

# ***SOMMAIRE***

---

Introduction

Format Propriétaire -Standard

Code Alphanumérique

Entrée Alphanumérique

Format d'Images

Format Audio et Vidéo

Compression de données

Format de Données interne



# ***Format de donnée***

---

Le *format de donnée* est la manière utilisée en informatique pour représenter des données sous forme de nombres binaires

Un format de donnée est une convention utilisée pour représenter un type de donnée

# ***Format propriétaire - Standard***

**Format propriétaire:** Unique à un produit ou une compagnie

**Standard:** Documenté, adoption encouragée partout

*de facto:* Un format propriétaire peut devenir un standard (e.g., Adobe, Postscript, etc.)

*par comité:* Un comité d'expert est constitué pour résoudre un problème et proposer un standard pour un problème particulier

# ***Organisations de standardisation***

---

ISO	International Standards Organization
CSA	Canadian Standards Association
ANSI	American National Standards Institute
IEEE	Institute for Electrical and Electronics Engineers
IETF	Internet Engineering Task Force

# *Exemples de standards*

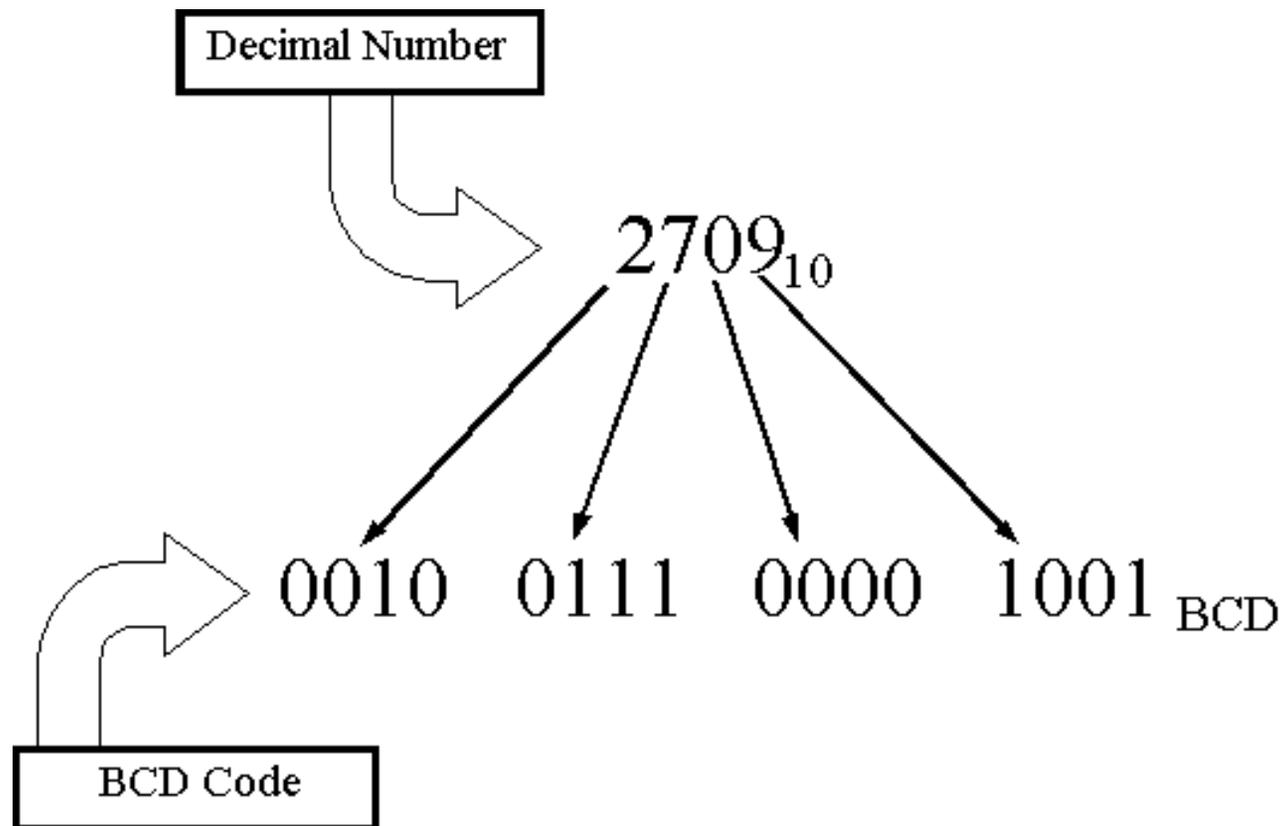
Alphanumerique	Unicode, ASCII, EBCDIC
Image (bitmap)	PNG, JPEG
Image (vectoriel)	SVG, PDF
Fontes	OpenType
Son	WAV, Vorbis, Flac, Speex
Video	Theora, VP8
E-mail	RFC822
Rich text	HTML, OpenDocument

