Programmation 1: Fondamentaux

Binaire et Codage

- Codage en binaire, octal et hexadécimal
- Conversions
- Codage des caractères

Forme des données en mémoire

- Toutes les données de la mémoire sont des **séquences de 0 et de 1**, appelés bits.

Type de séquence	Nombre de bits utilisés	exemple
Bit	1	1 ou 0
Octet ou byte	8	10101111
Mot 16 bits ou Word	16	1111 0000 1111 0000
Double mot ou DWORD	32	1001 1110 0000 1111 1010 0101 0000 0001
QWORD	64	1001 1110 0000 …. …. …. …. …. ……. …. …. …. 1010 0101 0000 0001
Longueur variable	n	01101010100100101010

Format des données en mémoire

Dans les langages de programmation, les types de données donnent du sens aux bits : c'est le développeur qui décide de la manière d'interpréter les données.

- Par exemple la valeur binaire 0100 0001 peut être comprise comme :
  - La représentation d'un nombre entier décimal égal à 65
  - La représentation du caractère A
  - Un simple champ de bits : 8 bits consécutifs qui représentent l'état de 8 variables binaires

- Suivant le nombre de bits utilisés on peut coder aussi d'autres types de données:
  - Des valeurs booléennes (vrai – faux)
  - Des entiers courts, longs, signés ou non signés
  - Des nombres décimaux en simple ou double précision
  - Des caractères (Unicode – UTF 8 ou autres)
  - Des adresses (notion de pointeur)

Format des données en mémoire

Le **codage** est donc :

- Le **nombre de bits** utilisés pour représenter la donnée
- Le processus d'**interprétation** de ces bits

Les bases

Une base indique le nombre de symboles utilisés pour représenter une donnée:

Nom	Nombre de symboles	symboles
binaire	2	0, 1
octal	8	0,1,2,3,4,5,6,7
décimal	10	0,1,2,3,4,5,6,7,8,9
hexadécimal	16	0..9, A, B, C, D, E, F

Les bases

Notation:

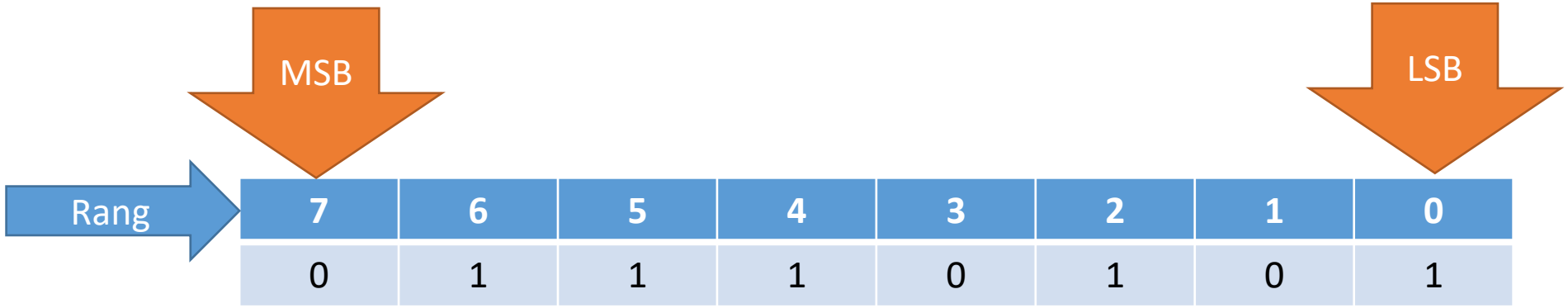
- On indique la valeur de la base en indice du nombre
- Dans le code et suivant le langage, avec une syntaxe spécifique

Nom	écriture	Syntaxe en C	Valeur en décimal
binaire	n_2	0b1000010	66
octal	n_8	0102	66
décimal	n_{10}	66	66
hexadécimal	n_{16}	0x42	66

Les bases

Base 2:

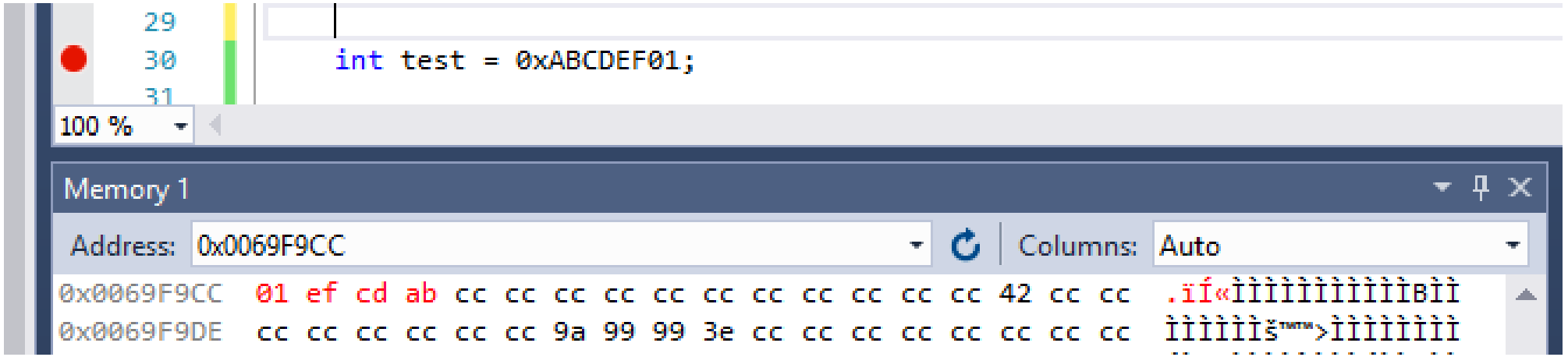
- Les digits sont des **0** et des **1**
- Dans une séquence de bits, celui qui est le plus à gauche est le **MSB** – Most Significant Bit, celui dont le poids est le plus élevé. On parle de poids fort.
- Dans une séquence de bits, celui qui est le plus à droite est le **LSB** – Less Significant Bit, celui dont le poids est le plus faible. On parle de poids faible.
- Quand on parle de MSB, il est important de connaitre la longueur de la séquence (8, 16, 32, 64 bits ou quelconque, mais cela a moins de sens)



Les bases

Base 2:

- La mémoire de l'ordinateur et les fichiers stockent des Octets
- Suivant le type d'architecture, on peut trouver des boutismes différents, c'est-à-dire que les octets qui représentent les données sont rangés soit en commençant par celui qui contient le LSB (Little-endian) soit celui qui contient le MSB (Big-endian)



Les bases

Base 2:

- Comme en base 10, le digit le plus à droite représente les unités
- Chaque rang représente une puissance de 2

rang	7	6	5	4	3	2	1	0
2^n	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
bit	0	1	1	1	0	1	0	1
Valeur décimale	128 x 0	64 x 1	32 x 1	16 x 1	8 x 0	4 x 1	2 x 0	1 x 1

Les bases

Base 2: conversion en base 10 : $x_{10} = \sum_{i=0}^{n-1} d_i 2^i$

n : nombre de bits du mot binaire, d : valeur du digit (0 ou 1)

rang	7	6	5	4	3	2	1	0
2^n	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
bit	0	1	1	1	0	1	0	1
Valeur décimale	128 x 0	64 x 1	32 x 1	16 x 1	8 x 0	4 x 1	2 x 0	1 x 1

$0111\ 0101_2 = 64 + 32 + 16 + 4 + 1 = 117_{10}$

Les bases

Base 8: Les bits sont regroupés par 3

rang	7	6	5	4	3	2	1	0
2^n	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
bit	0	1	1	1	0	1	0	1
Valeur décimale	128 x 0	64 x 1	32 x 1	16 x 1	8 x 0	4 x 1	2 x 0	1 x 1
Octal	1			6			5	
8^n	8^2			8^1			8^0	

