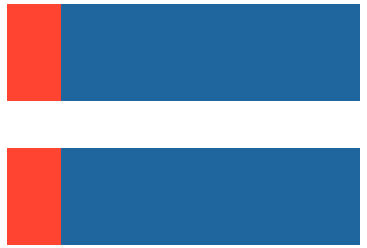
Codage et stockage de l'information



Yannick Prié
UFR Informatique
Université Claude Bernard Lyon 1

2007-2008 – Master SIB
M1 – UE 3 / Bloc 4 – Cours 1

Bloc 4 : architecture et fonctionnement des systèmes d'information documentaires

- Yannick Prié / Geneviève Lallich
- Objectif : « Etre à même de comprendre le fonctionnement d'un système d'information documentaire réparti. Découvrir un certain nombre de systèmes documentaires. Plus précisément, être capable de comprendre l'architecture des systèmes d'information (machines impliquées, programmes variés, circulation des informations, *etc*.). Une attention particulière sera portée au fonctionnement des outils de recherche d'information. »

Bloc 4 : architecture et fonctionnement des systèmes d'information documentaires

- Codage et stockage de l'information (1 CM) - YP
- Informatique de la recherche d'informations (2 CM) - GLB
- Architecture logicielle des ordinateurs (1 CM) - YP
- Architecture client-serveur (1 CM) - YP
- Systèmes d'information répartis (2 CM) - YP

Bloc 4 : architecture et fonctionnement des systèmes d'information documentaires

- TD 4.1 – Codage de l'information (YP)
- TD 4.2 – Cindoc 1/2 (GLB)
- TD 4.3 – Cindoc 2/2 (GLB)
- TD 4.4 – EndNote 1/3 (FCA)
- TD 4.5 – EndNote 2/3 (FCA)
- TD 4.6 – EndNote 3/3 (FCA)
- TD 4.7 – Alexandrie 1/2 (GLB)
- TD 4.8 – Alexandrie 2/2 (GLB)
- TD 4.9 – Analyse de documentation de logiciel (GLB)

CM1 : Codage et stockage de l'information

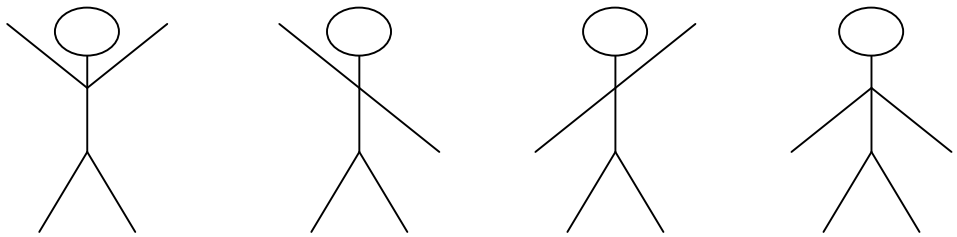
- Objectifs du cours
  - Codage de l'information en numérique
  - Passage d'un codage à un autre
  - Fichiers et types de fichiers
  - Stockage de l'information : différents supports

Codage binaire

- Unité d'information : le bit (*binary digit*)
  - 0/1, ouvert/fermé, allumé/éteint, présent/absent, ...
- Avec un bit
  - compter de 0 à 1, exprimer un choix parmi deux possibilités
- Remarque
  - un bit n'exprime quelque chose que si on sait l'interpréter pour en faire quelque chose (comprendre, agir, manipuler, etc.)
- Plusieurs bits
  - peuvent être associés/combinés
  - pour exprimer des choses plus complexes
- Codage binaire
  - inscrire de l'information dans une suite de bit
  - suivant un code que celui qui écrit et celui qui lit doivent partager

Codages binaires sans ordinateur

- Morse
  - symboles : point/trait, court/long
  - codage morse : 3 symboles = 1 lettre
  - - . . - - - . . - - . . . . - . - - - .
- Randonneurs à hélicoptère
  - symbole : bras levé, bras baissé
  - codage : 2 bras = « A l'aide » ou « Tout va bien »