# Caractères Alphanumériques

- Lettres de l'alphabet (minuscule et majuscules)
- Les chiffres : 1, 2, 3, 4, ...
- Ponctuations : !, ?, ", (, ...
- Caractères spéciaux : \*, \$, ¿, ...

Quelques standards utilisés pour les coder en binaires

1. BCD (*Binary Coded Decimal*)
2. EBCDIC (*Extended Binary Coded Decimal Interchange*)
3. ASCII (*American Standard Code for Information Interchange*)
4. Unicode



Développé initialement par le *American National Standards Institute* (ANSI)

Code de 7 bits (128 entrées possibles, 95 graphiques et 33 de contrôle), stocké sur un octet [*byte*]

Le 8<sup>e</sup> bit est quelquefois inutilisé, utilisé comme bit de parité, ou pour coder 128 autres symboles

# Table de Codage ASCII

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	Ⓐ	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	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

# Codage ASCII

G à le code  $47_{16}$  ou  $0100\ 0111_2$

- 95 codes *graphiques* de  $20_{16}$  à  $7E_{16}$ 
  - codes alphabétiques
  - codes numériques
  - codes de ponctuation
- 33 codes de *contrôle* de  $00_{16}$  à  $1F_{16}$  et  $7F$
- Latin-1: variante qui ajoute des caractères accentués et spéciaux

**Exemple:** La chaîne de caractère Hello, world !, à pour code (en hexadécimal),

$48\ 65\ 6C\ 6C\ 6F\ 2C\ 20\ 77\ 6F\ 72\ 6C\ 64\ 21$

# Caractères graphiques

`a` à le code hexadécimal  $61_{16}$ . Pour convertir ce caractère en caractère majuscule (i.e., `A`), on doit soustraire au code  $20_{16}$  (touche *shift*)

L'ordre des lettres est respecté (classement par ordre alphabétique par simple algorithme de trie)

Le caractère `5` codé par le code  $35_{16}$  est différent du nombre  $5$ . Pour convertir le caractère en nombres on doit soustraire au code la valeur  $30_{16}$ .

# Caractères de contrôle

<b>NUL</b>	(Null) No character; used to fill space	<b>DLE</b>	(Data Link Escape) Similar to escape, but used to change meaning of data control characters; used to permit sending of data characters with any bit combination
<b>SOH</b>	(Start of Heading) Indicates start of a header used during transmission	<b>DC1, DC2, DC3, DC4</b>	(Device Controls) Used for the control of devices or special terminal features
<b>STX</b>	(Start of Text) Indicates start of text during transmission	<b>NAK</b>	(Negative Acknowledgment) Opposite of ACK
<b>ETX</b>	(End of Text) Similar to above	<b>SYN</b>	(Synchronous) Used to synchronize a synchronous transmission system
<b>EOT</b>	(End of Transmission)	<b>STB</b>	(End of Transmission Block) Indicates end of a block of transmitted data
<b>ENQ</b>	(Enquiry) A request for response from a remote station; the response is usually an identification	<b>CAN</b>	(Cancel) Cancel previous data
<b>ACK</b>	(Acknowledge) A character sent by a receiving device as an affirmative response to a query by a sender	<b>EM</b>	(End of Medium) Indicates the physical end of a medium such as tape
<b>BEL</b>	(Bell) Rings a bell	<b>SUB</b>	(Substitute) Substitute a character for one sent in error
<b>BS</b>	(Backspace)	<b>ESC</b>	(Escape) Provides extensions to the code by changing the meaning of a specified number of contiguous following characters
<b>HT</b>	(Horizontal Tab)	<b>FS, GS, RS, US</b>	(File, group, record, and united separators) Used in optional way by systems to provide separations within a data set
<b>LF</b>	(Line Feed)	<b>DEL</b>	(Delete) Delete current character
<b>VT</b>	(Vertical Tab)		
<b>FF</b>	(Form Feed) Moves cursor to the starting position of the next page, form, or screen		
<b>CR</b>	(Carriage return)		
<b>SO</b>	(Shift Out) Shift to an alternative character set until SI is encountered		
<b>SI</b>	(Shift In) see above		