$117_{10} = 0111\ 0101_2 = 165_8 = (8^2 \times 1) + (8^1 \times 6) + 5$

Les bases

Base 16: Les bits sont regroupés par 4

Les digits sont 0 1 2 3 4 5 6 7 8 9 A B C D E F

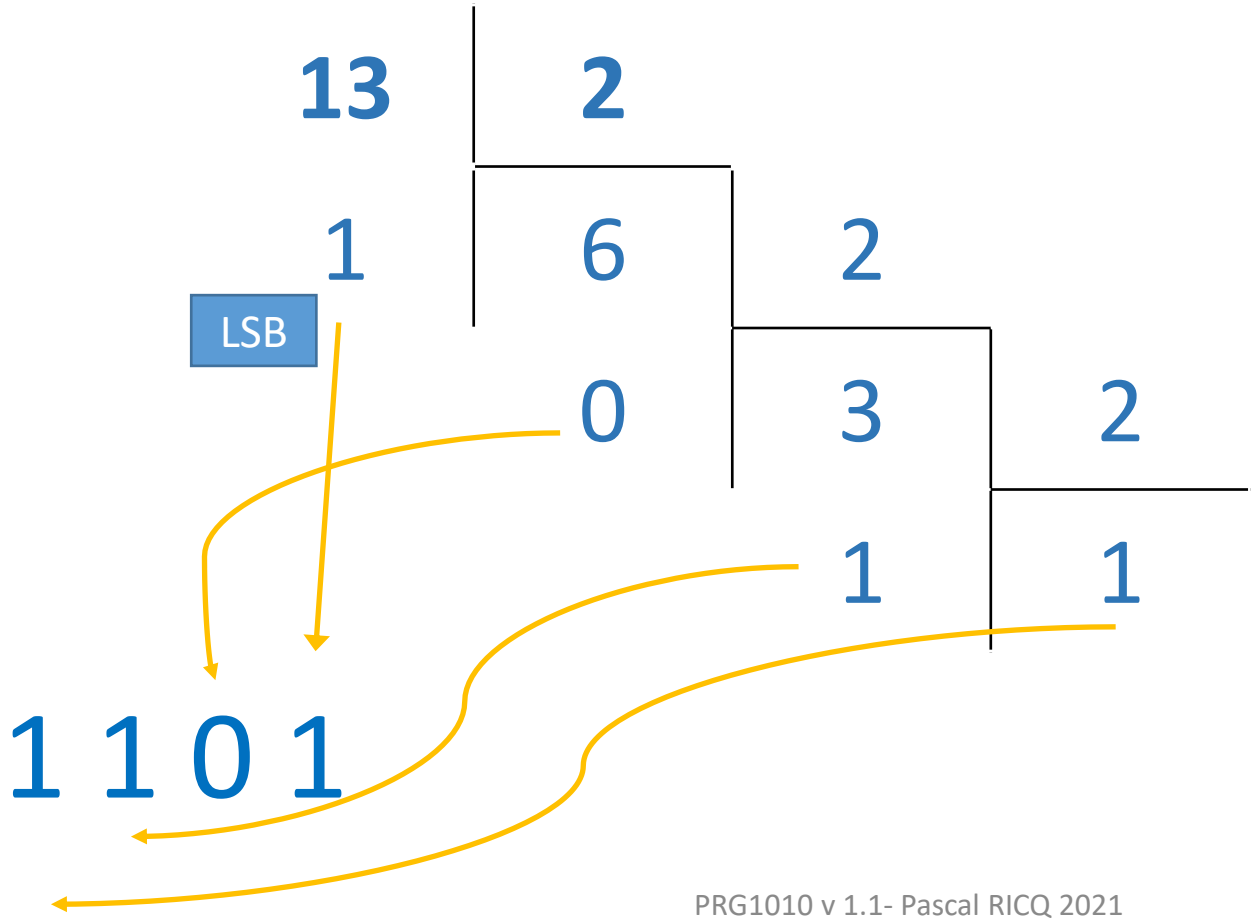
rang	7	6	5	4	3	2	1	0
2^n	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
bit	0	1	1	1	0	1	0	1
Valeur décimale	128 x 0	64 x 1	32 x 1	16 x 1	8 x 0	4 x 1	2 x 0	1 x 1
Hexa	7				5			
16^n		16^1				16^0		

$117_{10} = 0111\ 0101_2 = 75_{16} = (16^1 \times 7) + 5$

Les bases

Conversion: Valeur décimale en binaire

- Par divisions Euclidiennes successives par 2



Les bases

Exercice 1

Convertissez les mots suivants de la base 2 vers la base 10 :

- $000101_2 = ?_{10}$
- $101010_2 = ?_{10}$
- $010100001_2 = ?_{10}$
- $111000010_2 = ?_{10}$
- $001100101001_2 = ?_{10}$
- $010110100000_2 = ?_{10}$