Coder des entiers dans des suites de bits

- Compter en base 2
- Deux symboles : « 0 », « 1 »

0 → 0	1 → 1	2 → 10	3 → 11
4 → 100	5 → 101	6 → 110	7 → 111
8 → 1000	9 → 1001	10 →	11 →
12 →	13 →	14 →	15 →
16 →	17 →	18 →	19 →
20 →	21 →	22 →	

Entier = somme de puissances de 2

1	00 00 00 00	00 00 00 01
2	00 00 00 00	00 00 00 10
4	00 00 00 00	00 00 01 00
8	00 00 00 00	00 00 10 00
16	00 00 00 00	00 01 00 00
32	00 00 00 00	00 10 00 00
64	00 00 00 00	01 00 00 00
128	00 00 00 00	10 00 00 00
256	00 00 00 01	00 00 00 00
512	00 00 00 10	00 00 00 00
1024	00 00 01 00	00 00 00 00
1143	00 00 01 00	01 11 01 11

$1143 = 2^E10 + 2^E6 + 2^E5 + 2^E4 + 2^E2 + 2^E1 + 2^E0$

Jusqu'où peut-on compter

- 8 bits → $2^E8 = 256$
- 16 bits → $2^E16 = 65536$
- 32 bits → $2^E32 = 18\ 446\ 744\ 073\ 709\ 551\ 616$
- 64 bits → 2^E64 = beaucoup
- Remarque
  - les microprocesseurs des ordinateurs manipulent des nombres de 32 ou 64 bits

Octets, kilo-octets, méga-octets, giga-octets...

- 1 octet = 1 byte (en) = 8 bits
- 1 ko = 1024 octets = 2^E10 octets
- 1 Mo = 1024 ko = 2^E20 octets
- 1 Go = 1024 Mo = 2^E30 octets
- 1 To = 1024 Go = 2^E40 octets
- ...

- Remarque
  - À ce niveau là, on ne code plus des entiers, mais de très longues séquences d'entiers

Et pour des entiers négatifs ?

- Une idée :
  - utiliser un bit pour coder le signe
  - utiliser les autres bits pour coder des entiers positifs
- Exercice
  - soient les octets suivants qui représentent des entiers signés
    - 10010001 = -17
    - 10000001 = -1
    - 00010101 = 21
  - Questions
    - quel bit code le signe ?
    - comment représenter -64, 123, -135 ?

Et pour des réels ?

- Idée
  - un réel = -1 ou 1* mantisse * une puissance de 10
    - 1234,5633 = 1 * 0,12345633 * 10^E4
    - -0,000564 = -1 * 0,564 * 10^E-3
    - *etc.*
  - par exemple si on dispose de 32 bits :
    - 1 bit de signe de la mantisse
    - 8 bits pour l'exposant
    - 23 bits de mantisse
- Exemple : -0,000564
  - signe exposant mantisse
  - 1 1000 0011 00000000000001000110100
- Remarque
  - codage inexact !
    - pi # 0,314150 * 10^E1
  - calculs inexacts !

Codage hexadécimal

- Base 16
- Symboles = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

0 → 0	1 → 1	2 → 2	3 → 3
8 → 8	9 → 9	10 →A	11 →B
12 →C	13 →D	14 →E	15 →F
16 →10	17 →11	18 →12	19 →13
24 →18	25 →19	26 →1A	27 →1B

- Remarque
  - on représente un octet avec 2 symboles
    → plus facile à écrire pour un humain

Codage de l'information

- On peut coder n'importe quoi avec des bits (des octets), en particulier des nombres
- Idée : tout représenter avec des nombres
  - Lettres → nombres
  - Couleurs → nombres
  - Texte simple → lettres → nombres
  - Images → couleurs → nombres
  - Sons → échantillons → nombres
  - Vidéo → images + sons → nombres
  - Texte mis en page → texte simple + instructions de mise en page
    → texte simple + texte instructions → nombres
  - Etc.