# EBCDIC

	0	1	2	3	4	5	6	7
0	NUL	DLE	DS		space	&	-	
1	SOH	DC1	SOS		RSP		/	
2	STX	DC2	FS	SYN				
3	ETX	DC3	WU5	IR				
4	SEL	ENP	BYP/INP	PP				
5	HT	NL	LF	TRN				
6	RNL	BS	ETB	NBS				
7	DEL	POC	ESC	EOT				
8	GE	CAN	SA	SBS				
9	SPS	EM	SFE	IT				
A	RPT	UB5	SM/SW	RFF	g	!		:
B	VT	CU1	CSP	CU3	.	\$	.	#
C	FF	IFS	MFA	DC4	<	*	%	⊗
D	CR	IGS	ENQ	NAK	(	)	~	'
E	SO	IRS	ACK		+	:	>	=
F	SI	IUS	BEL	SUB	:	~	?	*

	8	9	A	B	C	D	E	F
0					(	)	\	0
1	a	j	_		A	J	NSP	1
2	b	k	s		B	K	S	2
3	c	l	t		C	L	T	3
4	d	m	u		D	M	U	4
5	e	n	v		E	N	V	5
6	f	o	w		F	O	W	6
7	g	p	x		G	P	X	7
8	h	q	y		H	Q	Y	8
9	i	r	z		I	R	Z	9
A					SHY			
B								
C								
D								
E								
F								E0

Pas de caractères pourtant très utiles aujourd'hui !  
*comme [ ]* (langage C, C++, java, fortran, etc.), *{ }* (langage C, C++), *~* (Unix, Internet, etc.), etc.

Code 8 bits, inventé par IBM, désuet mais beaucoup d'archives l'utilisent

Code de 32 bits (4 milliards d'entrées possibles mais contient jusqu'à maintenant quelques millions de caractères distincts seulement)

Plusieurs encodages (représentation du code en séquence de bits))

Chaque caractère est stocké sur 1-5 octets

Le code *latin-1* est englobé dans ce code

Code multilingues: lettres et idéogrammes (Amérique, Europe, Afrique, Asie, etc.)

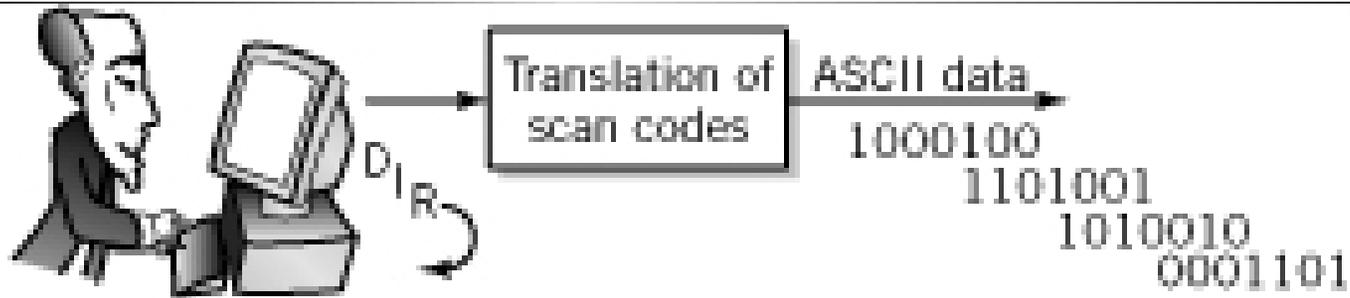
<http://www.unicode.org>

# Table de codage Unicode

## Code range (in hexadecimal)

0000–	} 0000–00FF Latin-1 (ASCII)
1000–	} General character alphabets: Latin, Cyrillic, Greek, Hebrew, Arabic, Thai, etc.
2000–	} Symbols and dingbats: punctuation, math, technical, geometric shapes, etc.
3000–	} 3000–33FF Miscellaneous punctuations, symbols, and phonetics for Chinese, Japanese, and Korean
4000–	} Unassigned
5000–	} Unassigned
•	
•	} 4E00–9FFF Chinese, Japanese, Korean ideographs
•	
A000–	} Unassigned
B000–	} Unassigned
C000–	} A000–D7AF Korean Hangeul syllables
D000–	} Unassigned
E000–	} Space for surrogates
F000–	} E000–F8FF Private use
FFFF –	} Various special characters

