Outils Informatique
Codage

E. Jeandel

Représentation des données
- Comment coder une image en un fichier ?
- Comment coder un texte en un fichier ?
- Comment représenter une couleur dans un ordinateur ?
- Comment représenter un graphe dans un ordinateur ?
- Comment représenter une base de données dans un ordinateur ?

Dans un ordinateur
- La notion de base est le *bit*.
- Un bit peut prendre deux valeurs, 0 ou 1.
- Les bits sont regroupés, pour simplifier, par 8, pour former ce qu'on appelle un *octet*.

Représenter des données, c'est donc les représenter comme une série de bits, ou comme une série d'octets.

Représentation des nombres

Un nombre entier est représenté par son écriture en base 2 :

51	110011
51	00110011
1664	11010000000

Plus d'informations dans le cours d'Architecture en L3

Représentation des dates

Plusieurs formats :
- Format utilisé (entre autres) sous DOS et Windows :
  - Date sur 16 bits : 5 pour le jour, 4 pour le mois, et 7 pour l'année, en prenant comme référence 1980

$$2010/01/19 = 0011110 \cdot 0001 \cdot 10011 = 0x3c33$$

  Bug de l'an… ?
  - Heure sur 16 bits : 5 pour l'heure, 6 pour les minutes, 5 pour les secondes. Problème ?

$$9:51:36 = 01001 \cdot 110011 \cdot 10010 = 0x4e72$$

- Format utilisé sous Unix : nombre de secondes écoulées depuis minuit UTC (temps universel coordonnée) le 1er janvier 1970, codé sur 32 bits, dont un bit pour le signe.

$$2010/01/19 \text{ à } 9:51:36 = 1263891096$$

Bug de l'an… ? (il y a $31557600 \sim 15 \times 2^{21}$ secondes dans un an)

Représentation des caractères

Un caractère est représenté par 8 bits (donc par un nombre entre 0 et 255). Des tables de correspondance expliquent comment on passe des 8 bits au caractère correspondant.

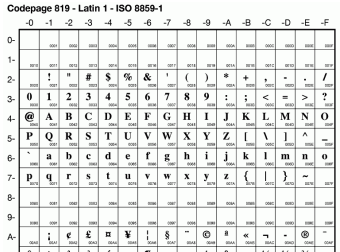
Exemple pour le caractère "é" :

Norme	Code binaire	Décimal
ISO/IEC 8859-1 (Latin-1 Western European)	11101001	233
Mac OS Roman	10001110	142
CP437	10000010	130

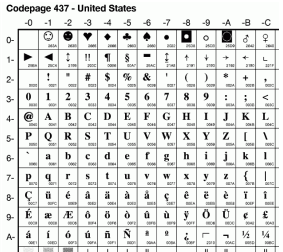
99% des normes ont les mêmes 128 premiers caractères, correspondant à la norme ASCII.

Chacune des normes ne permet de représenter que 256 caractères : insuffisant pour certaines langues.

Codepage 819 - Latin 1 - ISO 8859-1

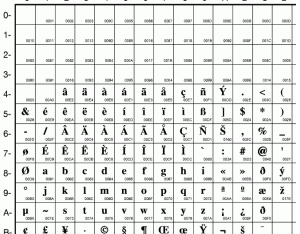


Codepage 437 - United States



Codepage 1275 - Apple Latin 1

Codepage 924 - Latin 9 - EBCDIC



Représentation des caractères : Unicode

Unicode est un standard qui explique comment représenter et manipuler du texte. Il contient en particulier une liste de plus de 100000 caractères.

On trouve ensuite plusieurs façons de les représenter :
- UTF-32 : Représente chaque caractère sur 32 bits (donc 4 octets)
- UTF-8 : Représente la majorité des caractères fréquents sur 8 bits, d'autres sur 16 bits, 24 ou 32 bits

Caractère	Code UTF8
a	01100001
é	11000011 10101001
€	11100010 10000010 10101100
𝟙	11110000 10011101 10011111 10011001

Table utilisée en cours et en TD

␣	00000	h	01000	p	10000	x	11000
a	00001	i	01001	q	10001	y	11001
b	00010	j	01010	r	10010	z	11010
c	00011	k	01011	s	10011	.	11011
d	00100	l	01100	t	10100	,	11100
e	00101	m	01101	u	10101	'	11101
f	00110	n	01110	v	10110	!	11110
g	00111	o	01111	w	10111	?	11111

ne permet de représenter que des minuscules et quelques signes de ponctuation, mais bien suffisant pour les exercices.