Codage des caractères

- Caractères alphanumériques
  - 10 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8 , 9
  - 26 lettres minuscules
  - 26 lettres majuscules
  - caractères de ponctuation (virgule, espace,...) ,
    caractères particuliers ($,£,...), caractères accentués
  - caractères "cachés" : bip (<bell>), changement de ligne (new line ou line feed : <lf>), retour en début de ligne (carriage return : <cr>), fin de fichier (end of file : <eof>),...
- Idée
  - un caractère se code comme un nombre, lui-même codé sur *n* octets
  - Donc un texte de *m* caractères est codé pas m nombre,
    codés sur *m* * *n* octets

Codage ASCII

- American Standard Code for Information Interchange
- ASCII base : codage sur 1 octet, et sur 7 bits
  - de 0 à 31
    → caractères de contrôle
  - de 32 (espace) à 126 (‘~’)
    → caractères « alpha-numériques » non-accentués.
- Au dessus de 127, extension de l’ASCII original
  - caractères accentués : À, Â, ë, ï, …
  - (voir la suite)

32		64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	□

Codage des caractères en ASCII

- Pour coder des caractères
  - on code le caractère en ASCII
  - on transforme le code ASCII en code binaire

caractère	ascii	décomposition de l'ascii	binaire
E	69	64 + 4 + 1	01 00 01 01
x	120	64 + 32 + 16 + 8	01 11 10 00
e	101	64 + 32 + 4 + 1	01 10 01 01
m	109	64 + 32 + 8 + 4 + 1	01 10 11 01
p	112	64 + 32 + 16	01 11 00 00
l	108	64 + 32 + 8 + 4	01 10 11 00
e	101	64 + 32 + 4 + 1	01 10 01 01
		2^6 2^5 2^4 2^3 2^2 2^1 2^0	

Codage des caractères en ISO 8859-1

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	*inutilisés*															
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8x	*inutilisés*															
9x																
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Remarques ISO 8859-1

- ISO 8859 permet de gérer les langues
  - albanais, allemand, anglais, basque, catalan, danois, gaélique écossais, espagnol, féringien, finnois, islandais, gaélique irlandais, italien, néerlandais, norvégien, portugais, romanche, suédois + afrikaans et swahili
  - utilisé donc en Europe de l'ouest, Amérique, Australie et dans une grande partie de l'Afrique.
  - manque pour le français : œ, Ÿ. Manque aussi : €
- Il existe donc des codages dérivés : utilisation des zones inutilisées
  - pour des caractères de contrôle
  - pour des caractères spéciaux. Exemple *Windows 1252*

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
8x	€		‚	*ƒ*	„	…	†	‡	ˆ	‰	Š	‹	Œ		Ž	
9x		‘	’	“	”	•	–	—	˜	™	š	›	œ		ž	Ÿ

Attention : *Windows 1252* → *Macintosh Roman* : Œ → å, å → Â, ç → Á, è →Ë, é → È !