# Du Clavier Au Binaire



Le clavier génère un code [*scan code*] lorsque la touche est pressée et un autre lorsque la touche est libérée

L'ordinateur le convertit en ASCII/Unicode par **conversion logiciel**:

- Adapté à différents langages ou claviers
- Multiples combinaisons possibles (shift, control, ...)

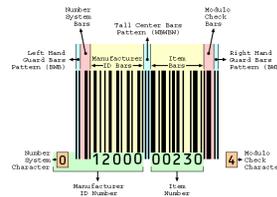
Les caractères sont stockés dans un *buffer*, (comme un *flot de données*)

# Autres sources d'entrées alphanumériques

Un Scanner et un logiciel OCR



Un lecteur de *code barre*



<http://www.digital.net/barcoder/barcode.html>

Un lecteur de bande magnétique



Convertisseur de signal vocal

Appareil de pointage



# Format d'images

## Images bitmap [*raster images*]

PNG, JPEG, ...

Désigne un format de donnée qui va représenter et stocker chaque point de l'image individuellement (niveaux de gris ou niveaux de rouge, vert, bleu)

## Images Vectorielle [*vector images*]

SVG, PDF, PostScript, ...

Désigne un format de donnée où l'image entière est décrite par un ensemble de forme géométrique (lignes, courbes, cercles, ellipse, ...).

**Préoccupations:** qualité de l'image, espace de stockage nécessaire, facilité de manipulation

# Image numérique bitmap



x =	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
y =	210	209	204	202	197	247	143	71	64	80	84	54	54	57	58
42	206	196	203	197	195	210	207	56	63	58	53	53	61	62	51
43	201	207	192	201	198	213	156	69	65	57	55	52	53	60	50
44	216	206	211	193	202	207	208	57	69	60	55	77	49	62	61
45	221	206	211	194	196	197	220	56	63	60	55	46	97	58	106
46	209	214	224	199	194	193	204	173	64	60	59	51	62	56	48
47	204	212	213	208	191	190	191	214	60	62	66	76	51	49	55
48	214	215	215	207	208	180	172	188	69	72	55	49	56	52	56
49	209	205	214	205	204	196	187	196	86	62	66	87	57	60	48
50	208	209	205	203	202	186	174	185	149	71	63	55	55	45	56
51	207	210	211	199	217	194	183	177	209	90	62	64	52	93	52
52	208	205	209	209	197	194	183	187	187	239	58	68	61	51	56
53	204	206	203	209	195	203	188	185	183	221	75	61	58	60	60
54	200	203	199	236	188	197	183	190	183	196	122	63	58	64	66
55	205	210	202	203	199	197	196	181	173	186	105	62	57	64	63

Image numérique en niveaux de gris: *matrice* où chaque élément (pixel) représente l'intensité *discrète* à ce point.