Récapitulatif

La séquence suivante :

11110000 10011101 10011111 10011001

peut donc représenter :
- Les 4 nombres 240, 157, 159, 153
- Les 2 nombres 61597 et 40857
- Le nombre 4036861849
- Les 4 caractères ≡¥ƒÖ (en CP437)
- Les 4 caractères üüö (en Mac OS Roman)
- Le caractère 𝟙 (en UTF-8)
- Les instructions assembleur x86 suivantes : `LOCK POPF; LAHF; CDQ`

Un exemple

Un logiciel d'archivage permet de regrouper en un seul fichier plusieurs fichiers et répertoires afin, par exemple, de les stocker ensuite plus facilement, ou de les compresser.

On va ici archiver deux fichiers :

toto.py (16 caractères)
print "bonjour"

readme.txt (7 caractères)
blabla

avec deux logiciels différents

Un exemple : Microsoft Cabinet

```
MSCF␀␀␀␀~␀␀␀␀␀␀␀␀,␀␀␀␀␀␀␀␀♥☺☺␀☻␀␀␀␀␀␀␀_␀␀␀☺␀␀␀►␀␀␀␀ ␀␀␀␀␀3<rN␣␀toto.py␀•␀␀␀►␀␀␀␀␀3<|N␣␀read
↨␀↨␀print␣"bonjour"◙blabla◙
```

Un exemple : Microsoft Cabinet

00000000	534d	4643	0000	0000	007e	0000	0000	0000	MSCF....~.......
00000016	002c	0000	0000	0000	0103	0001	0002	0000	,...............
00000032	0000	0000	005f	0000	0001	0000	0010	0000	...._..........
00000048	0000	0000	0000	3c33	4e72	0020	6f74	6f74	......3<rN .toto
00000064	702e	0079	0007	0000	0010	0000	0000	3c33	.py...........3<
00000080	4e7c	0020	6572	6461	656d	742e	7478	6100	|N .readme.txt.a
00000096	1943	170b	1700	7000	6972	746e	2220	6f62	C......print "bo
00000112	6a6e	756f	2272	620a	616c	6c62	0a61		njour".blabla.

– Entête, permettant d'identifier le type du fichier (ici : une archive CAB)
– Taille de l'archive (ici 127)
– Début du premier fichier de l'archive
– Nombre de fichiers dans l'archive (ici 2)

Suit ensuite une description de chaque fichier

– Taille du fichier (ici 16)
– Date et heure du fichier
– Permissions : 0x0020 signifie que le fichier est exécutable
– Nom du fichier

Puis le contenu de chaque fichier

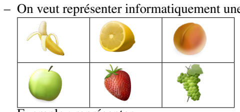
Autre exemple : GNU tar

Traité en TP :