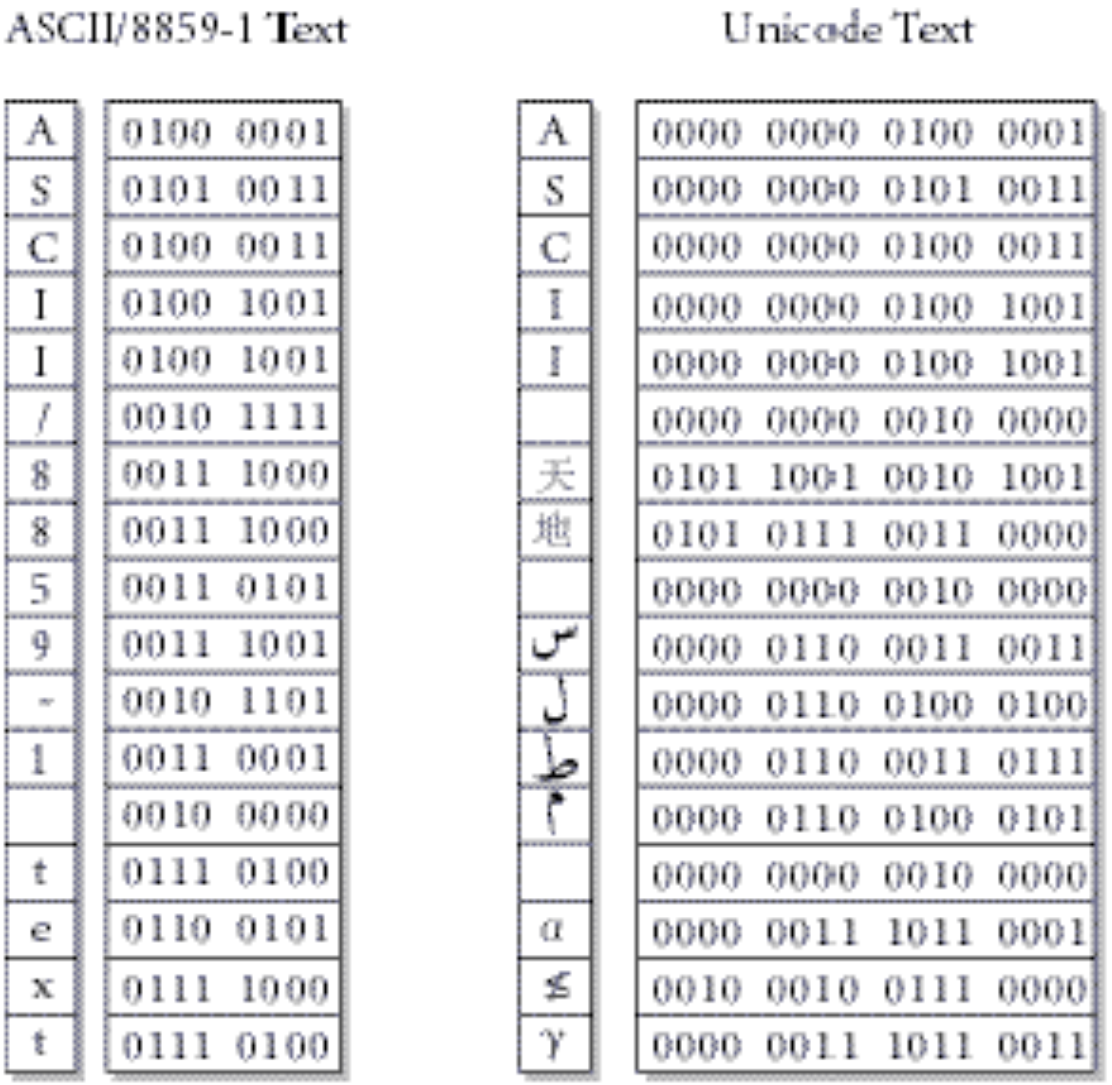
Unicode

- Objectif : mettre fin au chaos
- « Unicode spécifie un numéro unique pour chaque caractère, quelle que soit la plate-forme, quel que soit le logiciel et quelle que soit la langue » → tout caractère à un nom et correspond à un nombre
- Pour l'instant, à peu près 96,447 caractères pour toutes les langues du monde, les mathématiques, les symboles, etc.
- Plusieurs bases
  - BMP : basic multilingual plane, >64000 nombres (2^E16), 6300 non utilisé
  - autres : 870000 non utilisés
  - → il reste de quoi étendre !