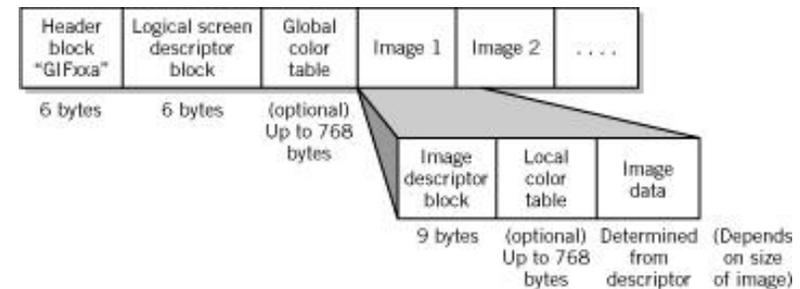
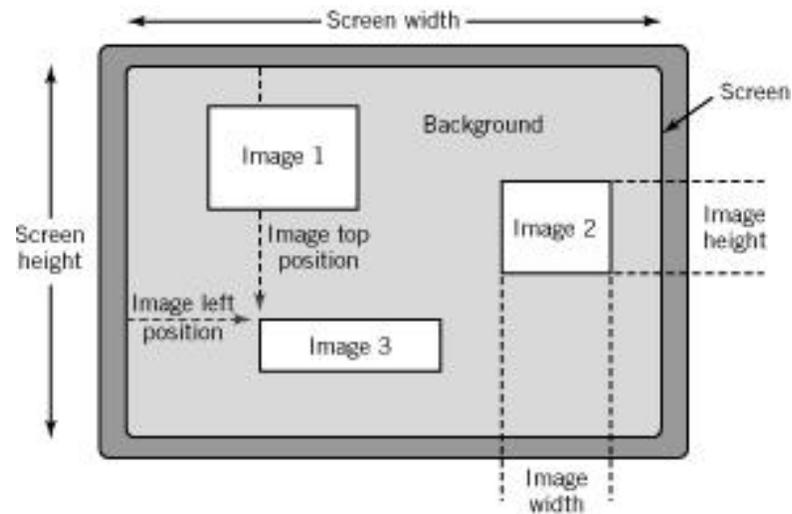
Stockage pour une image de  $128 \times 128$  pixels avec 256 couleurs:

$$128 \times 128 \times \ln 256 = 16KB$$

Pour réduire le stockage:

▷ compression

# Format GIF



Développé par Comuserve (1987)

GIF<sub>89</sub> permet l'animation d'images

nb couleurs : 256

Compression sans perte, algorithme LZW (Lempel, Ziv & Welch)

# Images numériques vectorielles

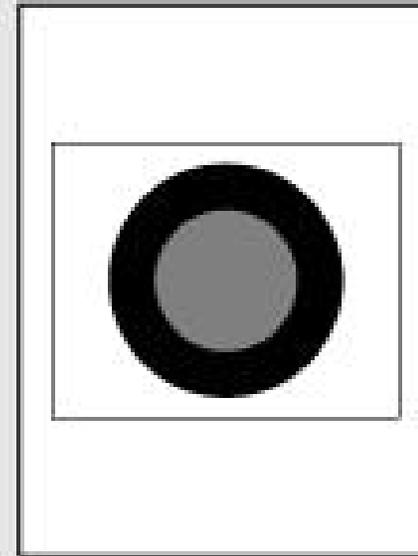
L'image est décomposée en formes géométriques (lignes, courbes, ...),  
i.e. en instructions spécifiant comment dessiner l'image

```
288 396 translate % move origin to center of page
0 0 144 0 360 arc % define 2" radius black circle
fill

0.5 setgray % define 1" radius gray circle
0 0 72 0 360 arc
fill

0 setgray % reset color to black
-216 -180 moveto % start at lower left corner
0 360 rmoveto % and define rectangle
432 0 rmoveto % ...one line at a time
0 -360 rmoveto
closepath % completes rectangle
stroke % draw outline instead of fill

showpage % produce the image
```



# Exemple PostScript

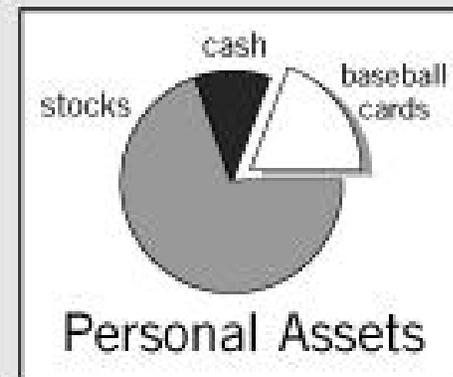
```
% procedure to draw pie slice
%arguments graylevel, start angle, finish angle
/wedge {
  0 0 moveto
  setgray
  /angle1 exch def
  /angle2 exch def
  0 0 144 angle1 angle2 arc
  0 0 lineto
  closepath } def

% add text to drawing
0 setgray
144 144 moveto
(baseball cards) show
-30 200 (cash) show
-216 108 (stocks) show
32 scalefont
(Personal Assets) show

showpage

%set up text font for printing
/Helvetica-Bold findfont
16 scalefont
setfont

.4 72 108 wedge fill % 108-72 = 36 = .1 circle
.8 108 360 wedge fill % 70%
% print wedge in three parts
32 12 translate
0 0 72 wedge fill
gsave
-8 8 translate
1 0 72 wedge fill
0 setgray stroke
grestore
```