Les bases

Corrigé 1

Convertissez les mots suivants de la base 2 vers la base 10 :

- $000101_2 = ?_{10}$

  5_{10}

- $101010_2 = ?_{10}$

  42_{10}

- $010100001_2 = ?_{10}$

  161_{10}

- $111000010_2 = ?_{10}$

  450_{10}

- $001100101001_2 = ?_{10}$

  809_{10}

- $010110100000_2 = ?_{10}$

  1440_{10}

Les bases

Exercice 2

Convertissez les mots suivants de la base 2 vers la base 8 :

- $000101_2 = ?_8$
- $101010_2 = ?_8$
- $010100001_2 = ?_8$
- $111000010_2 = ?_8$
- $001100101001_2 = ?_8$
- $010110100000_2 = ?_8$

Les bases

Corrigé 2

Convertissez les mots suivants de la base 2 vers la base 8 :

- $000101_2 = ?_8$

  5_8

- $101010_2 = ?_8$

  52_8

- $010100001_2 = ?_8$

  241_8

- $111000010_2 = ?_8$

  702_8

- $001100101001_2 = ?_8$

  1451_8

- $010110100000_2 = ?_8$

  2640_8

Les bases

Exercice 3

Convertissez les mots suivants de la base 2 vers la base 16 :

- $000101_2 = ?_{16}$
- $101010_2 = ?_{16}$
- $010100001_2 = ?_{16}$
- $111000010_2 = ?_{16}$
- $001100101001_2 = ?_{16}$
- $010110100000_2 = ?_{16}$

Les bases

Corrigé 3

Convertissez les mots suivants de la base 2 vers la base 16 :

- $000101_2 = ?_{16}$

  5_{16}

- $101010_2 = ?_{16}$

  $2A_{16}$

- $010100001_2 = ?_{16}$

  $A1_{16}$

- $111000010_2 = ?_{16}$

  $1C2_{16}$

- $001100101001_2 = ?_{16}$

  329_{16}

- $010110100000_2 = ?_{16}$

  $5A0_{16}$

Codages des caractères

Pour répondre aux besoins d'échanges de texte au format numérique,

l'ISO fut chargé dans les années 60 de fixer un standard pour coder les caractères.

Le premier standard retenu est le format **ASCII** (American Standard Code for Information Interchange),

Le format à taille fixe alors le plus fréquemment utilisé.

Codage

ASCII n'emploie que 7 bits pour représenter chaque caractère.
Pour la même raison que le codage des booléens, les ordinateurs ne manipulent que des octets (8 bits regroupés).
C'est pourquoi le format ASCII ajoute un bit (le MSB) qui vaut toujours 0.
Un caractère est donc codé selon la formule:

$$c = 0XXX\ XXXX_{ASCII}$$

De cette manière, tous les caractères sont alignés sur la taille des octets.

Codages des caractères

Contenu de la table ASCII Standard

La table ASCII contient 2^7 = 128 caractères.
Les caractères de 0 à 31 et le caractère 127 sont des caractères de contrôle: ils ne sont pas imprimables.
Ensuite, nous retrouvons les caractères imprimables: la ponctuation, les nombres, l'alphabet,...

La table ASCII est consultable à cette page.