Unicode

- Notion de séquences de caractères
  - caractère + marques.
  - A + ^ → Â
  - remarque : pour les caractères latins, il existe déjà des caractères précomposés, conservés pour la compatibilité
- Chaque caractère
  - *code point* unique
  - Forme hexadécimale après le préfixe "U".
  - Exemple : U+0041 → #0041 = 65 en décimal, représente le caractère « A » dans Unicode.
- Pour une langue
  - *code script* (ensemble de code points)

Exemples unicode



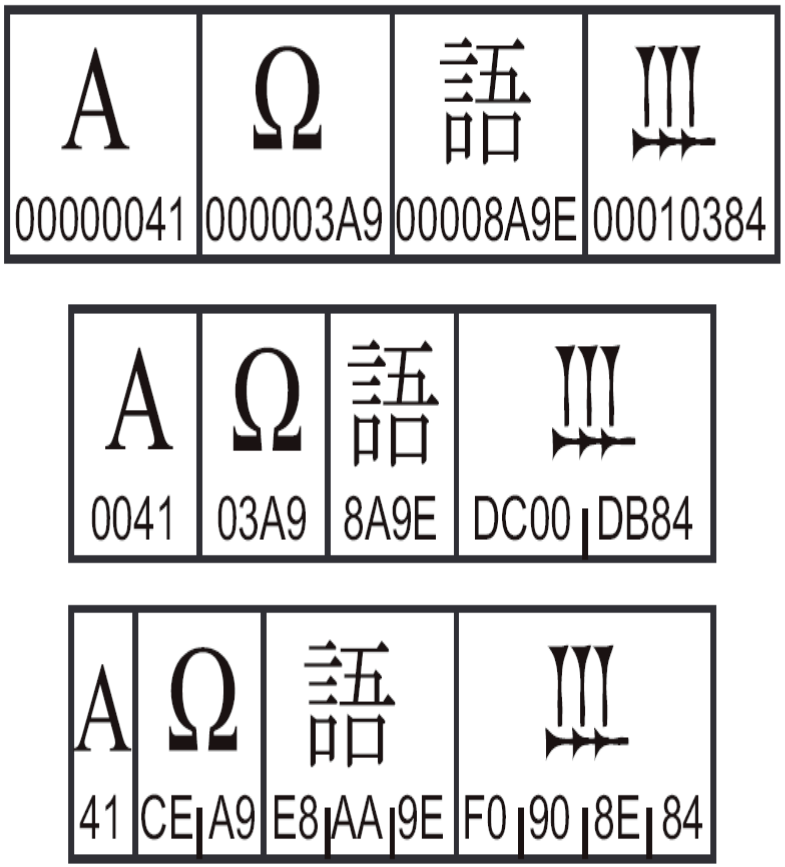
ASCII/8859-1 Text

A	0100 0001
S	0101 0011
C	0100 0011
I	0100 1001
I	0100 1001
/	0010 1111
8	0011 1000
8	0011 1000
5	0011 0101
9	0011 1001
-	0010 1101
1	0011 0001
	0010 0000
t	0111 0100
e	0110 0101
x	0111 1000
t	0111 0100

Unicode Text

A	0000 0000 0100 0001
S	0000 0000 0101 0011
C	0000 0000 0100 0011
I	0000 0000 0100 1001
I	0000 0000 0100 1001
	0000 0000 0010 0000
天	0101 1001 0010 1001
地	0101 0111 0011 0000
	0000 0000 0010 0000
س	0000 0110 0011 0011
ل	0000 0110 0100 0100
ط	0000 0110 0011 0111
م	0000 0110 0100 0101
	0000 0000 0010 0000
α	0000 0011 1011 0001
≤	0010 0010 0111 0000
γ	0000 0011 1011 0011

Encodage unicode



UTF-32

Codage sur 32 bits :
traduction directe du code point

UTF-16

Codage sur 16 bits :
traduction directe du code point
pour beaucoup de caractères (BMP)
Codage complexe éventuel sur
32 pour les autre

UTF-8

Codage sur 8 bits :
traduction directe du code point
pour les caractères de base latin
(127 premiers ISO 88596-1).
Codage complexe sur
16, 24 ou 32 bits pour les autres

Où en est-on ?