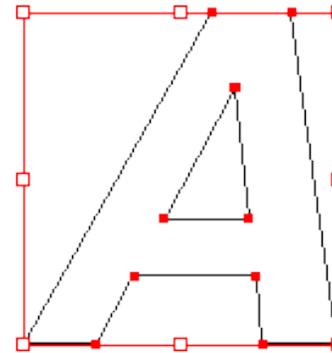
# Images vectorielles

Stockage: dépend de la complexité de l'image

Basé sur des formules mathématiques: l'image peut être facilement tournée, agrandie, sans perte de qualité



Bitmap

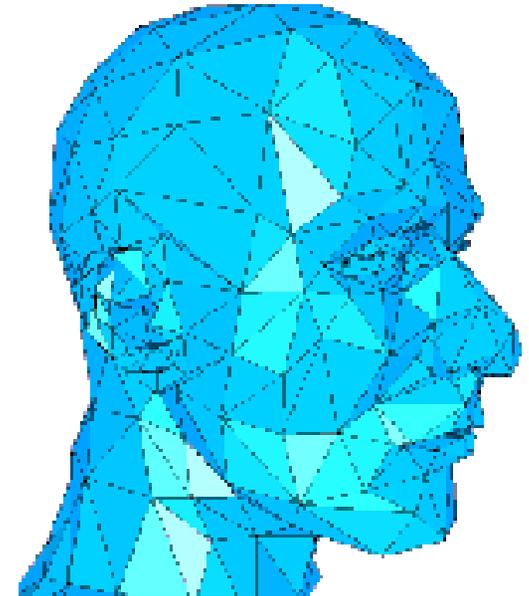
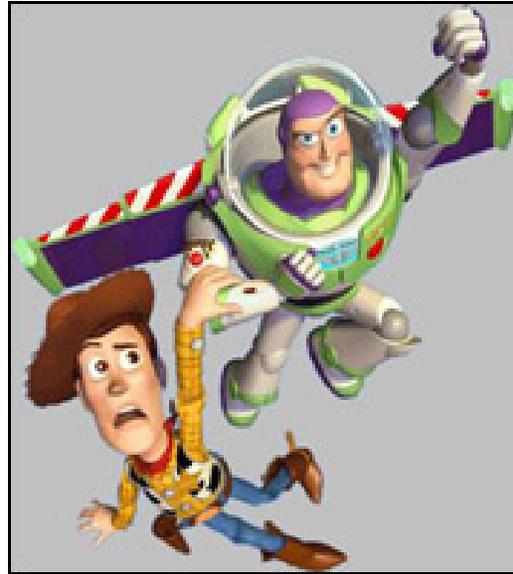