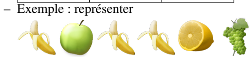
00000000	6f74	6f74	702e	0079	0000	0000	0000	0000	toto.py.........
00000016	0000	0000	0000	0000	0000	0000	0000	0000	....
…									
00000096	0000	0000	3030	3030	3537	0035	3030	3130	....0000755.0001
00000112	3537	0030	3030	3130	3537	0030	3030	3030	750.0001750.0000
00000128	3030	3030	3230	0030	3131	3233	3235	3137	0000020.11325271
00000144	3332	0030	3130	3532	3430	2000	0030	0000	230.012504. 0...
00000160	0000	0000	0000	0000	0000	0000	0000	0000	....
…									
00000256	7500	7473	7261	2020	6500	656a	6e61	6564	.ustar .ejeande
00000272	006c	0000	0000	0000	0000	0000	0000	0000	l...............
00000288	0000	0000	0000	0000	6500	656a	6e61	6564	.........ejeande
00000304	006c	0000	0000	0000	0000	0000	0000	0000	l...............
00000320	0000	0000	0000	0000	0000	0000	0000	0000	....
…									
00000512	7270	6e69	2074	6222	6e6f	6f6a	7275	0a22	print "bonjour".
00000528	0000	0000	0000	0000	0000	0000	0000	0000	....
…									
00001024	6572	6461	656d	742e	7478	0000	0000	0000	readme.txt......
00001040	0000	0000	0000	0000	0000	0000	0000	0000	....
…									
00001120	0000	0000	3030	3030	3436	0034	3030	3130	....0000644.0001
00001136	3537	0030	3030	3130	3537	0030	3030	3030	750.0001750.0000
00001152	3030	3030	3030	0037	3131	3233	3235	3137	0000007.11325271
00001168	3532	0035	3130	3133	3435	2000	0030	0000	255.013154. 0...
00001184	0000	0000	0000	0000	0000	0000	0000	0000	....
…									
00001280	7500	7473	7261	2020	6500	656a	6e61	6564	.ustar .ejeande
00001296	006c	0000	0000	0000	0000	0000	0000	0000	l...............
00001312	0000	0000	0000	0000	6500	656a	6e61	6564	.........ejeande
00001328	006c	0000	0000	0000	0000	0000	0000	0000	l...............
00001344	0000	0000	0000	0000	0000	0000	0000	0000	....
…									
00001536	6c62	6261	616c	000a	0000	0000	0000	0000	blabla..........
00001552	0000	0000	0000	0000	0000	0000	0000	0000	....
…									

Coder

– On veut représenter informatiquement une liste de fruits.

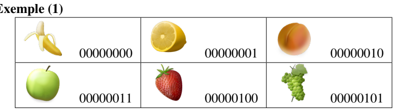


– Exemple : représenter



Dessins de Kevin Andersson - `www.kevinandersson.dk`

Exemple (1)



00000000	00000001	00000010
00000011	00000100	00000101

Codage :

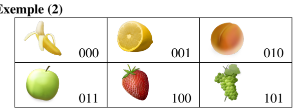
00000000 00000011 00000000 00000000 00000001 00000101

Décodage :

0000001000000001100000100000000000000000101000000000

00000010 00000011 00000100 00000000 00000101 00000000

Exemple (2)



000	001	010
011	100	101

Codage :

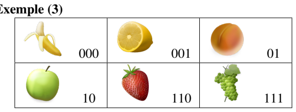
000 011 000 000 001 101

Décodage :

010011100000101000

010 011 100 000 101 000

Exemple (3)



000	001	01
10	110	111

Codage :

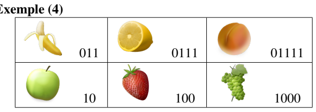
000 10 000 000 001 111

Décodage :

101100000101001

10 110 000 01 01 001

Exemple (4)

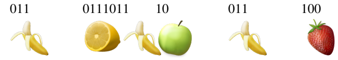


011	0111	01111
10	100	1000

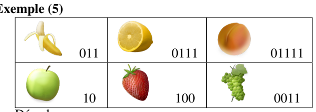
Décodage :

01101110011100

011 0111011 10 011 100



Exemple (5)

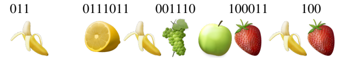


011	0111	01111
10	100	0011

Décodage :

01101110011100

011 0111011 001110 100011 100



Définitions

Définition 1. *Un code (binaire) est une façon d'associer à chaque objet (ici des fruits) une suite de* 0 *et de* 1 *(un* mot*)*

Définition 2. *Un code est uniquement déchiffrable (ou* non ambigu*) si on peut retrouver de façon unique toute liste d'objets à partir de la suite de* 0 *et de* 1*.*

Les 5 exemples précédents sont non ambigus, sauf le dernier.