- ASCII
  - encore utilisé par pas mal de programme (exemple : C)
  - a permis aux ordinateurs d'échanger des informations → réseaux
  - beaucoup de problèmes avec des applications grands public
- ISO 8859-1 :
  - pas très bon, mais très diffusé (windows, unix, mac)
  - nombreuses applications installées
- Unicode
  - UTF-8 utilisé (web), car compatibilité descendante avec ISO 8859-x.
    - Français : certains caractères sur 16 bits... → « PÃ©nÃ©lope »
  - Tout parser XML *doit* gérer UTF8 et UTF16
  - Vers un passage à UTF-32 généralisé ?

Remarques sur les codages

- Il existe une infinité de codage pour les nombres, les caractères, etc.
  - → tout l'enjeu est de se mettre d'accord
- Une fois qu'on sait coder des caractères
  - la langue écrite est un codage qui utilise les caractères, non réductible à un code
  - on peut faire des codes en utilisant des caractères
    - plus ou moins lisibles par l'homme
    - pour
      - donner des ordres aux machines : programmes
      - décrire des documents : HTML, RTF
      - décrire n'importe quoi : SGML, XML
      - ...
    - ces codes/manières de coder seront appelés des *langages*

Codage des couleurs

- Objectif
  - passer un ensemble de couleurs dans un espace de couleur mathématique, donc codable comme des nombres
- Plusieurs façons de faire
  - RGB
    - rouge, vert, bleu
    - synthèse additive
  - HSL
    - *Hue (teinte), Saturation (intensité), Luminance*
  - YUV
    - Télévision
    - Luminance + chrominance
  - CIE
    - luminance (0/100) + deux gammes de couleur (-120/+120)
    - standard industriel
  - CMY
    - Cyan, Magenta, Jaune → synthèse soustractive
  - ...

RGB

- Correspond à la manière dont les tubes cathodiques rendent les couleurs
  - Synthèse additive
  - Trois points = un pixel (picture element)
- Codage informatique
  - Une intensité pour chaque couleur
    → un nombre pour chaque couleur
- De façon classique
  - échelle entre 0 et 255 (8 bits = 1 octet)
  - 3 octets pour une couleur
- Exemples
  - Couleurs pures : Vert (0,255,0), Rouge (255,0,0), Jaune (255,255,0), Blanc (255,255,255)
  - Un autre vert (128,188,88)
  - Du noir au blanc : (x,x,x)

Codage des couleurs en hexadécimal

- Exercice pratique
  - Combien peut-on coder de couleurs différentes avec 3 octets ?
- Question de lisibilité
  - (0,255,255) = 000000001111111111111111 = #00FFFF
  - (128,188,88) = 100000001011110010110000 = #80BC58

Codage des images

- Une image est un tableau de pixels (h x l)
  → coder chaque pixel par une couleur, ligne par ligne
- Poids d'une image = taille occupée en mémoire
  - image RGB, 3 octets par couleur, 100 x 100
  - 100 x 100 x 3 = 30000 o = 29,3 ko
- Autres exemples

Définition de l'image	Noir et blanc (1 bit)	256 couleurs (8 bits)	65000 couleurs (16 bits)	True color (24 bits)
320x200	7.8 Ko	62.5 Ko	125 Ko	187.5 Ko
640x480	37.5 Ko	300 Ko	600 Ko	900 Ko
800x600	58.6 Ko	468.7 Ko	937.5 Ko	1.4 Mo
1024x768	96 Ko	768 Ko	1.5 Mo	2.3 Mo

Notion de fichier

- Suite d'octet sur un disque, codant un ensemble d'informations
  - Texte, image, vidéo, données météo, etc.
- A laquelle sont associées des propriétés
  - Nom
  - Taille
  - Date de création
  - Type de fichier
    - → codage utilisé pour inscrire les informations
    - → détermine la ou les applications qui pourront utiliser le fichier
  - ...

