1</sup> mais que se passera-t-il lorsque le SGBD devra insérer un enregistrement au milieu d'un fichier ?

---

1. Cela dit, est-il envisageable de charger 10 millions d'enregistrements en mémoire centrale ?

# Fichiers structurés

Pour des raisons de performance, les fichiers contenant les enregistrements sont munis de structures particulières permettant de manipuler les enregistrements aussi rapidement que possible. En particulier, l'**accès direct** à un enregistrement est souhaitable (pour ne pas devoir parcourir les 10 millions de ligne à chaque fois).

De tels fichiers sont parfois appelés **fichiers structurés**.

# Fichiers structurés

Les méthodes d'organisation des fichiers structurés ne seront pas étudiées dans ce cours-ci, le lecteur intéressé trouvera de plus amples détails dans le chapitre 4 du livre de Jean-Luc Hainaut.<sup>2</sup>

---

2. Bases de données Concepts, utilisation et développement. 4 ième édition.  
J-L Hainaut - Dunod, 2018

# Table des matières

## 1 Introduction

## 2 Les fichiers binaires

- Vocabulaire
- Codification

## 3 Les fichiers textes

## 4 Les formats de données textuelles

## 5 Manipulation avec Java

## Bit

Un *bit* (**Binary digit**) est l'unité de base de l'informatique. Deux valeurs sont possibles. Le 0 et le 1.

Notons que :

- ◊ ajouter 1 bit permet de doubler le nombre de valeurs possibles ;
- ◊ 4 chiffres binaires permettent de représenter un chiffre hexadécimal ;
- ◊ tout ce qui est affiché à l'écran est codé en binaire.

# Vocabulaire

## Byte

Un *byte* désigne autant en français qu'en anglais une unité digital d'information. Souvent composée de 8 bits (mais pas toujours).

Représente la plus petite unité adressable sur beaucoup d'architecture.

## Octet

Le terme *octet* permet de lever l'ambiguïté du nombre de bits composant un byte. Un octet est un byte de 8 bits (en anglais et en français).

# Les nombres

La valeur décimale est calculée en additionnant le produit du chiffre binaire par la valeur correspondant à sa position.

...	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
...	0	0	0	0	0	0

- ◊  $101010 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 32 + 8 + 2 = 42$
- ◊  $11111111 = \dots = 255$
- ◊  $00000000 = 0$

# Les couleurs

## Synthèse additive

Combinaison de plusieurs couleurs afin de former une nouvelle couleur. Généralement le Rouge, le vert et le bleu (RVB ou RGB en anglais) à raison d'un octet par couleur. Au total 24 bits et donc 16 millions de couleurs possibles (true colors).

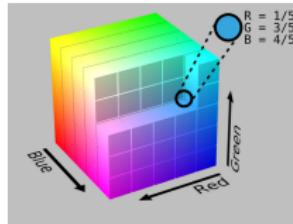
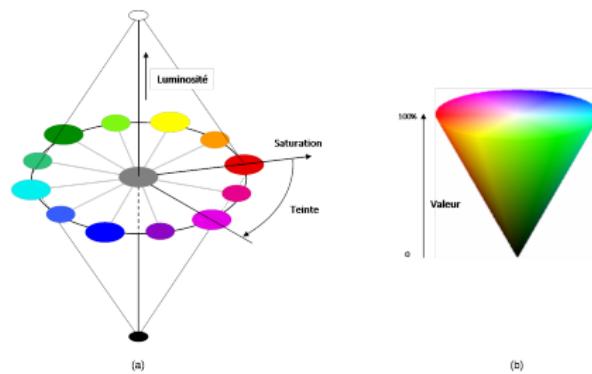


Figure – Couleurs RVB

# Les couleurs

## Forme vectorielle

Enregistrement de la couleur sous forme vectoriel selon 3 paramètres. La teinte, la saturation et la lumière.



**Figure – Couleur vectorielle**

# Le noir et blanc

## Nuance de gris

Une image en nuance de gris correspond à ce qui est communément appelée image en noir et blanc. Il est possible d'obtenir les nuances de gris avec le système RGB en mettant la même valeur pour R, G et B et de (0,0,0) pour le noir à (255,255,255) pour le blanc.