Vectorielle

Nécessite de convertir l'image vectorielle en bitmap avant affichage

# Page Description Language

- Stocké en ASCII ou Unicode
- Convertit par un programme en *bitmap*



# Séquence Vidéo

Demande une grande capacité de stockage

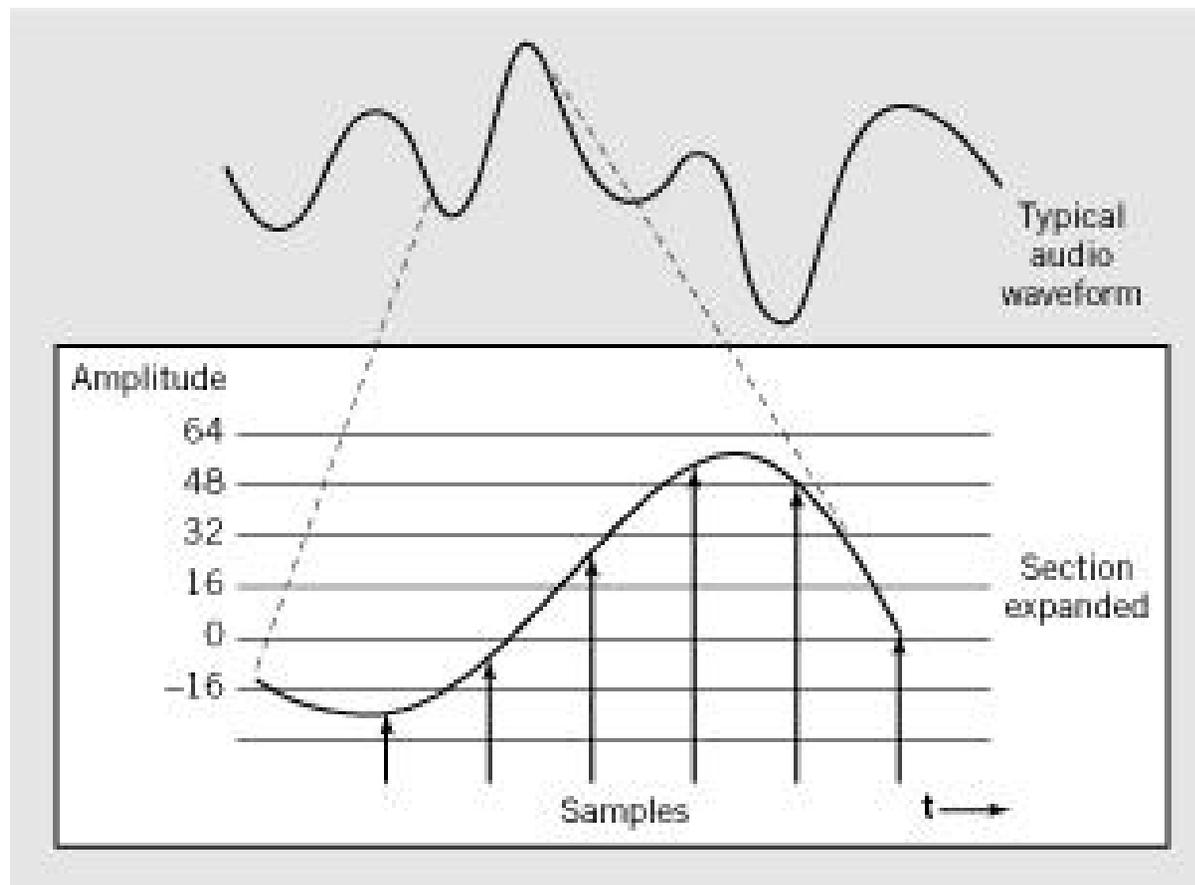
*Exemples* : Caméra vidéo produit des images  $640 \times 480$ , 3 octets par pixel, 30 images par seconde ▷ 27,65 MB/s  
(1 minute ▷ 1.6 GB)

*Streaming Video* Séquence vidéo téléchargée en temps réel (e.g.,  
video-conférence)

Compression possible (exemples : *Theora*, *VP8*)

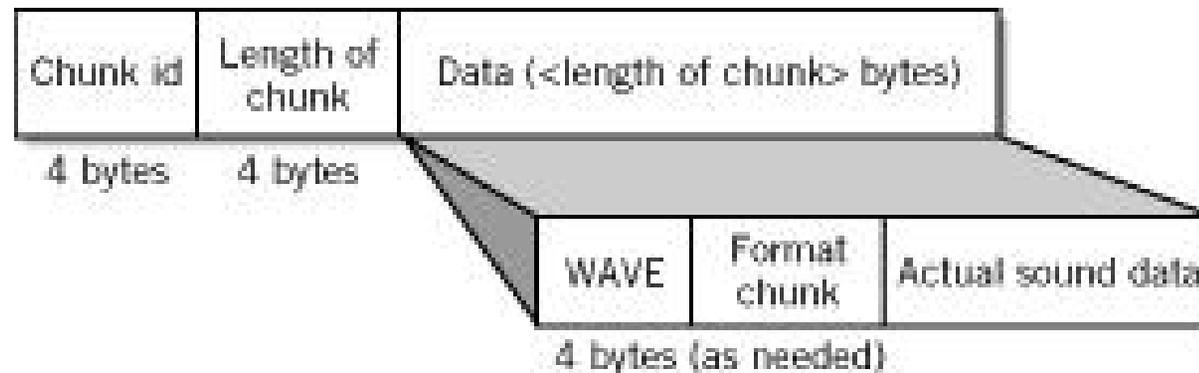
# Données Audio

Signal analogique digitalisé par un convertisseur A/D



# Format WAV

Inventé par Microsoft. Échantillon de son sur 8, 16 bits à une fréquence d'échantillonnage de 11.025 KHz, 22.05 KHz, 44.1 KHz en mono ou stéréo ( $2 \times 16$  bits)