Anatomie d'un fichier BMP (bitmap)

En-tête du fichier	Signature (2 octets), indiquant qu'il s'agit d'un fichier BMP à l'aide des deux caractères (BM → BitMap Windows) Taille totale du fichier en octets (4 octets) Champ réservé (4 octets) Offset de l'image (sur 4 octets)
En-tête de l'image	Taille de l'entête de l'image (4 octets). Largeur de l'image (4 octets), Hauteur de l'image (4 octets) Nombre de plans (2 octets, vaut toujours 1) Profondeur de codage de la couleur(2 octets), nombre de bits utilisés pour coder la couleur : 1, 4, 8, 16, 24 ou 32 Méthode de compression (4 octets). 0 si pas compressé Taille totale de l'image en octets (sur 4 octets). Résolution horizontale (4 octets), nombre de pixels / mètre Résolution verticale (4 octets), nombre de pixels / mètre Le nombre de couleurs de la palette (4 octets) Le nombre de couleurs importantes de la palette (4 octets).

Anatomie d'un fichier BMP (suite)

Palette (opt.)	4 octets pour chacune des entrées représentant : composante bleue (un octet), composante verte (un octet), composante rouge (un octet), champ réservé (un octet)
Corps de l'image	Se fait en écrivant successivement les bits correspondant à chaque pixel, ligne par ligne en commençant par le pixel en bas à gauche. Chaque ligne de l'image doit comporter un nombre total d'octets qui soit un multiple de 4; si ce n'est pas le cas, la ligne doit être complétée par des 0 de telle manière à respecter ce critère.

Question : poids exact d'une image bitmap de 120 x 100 pixels, en 24 bpp, sans palette de couleur ?

Fichiers : compression « généraliste »

- Objectif
  - Occuper moins de place sur le disque
- Principe
  - Recoder l'information binaire pour en exploiter la redondance
- Exemple
  - 1000000000000000 (17 caractères) = 1 suivi de 16 caractères 0
  - peut se coder comme 1.10000.0 (9 caractères)
  - gain = 8 caractères
- Nécessiter de décoder = décompresser pour accéder aux informations originelles
- Différentes méthodes
  - Exemple : Lempel-Ziv → code n'importe quel suite de bits
  - ZIP, RAR, *etc*.

Fichiers : compression d'images

- Tirer partie des propriétés de redondance des images
  - Algorithme de compression généraux (sans perte)
  - Ex. GIF
- Tirer partie des propriétés humaines (enlever des détails non distingables)
  - Algorithme de compression spécialisés (avec perte)
  - Ex. JPEG
- Remarque : coder/compression des vidéos
  - Coder une suite d'images (24 / seconde) → énorme
  - Tirer partie de la redondance entre les images qui se suivent
    - Ex. MPEG

Fichiers : archivage

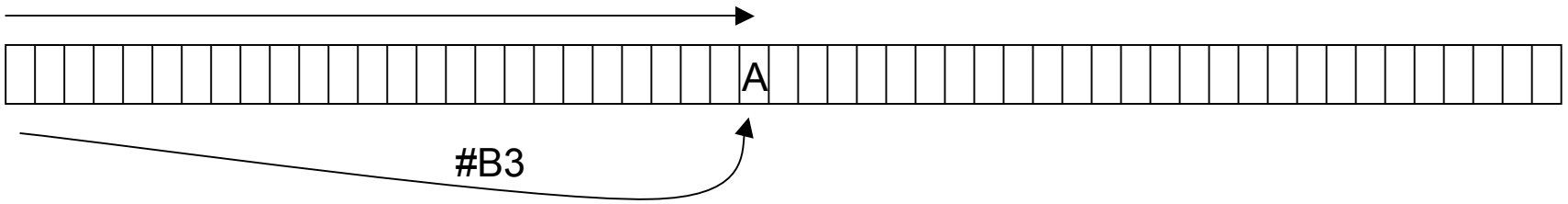
- Compresser (général) puis concaténer des fichiers dans un fichier unique

Fichiers : types MIME

- Pour l'échange de fichiers sur le mail et le web
  - MIME = Multipurpose Internet Mail Extensions
  - Forme
    - Content-type: type_principal/sous_type
  - Exemples
    - Content-type: image/jpeg
    - Content-type: application/pdf

Fichiers : quelques remarques

- Accès à un fichier
  - accès séquentiel : lire toutes les informations une par une
  - *vs* accès indexé : aller directement lire une information à une certaine adresse. Plus rapide, nécessité d'avoir un index.