Table des codes ASCII étendus OEM et ANSI

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR OEM	CHAR ANSI	DEC	HEX	CHAR OEM	CHAR ANSI	DEC	HEX	CHAR OEM	CHAR ANSI	DEC	HEX	CHAR OEM	CHAR ANSI
0	00		32	20		64	40	@	96	60	`	128	80	Ç	€	160	A0	á		192	C0	└	À	224	E0	α	à
1	01	☺	33	21	!	65	41	A	97	61	a	129	81	ü		161	A1	í	¡	193	C1	┴	Á	225	E1	ß	á
2	02	☻	34	22	"	66	42	B	98	62	b	130	82	é	‚	162	A2	ó	¢	194	C2	┬	Â	226	E2	Γ	â
3	03	♥	35	23	#	67	43	C	99	63	c	131	83	â	*f*	163	A3	ú	£	195	C3	├	Ã	227	E3	∏	ã
4	04	♦	36	24	$	68	44	D	100	64	d	132	84	ä	„	164	A4	ñ	¤	196	C4	─	Ä	228	E4	∑	ä
5	05	♣	37	25	%	69	45	E	101	65	e	133	85	à	…	165	A5	Ñ	¥	197	C5	┼	Å	229	E5	σ	å
6	06	♠	38	26	&	70	46	F	102	66	f	134	86	å	†	166	A6	ª	¦	198	C6	╞	Æ	230	E6	µ	æ
7	07	•	39	27	'	71	47	G	103	67	g	135	87	ç	‡	167	A7	º	§	199	C7	╟	Ç	231	E7	τ	ç
8	08	◘	40	28	(	72	48	H	104	68	h	136	88	ê	ˆ	168	A8	¿	¨	200	C8	╚	È	232	E8	Φ	è
9	09	○	41	29	)	73	49	I	105	69	i	137	89	ë	‰	169	A9	⌐	©	201	C9	╔	É	233	E9	θ	é
10	0A	◙	42	2A	*	74	4A	J	106	6A	j	138	8A	è	Š	170	AA	¬	ª	202	CA	╩	Ê	234	EA	Ω	ê
11	0B	♂	43	2B	+	75	4B	K	107	6B	k	139	8B	ï	‹	171	AB	½	«	203	CB	╦	Ë	235	EB	δ	ë
12	0C	♀	44	2C	,	76	4C	L	108	6C	l	140	8C	î	Œ	172	AC	¼	¬	204	CC	╠	Ì	236	EC	∞	ì
13	0D	♪	45	2D	-	77	4D	M	109	6D	m	141	8D	ì		173	AD	¡	-	205	CD	═	Í	237	ED	Ø	í
14	0E	♫	46	2E	.	78	4E	N	110	6E	n	142	8E	Ä	Ž	174	AE	«	®	206	CE	╬	Î	238	EE	∈	î
15	0F	☼	47	2F	/	79	4F	O	111	6F	o	143	8F	Å		175	AF	»	¯	207	CF	╧	Ï	239	EF	∩	ï
16	10	►	48	30	0	80	50	P	112	70	p	144	90	É		176	B0	░	°	208	D0	╨	Đ	240	F0	≡	ð
17	11	◄	49	31	1	81	51	Q	113	71	q	145	91	æ	‘	177	B1	▒	±	209	D1	╤	Ñ	241	F1	±	ñ
18	12	↕	50	32	2	82	52	R	114	72	r	146	92	Æ	’	178	B2	▓	²	210	D2	╥	Ò	242	F2	≥	ò
19	13	‼	51	33	3	83	53	S	115	73	s	147	93	ô	“	179	B3	│	³	211	D3	╙	Ó	243	F3	≤	ó
20	14	¶	52	34	4	84	54	T	116	74	t	148	94	ö	”	180	B4	┤	´	212	D4	╘	Ô	244	F4	⌠	ô
21	15	§	53	35	5	85	55	U	117	75	u	149	95	ò	•	181	B5	╡	µ	213	D5	╒	Õ	245	F5	⌡	õ
22	16	▬	54	36	6	86	56	V	118	76	v	150	96	û	–	182	B6	╢	¶	214	D6	╓	Ö	246	F6	÷	ö
23	17	↨	55	37	7	87	57	W	119	77	w	151	97	ù	—	183	B7	╖	·	215	D7	╫	×	247	F7	≈	÷
24	18	↑	56	38	8	88	58	X	120	78	x	152	98	ÿ	˜	184	B8	╕	¸	216	D8	╪	Ø	248	F8	°	ø
25	19	↓	57	39	9	89	59	Y	121	79	y	153	99	Ö	™	185	B9	╣	¹	217	D9	┘	Ù	249	F9	¨	ù
26	1A	→	58	3A	:	90	5A	Z	122	7A	z	154	9A	Ü	Š	186	BA	║	º	218	DA	┌	Ú	250	FA	·	ú
27	1B	←	59	3B	;	91	5B	[	123	7B	{	155	9B	ø	›	187	BB	╗	»	219	DB	█	Û	251	FB	√	û
28	1C	∟	60	3C	<	92	5C	\	124	7C	|	156	9C	£	œ	188	BC	╝	¼	220	DC	▄	Ü	252	FC	³	ü
29	1D	↔	61	3D	=	93	5D	]	125	7D	}	157	9D	Ø		189	BD	╜	½	221	DD	▌	Ý	253	FD	²	ý
30	1E	▲	62	3E	>	94	5E	^	126	7E	~	158	9E	×	ž	190	BE	╛	¾	222	DE	▐	Þ	254	FE	■	þ
31	1F	▼	63	3F	?	95	5F	_	127	7F	⌂	159	9F	*f*	Ÿ	191	BF	┐	¿	223	DF	▀	ß	255	FF		ÿ