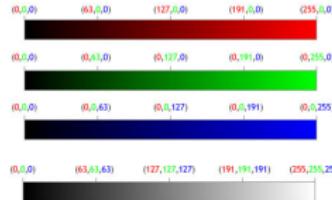


Figure – Nuances de couleur

# Le noir et blanc

## Noir et blanc

Pour une image en noir et blanc, uniquement deux valeurs sont possibles ; noir et blanc. Seulement 1 bit est suffisant pour définir la couleur (1=noir, 0=blanc).

# Les images

Les écrans sont composés de pixels.

**HD** 1920 colonnes × 1080 lignes.

**UHD** 3840 colonnes × 2160 lignes.

**4K** 4096 colonnes × 2160 lignes (cinéma).

## Image

Une image regroupe les informations nécessaires à son affichage sur un écran.

# Les images matricielles

## Image matricielle

Une image matricielle est une correspondance, entre chacun des pixels qui la compose et une couleur.

## Pixellisation

Lorsque l'image est agrandie, plusieurs pixels de l'écran sont utilisés pour afficher un même pixel de l'image. Les pixels de l'image deviennent alors visible à l'œil nu.

# Les images matricielles

## Réflexion

Si l'on considère que l'on connaît la taille de la matrice, la position des pixels dans la matrice peut être déduite par leur position dans la séquence.

## Informations nécessaires

Le format de donnée doit donc contenir :

- ◊ la palette de couleur,
- ◊ la taille de l'image
- ◊ et la liste des couleurs de chaque pixel.

# Les images matricielles

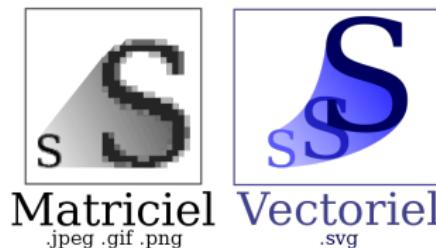
## Exemple

Prenons l'exemple d'une image dont la palette de couleur est noir et blanc, de taille  $5 \times 5$  et dont la matrice est définie par la suite de bits  $\{1000101010001000101010001\}$  est une croix.

# Les images vectorielles

## Image vectorielle

Une image vectorielle est définie par un ensemble d'objets mathématiques tels que des points, des cercles... À cela, on peut ajouter des attributs comme l'épaisseur, la couleur ; mais aussi des effets.



**Figure –** Image matricelle - vectorielle

# Les sons

## PCM

Le procédé de codification du son le plus courant est celui dit de modulation d'impulsion codée (abrégé PCM). Le son est d'abord échantillonné, l'échantillon obtenu est alors quantifié.

## CD

L'échantillonnage sur un CD se fait à raison de 41,1kHz et la quantification se fait sur 16 bits.

# Les sons

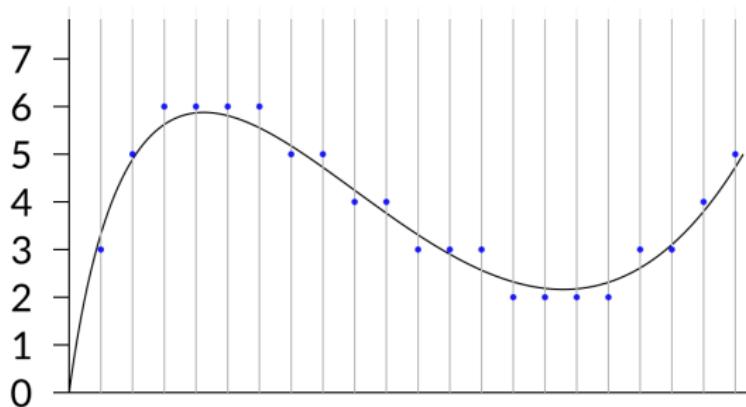


Figure – Échantillonnage en gris, quantification sur 3 bits en bleu.

## Résultat de la codification

```
011 101 110 110 110 110 101 101 100 100 011 011  
011 010 010 010 010 011 011 100 101
```

# Autre

## Texte

Chaque caractère est associé à un code binaire (voir section suivante).

## vidéo

Pour rappel, une vidéo est constituée :

- ◊ de sons,
- ◊ d'images,
- ◊ de texte (sous-titre).

À votre imagination...