- Fichiers ASCII (texte) → lecture octet par octet en ASCII
  - Ex : 12345 → « 12345 » (5 octets, lus un par un)
- Fichiers binaire → lecture par n octets suivant le type
  - Ex : 12345 → 00000000 00000000 00110000 00111001 (on lit et écrit directement un entier codé sur 4 octets)

Stocker de l'information numérique

- Ici stockage permanent vs stockage temporaire (mémoire vive)
- Différents types de supports
- Critères
  - Capacité = nombre d'octets
  - Vitesse d'écriture / lecture
  - Durée de vie
  - Transportabilité (rapport capacité / volume / masse)
  - ...
- Remarque
  - on peut stocker de l'information binaire dans la pierre

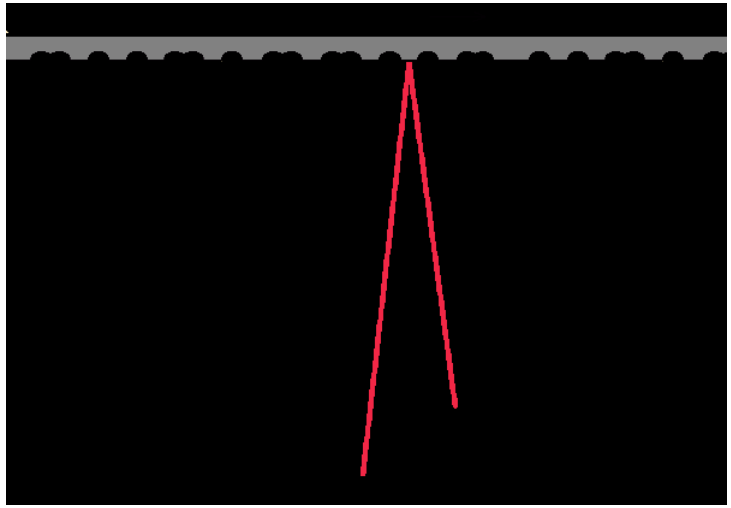
Stockage : supports magnétiques

- Disquettes
  - plastique + oxyde magnétique
  - 3 pouces ½ , 5 pouces ¼
  - petites capacités
- Disques durs
  - plateaux métalliques oxydés, têtes de lecture/écriture magnétique
  - SCSI, IDE,
  - 120 Go - 1To
- Disques amovibles
  - Jaz/Zip
  - 100-200 Mo
- Bandes
  - Systèmes de sauvegardes
  - DAT
  - Tera-octets

(http://tboivin.free.fr)

Stockage : supports optiques

- Cédérom
  - Plastique + film métallique,
  - Gravage au laser (trou/pas trou)
  - 640 Mo / 700 Mo
  - CD-R, CD-RW, etc.
- DVD (Digital Video/Versatile Disk)
  - double couche : couche or + couche argent
  - 4,7 Go
- En début de commercialisation
  - DVD Blue ray
  - HD DVD



(http://tboivin.free.fr)

Stockage : mémoires flash

- Mémoire électroniques non volatiles, réinscriptibles
  - CompactFlash, Memory Stick, SD
  - "clés" USB, etc.
- 32 Mo → 1 Go, 4 Go, 16 Go

Conclusion

- Codage de l'information
  - binaire
  - entiers
  - caractères
  - images
- Supports de stockage
- Suite du cours (Y. Prié) en janvier
  - systèmes d'exploitation
  - réseaux
  - systèmes d'information distribués
- En attendant
  - Représentation des données et des documents (XML)