Codages des caractères

Chaîne de caractères

Sous leur forme la plus simple comme en langage C, les chaînes de caractères sont un assemblage de codes caractères.
Par exemple pour représenter la lettre A ont utilise le code 65. La table ASCII permet de faire la correspondance entre un code et une typographie de caractère.

Caractère de terminaison

Une chaîne de caractère s'achève nécessairement par un caractère de terminaison: le caractère NUL.
Sa valeur est 0 dans la table ASCII

Codages des caractères

Caractère

En langage C, pour manipuler un **caractère unique**, on le place entre côtes lorsqu'il s'agit de la forme littérale 'A'

Une variable qui contient un unique caractère sera déclarée de la façon suivante :

```
char uneLettre = 'A';
```

Le code suivant produit le même résultat :

```
char uneLettre = 65;
```

Codages des caractères

Caractères non imprimables

Pour insérer un caractère c non imprimable dans un programme,
un antislash '\' est utilisé pour différencier les caractères non imprimables des caractères imprimables.
Quelques caractères usuellement rencontrés:

caractère	notation	valeur	description
NUL	'\0'	0	terminaison
HT	'\t'	9	Horizontal tabulation
CR	'\n'	13	retour à la ligne

Codages des caractères

Chaîne de caractères

Le langage C n'utilise pas de type spécifique pour manipuler les chaînes de caractères.
Le type string n'existe pas.

Le langage C utilise un tableau de caractères. Notons la place supplémentaire prise par le caractère de terminaison à la fin de la chaîne.

indice	0	1	2	3	4	5	6
caractère	's'	't'	'r'	'i'	'n'	'g'	'\0'

Les implementations modernes utilisent des encodages de caractères sur un ou plusieurs octets, grâce à l'utilisation de **wide char** :

```
wchar_t unMot[6] = { L'h', L'e', L'l', L'l', L'o', 0 }; // hello est formé de caractères codés sur 2 octets
```

Codages des caractères

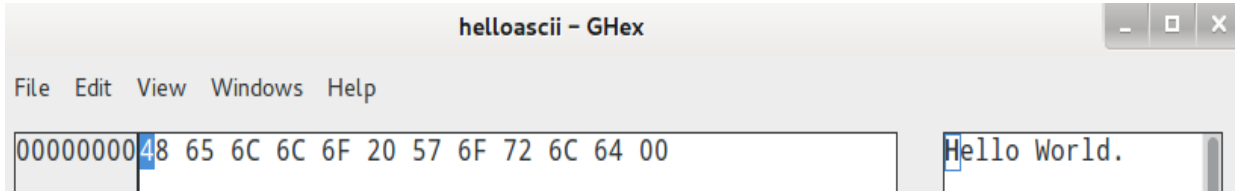
Codage de Hello World

indice	0	1	2	3	4	5	6	7	8	9	10	11
caractère	'H'	'e'	'l'	'l'	'o'	' '	'W'	'o'	'r'	'l'	'd'	'\0'
décimal	72	101	108	108	111	32	87	111	114	108	100	0
binaire	0100 1000	0110 0101	0110 1100	0110 1100	0110 1111	0010 0000	0101 0111	0110 1111	0111 0010	0110 1100	0110 0100	0000 0000

La séquence de bits vaut:

0100 1000 0110 0101 0110 1100 0110 1100 0110 1111 0010 0000 0101 0111 0110 1111 0111 0010 0110 1100 0110 0100 0000 0000$_{ASCII}$

Ou encore 48 65 6C 6C 6F 20 57 6F 72 6C 64 00 en hexadécimal (tel que l'on pourra la voir dans un éditeur hexadécimal



Codages des caractères

ASCII Etendu

Il existe de nombreuses extensions au format ASCII.
Le format Latin-1 (connu également sous le nom de ISO-8859-1 ou Europe Occidentale) ajoute des caractères accentués qui ne sont pas présents dans le format d'origine.