# Table des matières

## 1 Introduction

## 2 Les fichiers binaires

## 3 Les fichiers textes

- Introduction
- ASCII
- UNICODE
- UTF-8/16/32

## 4 Les formats de données textuelles

## 5 Manipulation avec Java

# Introduction

## Fichier texte

Un fichier texte est un fichier binaire. Le sens de la succession de bits est une suite de caractères codifiés selon un format donné tel que l'ASCII, le Latin-1 ou mieux, l'UTF-8.



- ◊ Un fichier Writer/Word est-il un fichier texte ?
- ◊ Donnez au moins un type de fichier texte.

# ASCII - ISO/CEI 646

## American Standard Code for Information Interchange (ASCII)

ASCII (1977/1986)																
	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1_16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2_32	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3_48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	-
6_96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_112	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
	<input type="checkbox"/>	Letter	<input type="checkbox"/>	Number	<input type="checkbox"/>	Punctuation	<input type="checkbox"/>	Symbol	<input type="checkbox"/>	Other	<input type="checkbox"/>	undefined	<input type="checkbox"/>	Changed from 1963 version		

Figure – <https://en.wikipedia.org/wiki/ASCII>

# ASCII - ISO/CEI 646

## Codage

À l'origine, les caractères sont codés sur 7 bits. 128 valeurs sont donc possibles. Actuellement, il est codé sur un octet. Le 8<sup>ème</sup> vaut 0.

Dans les faits, il contient 95 caractères imprimables :

- ◊ les chiffres arabes [0-9],
- ◊ les lettres latines [a-z,A-Z],
- ◊ des symboles mathématiques {+, -, /},
- ◊ les caractères de ponctuation et quelques autres.

# ASCII - ISO/CEI 646

## Caractères non imprimables

Parmi les 128 valeurs, Les 32 premiers caractères ainsi que le dernier ne sont pas affichable.

Quelques exemples :

- 7** Bell (bip sonore),
- 10** LF (line feed),
- 13** CR (carriage return),
- 127** DEL (effacer).

# ASCII étendu

## Constat

Pour écrire un texte en français, il manque certains caractères ({è, é, à, ù...}).

## extension de l'ASCII

Pour rappel, on utilise 7 bits parmi les 8 bits d'un octet. Il en reste 1 de non utilisé qui permet d'associer 128 caractères.

# ASCII étendu

## Problèmes

- ◊ Extension propre à la langue,
- ◊ Pas suffisant pour certaines langues.

## Exemples

- ◊ ISO-8859-1 (Western),
- ◊ WINDOWS-1252,
- ◊ ISO-8859-5,
- ◊ ...

Jouons avec les formats.

# ASCII étendu : ISO-8859-1 alias Latin-1

ISO/CEI 8859-1																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x																
1x	<i>positions inutilisées</i>															
2x	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8x																
9x	<i>positions inutilisées</i>															
Ax	NBSP	í	¢	£	¤	¥	¦	§	“	©	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	Ї	Ї
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	Þ
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	ї	ї
Fx	ó	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Figure – ASCII étendu (ISO-8859-1)