Questions

Problème 1. *Comment peut-on savoir qu'un code donné est uniquement déchiffrable ?*

Problème 2. *Comment créer un code uniquement déchiffrable ?*

Code bloc

Définition 3. *Un block code est un code où tous les mots du code sont différents et de même longueur.*

 000	 101	 100
 011	 110	 111

Proposition 3. *Un block code est non-ambigu.*

Inégalité de Kraft-McMillan

Théorème 4. *On note l_i la longueur du code pour le i-ème objet. On suppose qu'il y a n objets.*
Si un code est non-ambigu, alors

$$\frac{1}{2^{l_1}} + \frac{1}{2^{l_2}} + \cdots + \frac{1}{2^{l_n}} \leq 1$$

Exemple : peut-on trouver un code pour les fruits de sorte que
-  soit codé sur 1 bit ;
-  soient codés sur 3 bits ;
- sur 4 bits ?

Test de Sardinas-Patterson
Soit C le code. On le met dans une colonne C_0
A chaque étape i
- Si un mot de la première colonne C_0 commence un mot de la dernière colonne C_i (ou vice versa), on met le reste dans une nouvelle colonne C_{i+1}

On arrête le calcul lorsqu'on boucle

Théorème 5. *C est un code non-ambigu si et seulement si on n'atteint jamais un mot de C au cours de l'exécution*

Exemple

1000	1	000	0	000	111	00
0111		001	1	111	0	000
1001		110		001	10	111
1110				110	11	01
011				11	1	001
00				0		0
						10
						110
						11

C est ambigu

Exemple

1000	0	000	0	00	111	00	00
0111	1	111	10	111	0	111	111
1110		110		11	10	0	0
011		11		0	11	10	10
100		00				11	11

C est non ambigu

Code préfixe

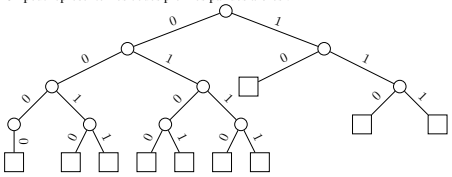
Définition 4. *Un code est préfixe si aucun mot n'est un préfixe d'un autre mot.*

Exemple :

0000	0101		110
0010	0110		111
0011	0111		
0100	10		

Remarque : un block code est préfixe.

Code préfixe
On peut représenter les codes préfixes par des arbres :



Codes préfixes

Théorème 6. *Un code préfixe est non-ambigu.*

Preuve : pour décoder, on suit le chemin dans l'arbre. Dès qu'on arrive à une feuille, on a fini et on repart du début.
- Les codes préfixes sont *instantanés* : Dès qu'on a fini de lire le premier mot de code, on *sait* qu'on l'a lu et qu'on peut passer au deuxième.

Théorème de Kraft-McMillan

Théorème 7. *Soit l_i des entiers. Si les l_i vérifient*

$$\frac{1}{2^{l_1}} + \frac{1}{2^{l_2}} + \cdots + \frac{1}{2^{l_n}} \leq 1$$

Alors il existe un code préfixe tel que le $i$ème objet a pour longueur l_i.

Ensemble

Théorème 8. *On note l_i la longueur du code pour le i-ème objet. Si le code est non-ambigu, alors*

$$\frac{1}{2^{l_1}} + \frac{1}{2^{l_2}} + \cdots + \frac{1}{2^{l_n}} \leq 1$$

Théorème 9. *Soit l_i des entiers. Si les l_i vérifient*

$$\frac{1}{2^{l_1}} + \frac{1}{2^{l_2}} + \cdots + \frac{1}{2^{l_n}} \leq 1$$

Alors il existe un code préfixe tel que le $i$ème objet a pour longueur l_i.

Corollaire

Théorème 10. *Si C est un code non-ambigu, on peut trouver un code préfixe avec exactement les mêmes longueurs de mot.*

Les codes qui ne sont pas préfixes ne servent à rien.
Comment trouver, sachant les l_i, le code préfixe qui convient ?

Conclusion
- Pour coder des objets, on utilisera des codes non-ambigus, et souvent des codes préfixes.
- Si on sait qu'un objet apparait très souvent, il faut lui donner un code plus petit que les autres.

Comment faire ? C'est le prochain cours.