MIDI : *Musical Instrument Digital Interface*

Utilisé par les compositeurs musiciens, les professionnels du son et de l'acoustique

Instructions permettant de recréer et synthétiser des nouveaux sons et d'interfacer avec des synthétiseurs (mais ne permet pas de recréer efficacement de la voix humaine)

3 minutes de son  $\approx$  10 KB

# Format MP3

Dérive du format MPEG-2 (*Moving Picture Expert Group*)

Compression avec perte

3 minutes de musique  $\approx$  2 MB

# *Compression des Données*

Recoder les données de telle façon qu'elles nécessitent moins d'octets pour le stockage

- Réduction du coût de stockage
- Transmission rapide des données

Compression avec (e.g., Ogg, JPEG, Theora, ...) ou sans perte (l'algorithme inverse restaure les données dans leur forme originale sans altération) (ex: PNG, GZip, ...)

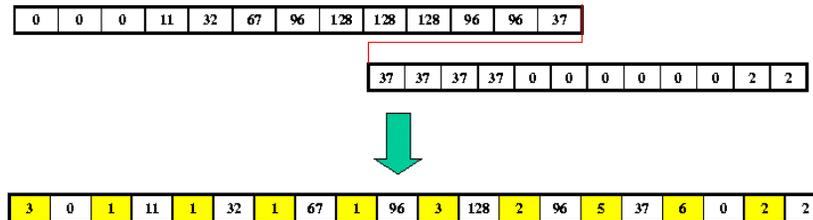
# Taux de compression

$$C = \frac{\text{Nb. de bits après compression}}{\text{Nb. de bits avant compression}}$$

Ex: Compression avec un facteur de 10 : 1

Méthodes simples de compression: RLE, dictionnaire.

# Compression RLE



RLE: Run Length Encoding

Création d'une nouvelle séquence dans laquelle le deuxième élément correspond au niveau de gris et le premier élément correspond au nombre de pixels consécutifs possédant ce niveaux de gris

Codage séparé du niveaux de gris et de l'occurrence de chaque pixel



Fort taux de compression pour des images possédant de nb. zones de régions homogènes

# Compression avec dictionnaire

“Peter Piper picked a peck of pickled peppers”

[Pe] t [er] [Pi] p [er] [pi] [ck] [ed] a [pe] [ck] of

[pi] [ck] l [ed] [pe] pp [er]s

En utilisant le dictionnaire suivant

[Pe:▲] [pi:▼] [ed : ◆] [er: ★] [ck : ►] [ck : ✚] [pe: ✓] [Pi : □]

Et on transmet le dictionnaire et la phrase

▲t★ □p★ ▼✚◆ a ✓ ► of ▼✚l◆ ✓pp★s

# ***Format de données interne***

---

Les données sont stockées sous forme binaire de taille différentes

Ces données peuvent être interprétées pour représenter des données de différents type et format *via* un programme

Float, char, boolean, int, ...

# Exemple: Programme en Langage Fortran

```
//VARIABLES USED
key: CHARACTER;
number: INTEGER;
error, stop: BOOLEAN;

(
  stop = false;
  error = false;
  ReadAKey;
  WHILE NOT stop AND NOT error (
    number = 10 * number + (ASCIIVALUE(key)- 48);
    ReadAKey;
  );
  IF error
    PRINTOUT ('Illegal Character in Input')
  ELSE PRINTOUT ('Input number is ', number);
);

PROCEDURE ReadAKey;{
  READ (key);
  IF (ASCIIVALUE(key)=13 or ASCIIVALUE(key)=32 or
    ... ASCIIVALUE(key)=44)
    stop = TRUE;
  ELSE IF (key < '0') or (key > '9')
    error = TRUE;
};
```