# ASCII étendu : ISO-8859-5 alias Google it

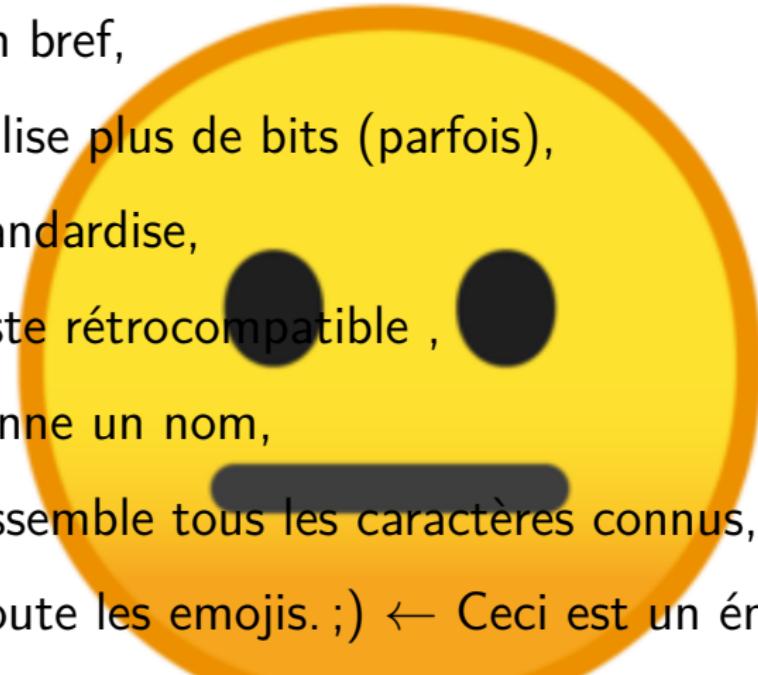
ISO/CEI 8859-5:1999																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	Inutilisé															
1x	Inutilisé															
2x	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	А	В	С	Д	Е	Ғ	Ғ	Ҳ	Ӣ	Ҷ	Ҹ	ҹ	һ	ҵ	ҷ
5x	Р	ҽ	Ҿ	Ҵ	ҳ	Ҳ	ұ	Ұ	Ү	ү	ҭ	Ү	Ұ	ұ	Ҳ	Ү
6x	‘	а	в	с	д	е	ғ	ғ	ҳ	ӣ	ҷ	Ҹ	һ	ұ	Ҳ	Ү
7x	ҽ	Ҿ	Ҵ	ҳ	Ҳ	ұ	Ұ	Ү	ү	ҭ	ұ	Ҳ	ұ	Ҳ	Ү	ү
8x	Inutilisé															
9x	Inutilisé															
Ax	NBSP	Ё	Ҷ	Ҵ	Ҹ	Ү	Ұ	ұ	Ҳ	Ү	Ү	Ү	Ү	Ү	Ү	Ү
Bx	А	ҽ	Ҿ	Ҵ	ҳ	Ҳ	ұ	Ұ	Ү	Ү	Ү	Ү	Ү	Ү	Ү	Ү
Cx	Р	ҽ	Ҿ	Ҵ	ҳ	Ҳ	ұ	Ұ	Ү	Ү	Ү	Ү	Ү	Ү	Ү	Ү
Dx	а	ҽ	Ҿ	Ҵ	ҳ	Ҳ	ұ	Ұ	Ү	Ү	Ү	Ү	Ү	Ү	Ү	Ү
Ex	ҽ	Ҿ	Ҵ	ҳ	Ҳ	ұ	Ұ	Ү	Ү	Ү	Ү	Ү	Ү	Ү	Ү	Ү
Fx	№	ё	Ҷ	Ҵ	Ҹ	Ү	Ұ	ұ	Ҳ	Ү	Ү	Ү	Ү	Ү	Ү	Ү

Figure – ASCII étendu (ISO-8859-5)

# La route vers les emojis

Unicode en bref,

- ◊ on utilise plus de bits (parfois),
- ◊ on standardise,
- ◊ on reste rétrocompatible ,
- ◊ on donne un nom,
- ◊ on rassemble tous les caractères connus,
- ◊ on ajoute les emojis. ;) ← Ceci est un émoticône.



Hé, ce n'est pas poker face, mais « neutral face ».  
U+1F644

# UNICODE

## Unicode

Unicode est un standard de l'informatique pour l'encodage de la plupart des écritures du monde.

- ◊ Il est maintenu conjointement avec le standard ISO/IEC 10646.
- ◊ Implémenté dans de nombreux technologies récentes incluant les systèmes d'exploitation, le XML, le Java...
- ◊ La grande majorité des sites webs (> 92%) l'utilisent au travers de l'UTF-8.

# Vocabulaire Unicode

Commençons avec un peu de vocabulaire Unicode

**Character (caractère)** Plus petit élément d'écriture ayant une signification (espacement, ponctuation, lettre...).

Exemple : a, b, &, à (et beaucoup d'autres. Beaucoup !)

**Character set (charset)** Collection d'éléments utilisés pour représenter une information textuelle.

Exemple : {0, 1} (vive le binaire).

# Vocabulaire Unicode suite

**Coded character set** Ensemble de caractères dans lequel chaque caractère est associé à un *code point* numérique.

Exemple : { a → U+0061 ; & → U+0026 ... }

<https://unicode-table.com/>

**Code point** Valeur unique dans l'espace code de Unicode (0 → 10FFFF)

# Vocabulaire

**Code unit** Combinaison minimale de bits pouvant représenter un caractère.

Exemple 'a' :

**UTF-8** 01100001 (8 bits),

**UTF-16** 00000000 01100001 (16 bits).

## **Unicode Encoding Form (UEF)**

Attribution/mise en forme du code point avec une séquence unique de « codes units ».

# UTF-8

Type	Caractère	Point de code (hexadécimal)	Valeur scalaire		Codage UTF-8	
			décimal	binnaire	binnaire	hexadécimal
Contrôle	[NUL]	U+0000	0	00000000	00000000	00
	[US]	U+001F	31	00111111	00011111	1F
Texte	[SP]	U+0020	32	01000000	00100000	20
	A	U+0041	65	10000001	01000001	41
Contrôle	~	U+007E	126	11111110	01111110	7E
	[DEL]	U+007F	127	11111111	01111111	7F
	[PAD]	U+0080	128	00010 000000	11000010 10000000	C2 80
Texte	[APC]	U+009F	159	00010 011111	11000010 10011111	C2 9F
	[NBSP]	U+00A0	160	00010 100000	11000010 10100000	C2 A0
Texte	é	U+00E9	233	00011 101001	11000011 10101001	C3 A9
	□	U+07FF	2047	11111 111111	11011111 10111111	DF BF
	□	U+0800	2048	0000 100000 000000	11100000 10100000 10000000	E0 A0 80
	€	U+20AC	8 364	0010 000010 101100	11100010 10000010 10101100	E2 82 AC
	□	U+D7FF	55 295	1101 011111 111111	11101101 10011111 10111111	ED 9F BF

# Encodage de UNICODE

Dans les standards UNICODE, la façon d'encoder selon la taille des codes units est également donnée. Ainsi ;

**code units de 8 bits** avec l'UTF-8 encoding form ;

**code units de 16 bits** avec l'UTF-16 encoding form ;

**code units de 32 bits** avec l'UTF-32 encoding form .

# UTF-8

Il utilise de 1 à 4 octets. Sur le premier octet, on retrouve uniquement les caractères ASCII.

- ◊ Il est donc rétrocompatible avec l'ASCII,
- ◊ mais pas avec l'ASCII étendu. Pourquoi ?

**Selon le vocabulaire Unicode**, UTF-8 utilise des codes units de 8 bits. La lettre 'a' est encodé sur les 8 bits 01100001 ( $61_{hex}$ ).

# UTF-8

Caractères codés	Représentation binaire UTF-8	Premier octet valide (hexadécimal)	Signification
<b>U+0000 à U+007F</b>	<b>0xxxxxx</b>	00 à 7F	1 octet, codant 7 bits
<b>U+0080 à U+07FF</b>	<b>110xxxxx 10xxxxxx</b>	C2 à DF	2 octets, codant 11 bits
<b>U+0800 à U+0FFF</b>	<b>11100000 101xxxxx 10xxxxxx</b>	E0 (le 2 <sup>e</sup> octet est restreint de A0 à BF)	3 octets, codant 16 bits
<b>U+1000 à U+1FFF</b>	<b>11100001 10xxxxxx 10xxxxxx</b>	E1	
<b>U+2000 à U+3FFF</b>	<b>1110001x 10xxxxxx 10xxxxxx</b>	E2 à E3	
<b>U+4000 à U+7FFF</b>	<b>111001xx 10xxxxxx 10xxxxxx</b>	E4 à E7	
<b>U+8000 à U+BFFF</b>	<b>111010xx 10xxxxxx 10xxxxxx</b>	E8 à EB	
<b>U+C000 à U+CFFF</b>	<b>11101100 10xxxxxx 10xxxxxx</b>	EC	
<b>U+D000 à U+D7FF</b>	<b>11101101 10xxxxxx 10xxxxxx</b>	ED (le 2 <sup>e</sup> octet est restreint de 80 à 9F)	
<b>U+E000 à U+FFFF</b>	<b>1110111x 10xxxxxx 10xxxxxx</b>	EE à EF	
<b>U+10000 à U+1FFFF</b>	<b>11110000 1001xxxx 10xxxxxx 10xxxxxx</b>	F0 (le 2 <sup>e</sup> octet est restreint de 90 à BF)	4 octets, codant 21 bits
<b>U+20000 à U+3FFFF</b>	<b>11110000 101xxxxx 10xxxxxx 10xxxxxx</b>		
<b>U+40000 à U+7FFFF</b>	<b>11110001 10xxxxxx 10xxxxxx 10xxxxxx</b>	F1	
<b>U+80000 à U+FFFFF</b>	<b>1111001x 10xxxxxx 10xxxxxx 10xxxxxx</b>	F2 à F3	
<b>U+100000 à U+10FFFF</b>	<b>11110100 100xxxxx 10xxxxxx 10xxxxxx</b>	F4 (le 2 <sup>e</sup> octet est restreint de 80 à 8F)	

# UTF-8

Type	Caractère	Point de code (hexadécimal)	Valeur scalaire		Codage UTF-8	
			décimal	binaire	binaire	hexadécimal
Contrôle	[NUL]	U+0000	0	0000000	00000000	00
	[US]	U+001F	31	0011111	00011111	1F
Texte	[SP]	U+0020	32	0100000	00100000	20
	A	U+0041	65	1000001	01000001	41
Contrôle	~	U+007E	126	1111110	01111110	7E
	[DEL]	U+007F	127	1111111	01111111	7F
Texte	[PAD]	U+0080	128	00010 000000	11000010 10000000	C2 80
	[APC]	U+009F	159	00010 011111	11000010 10011111	C2 9F
Texte	[NBSP]	U+00A0	160	00010 100000	11000010 10100000	C2 A0
	é	U+00E9	233	00011 101001	11000011 10101001	C3 A9
	□	U+07FF	2047	11111 111111	11011111 10111111	DF BF
	□	U+0800	2048	0000 100000 000000	11100000 10100000 10000000	E0 A0 80
	€	U+20AC	8 364	0010 000010 101100	11100010 10000010 10101100	E2 82 AC
	□	U+D7FF	55 295	1101 011111 111111	11101101 10011111 10111111	ED 9F BF

# Exercice

Et si on essayait de retrouver les caractères cachés dans le contenu d'un fichier codé en UTF-8.

1111000010011111010010110110101  
01100001111000101000000110001001

# Vocabulaire

**Unicode Encoding Scheme (UES)** Un *Unicode Encoding Scheme* permet l'association des « codes units » avec des octets (incluant la prise en charge du *BOM (Byte Order Mask)*).

Unicode reprend 7 UES :

- ◊ UTF-8 (UEF → UES trivial),
- ◊ UTF-(16/32),
- ◊ UTF-(16/32)BE (Big endian),
- ◊ UTF-(16/32)LE (Little endian).

# UTF-16

## Codes units

Codes units de 16 bits. Par exemple, la lettre 'a' est encodé sur 16 bits (00000000 01100001 en binaire, 0061 en hexadécimale).

Character	Binary code point	Binary UTF-16	UTF-16 hex code units	UTF-16BE hex bytes	UTF-16LE hex bytes
\$ U+0024	0000 0000 0010 0100	0000 0000 0010 0100	0024	00 24	24 00
€ U+20AC	0010 0000 1010 1100	0010 0000 1010 1100	20AC	20 AC	AC 20
¥ U+10437	0001 0000 0100 0011 0111	1101 1000 0000 0001 1101 1100 0011 0111	D801 DC37	D8 01 DC 37	01 D8 37 DC
฿ U+24B62	0010 0100 1011 0110 0010	1101 1000 0101 0010 1101 1111 0110 0010	D852 DF62	D8 52 DF 62	52 D8 62 DF

Figure – <https://en.wikipedia.org/wiki/UTF-16>

# Character Encoding Scheme

Remarques,

- ◊ la plupart des protocoles de communication et de stockage sont définis pour des octets,
- ◊ l'ordre de lecture des bytes dépend de l'architecture des ordinateurs (endianness).

Deux manières d'encoder. Défini par

- ◊ le Byte Order Mark (code point U+FFFE en en-tête),
- ◊ ou décrit explicitement (UTF-16BE et UTF-16LE).



Hutomo Abrianto

UTF-32



# Liens

Quelques liens :

- ◊ <http://namok.be/blog/?post/2009/11/30/unicode-UTF8-UTF16-UTF32-et-tutti-quanti>,
- ◊ [http://unicode.org/faq/utf\\_bom.html#gen5](http://unicode.org/faq/utf_bom.html#gen5),
- ◊ <http://www.unicode.org/versions/Unicode11.0.0/ch03.pdf#G7404>,
- ◊ <https://unicode-table.com>,
- ◊ <https://fr.wikipedia.org>,
- ◊ <https://en.wikipedia.org>

# Table des matières

- 1 Introduction
- 2 Les fichiers binaires
- 3 Les fichiers textes
- 4 Les formats de données textuelles
  - CSV
  - Famille SGML
  - JSON
  - Properties
- 5 Manipulation avec Java

# Introduction

## Format de données textuelles

On parle de format de données textuelles lorsqu'on souhaite faire référence à la façon de représenter des données sous forme d'une suite de caractères.

- ◊ Permet de sauvegarder les données,
- ◊ transmettre les données entre différents programmes, ou entre un programme et un humain (configurations, import...),
- ◊ et permet une mise en forme d'un contenu différent selon les choix de chacun (SGML, html...)...

# CSV

## CSV

Le format CSV (Comma-Separated Values) est un format texte représentant des données tabulaires sous forme de valeurs séparées par des virgules (ou un point-virgule).

Souvent utilisé pour importer/exporter des données d'un logiciel utilisant des tableaux.

```
1 firstName,lastName  
2 Geneviève,Cuvelier  
3 Selim,Rexhep  
4 Sébastien,Drobisz
```

[codes/CSVExample.csv](#)

# XML

## Objectif

L'objectif était de définir un langage plus simple que le SGML, mais plus générique

Notamment utilisé pour :

- ◊ le transport des données entre des applications différentes (webs services) ;
- ◊ l'encodage de documents ;
- ◊ les fichiers de configurations

<https://www.w3schools.com/xml/>

# XML exemple

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <students>
3   <student firstName="G." lastName="Cuvelier"/>
4   <student firstName="S." lastName="Rexhep"/>
5   <student firstName="S." lastName="Drobisz"/>
6 </students>
```

[codes/XMLExample.xml](#)

Le SVG fait partie des nombreux formats basés sur le XML.

# HTML

## HTML

L'HyperText Markup Language est un langage de balisage conçu pour représenter le contenu et la nature des éléments d'une page web (Version actuelle HTML5).

- ◊ Titre, paragraphe,
- ◊ Section, en-tête,
- ◊ media,
- ◊ lien...

# JSON

JavaScript Object Notation (JSON) est un format dérivé de la notation des objets du langage JavaScript

Notamment utilisé pour :

- ◊ le transport des données entre des applications différentes, à l'instar du XML ;
- ◊ l'encodage de documents ;
- ◊ les fichiers de configurations

# JSON

```
1  {
2      "students": {
3          "student": [
4              { "fName": "G.", "lName": "Cuvelier" },
5              { "fName": "S.", "lName": "Rexhep" },
6              { "fName": "S.", "lName": "Drobisz" }
7          ]
8      }
9  }
```

[codes/JSONExample.json](#)

# Configuration java

## Properties java

Permet de configurer un logiciel java.

### Exemple Configuration de connexion postgresql

```
1 ip=postgresql  
2 port=5432  
3 db=esidb  
4 user=etu  
5 passwd=etu
```

[codes/config.properties](#)

# Table des matières

- 1 Introduction
- 2 Les fichiers binaires
- 3 Les fichiers textes
- 4 Les formats de données textuelles
- 5 Manipulation avec Java

- L'attaque du clone du Hello World
- Fichiers Textes

# Raised Hand with Part Between Middle and Ring Fingers

U+1F596

« Raised Hand with Part Between Middle and Ring Fingers »

```
1 public class Hello {  
2     public static void main(String[] args) {  
3         System.out.println("\ud83d\udd96");  
4     }  
5 }
```

Essayez chez vous !

# Java properties

```
1 Properties prop = new Properties();
2 String propFileName = "resources/config.properties";
3
4 try (InputStream inputStream =
5     Files.newInputStream(Paths.get(propFileName))){
6     if (inputStream != null) {
7         prop.load(inputStream);
8     } else {
9         throw new FileNotFoundException("...");
10    }
11 } catch (IOException e) {
12     System.out.println("Exception : " + e);
13 }
14 String ip = prop.getProperty("ip", "postgresql");
```

[codes/PropertyExample.java](#)

# Scanner

```
1 public class ScannerExample {  
2     public static void main(String args[]) throws  
3         ↪ IOException {  
4         Scanner in  
5             = new Scanner(Paths.get("/temp/test.txt"));  
6         int lineNumber = 1;  
7         while(in.hasNextLine()){  
8             String line = in.nextLine();  
9             System.out.println("line "  
10                + (lineNumber++)  
11                + " :"  
12                + line);  
13        }  
14    }
```

codes/ScannerExample.java