Codage

Latin-1 exploite le bit du MSB laissé à 0 dans le format ASCII.
Un caractère est donc codé selon la formule:

$$c = XXXX\ XXXX_{Latin\text{-}1}$$

De cette manière, tous les caractères de Latin-1 occupent un octet complet.

Codages des caractères

ASCII Etendu

La table Latin-1 contient $2^8 = 256$ caractères.
Les 128 premiers caractères sont identiques à ceux de la table ASCII:
ils sont codés sous la forme $0XXX\ XXXX_{Latin\text{-}1}$
Les nouveaux caractères occupent les valeurs suivantes:
ils sont codés sous la forme $1XXX\ XXXX_{Latin\text{-}1}$

La table Latin-1 est consultable à cette page.

Codages des caractères

Motivations

Le format **Latin-1** permet de représenter deux fois plus de caractères que **ASCII**, mais cela se révèle insuffisant pour représenter tous les symboles de tous les langages du monde.

Utiliser un **format à longueur fixe** sur encore plus de bits amènerait à augmenter l'espace mémoire occupé de manière systématique, même si seuls les caractères correspondant aux langues à écriture latine, en particulier l'anglais, sont utilisés.

Les **formats à longueur variable** permettent de conserver une occupation en mémoire similaire à Latin-1 pour les langues à écriture latine, et plus grande pour les caractères "étendus".

Codages des caractères

UTF-8

UTF-8 (Universal Character Set Transformation Format - 8 bits) est le format à longueur variable dominant pour le codage des pages Web.
Il permet de coder les caractères définis par Unicode sous la forme d'octets.
Notons qu'il existe également les formats UTF-16 et UTF-32 (manipulant des mots de 16 et 32 bits), mais attention, contrairement à UTF-8 et UTF-16, UTF-32 est un format à longueur fixe. Son usage est très limité.

Unicode

Unicode est une norme ayant pour but de donner un nom et un identifiant (le code point, noté U+xxxx en hexadécimal) à tous les caractères de tous les langages du monde, mais aussi les symboles monétaires, mathématiques, ...
Il y a 1 114 112 identifiants possibles, mais beaucoup sont réservés à un usage futur.
Le standard Unicode est constitué d'un répertoire de 143 859 caractères, couvrant plus de 150 écritures.

Codages des caractères

Principe

UTF-8 utilise 1 à 4 octets pour représenter un caractère.
Il est compatible avec ASCII: les 128 premiers caractères de UTF-8 sont ceux de ASCII.
Mais il n'est pas aussi simple de couper un texte en caractères que dans un format fixe.

Pour chaque caractère, il faut :

- Déterminer sur combien d'octets il est codé.
- Récupérer l'information utile (le code point) dispersée sur ces octets et les "recoller".

Le nombre d'octets utilisé pour coder le caractère est contenu dans le caractère lui-même.

Codages des caractères

Concrètement

On regarde le premier octet du caractère.

- Si son MSB est à 0, il s'agit d'un caractère ASCII, de la forme 0XXXXXXX.
  On le décode directement avec la table ASCII ou UTF-8.

- Si son MSB est à 1, il s'agit du premier octet d'un ensemble d'octets.
  Il est de la forme 1 .. 10XX .. X, avec autant de 1 que d'octets utilisés.
  Comme les caractères sont codés sur une longueur variant de 1 à 4 octets, les prefixes possibles sont 10, 110, 1110, 11110.
  Les bits suivants dans l'octet commencent à coder le code point du caractère.
  Les octets suivants seront tous de la forme 10XXXXXX

Codages des caractères

Dynamique

UTF-8 est un codage à taille variable occupant de 1 à 4 octets

Définition du nombre d'octets utilisés

Représentation binaire UTF-8	Signification
0XXXXXXX	1 octet codant 1 à 7 bits
110XXXXX 10XXXXXX	2 octets codant 8 à 11 bits
1110XXXX 10XXXXXX 10XXXXXX	3 octets codant 12 à 16 bits
11110XXX 10XXXXXX 10XXXXXX 10XXXXXX	4 octets codant 17 à 21 bits

Les 128 premiers caractères sont identiques à ceux de la table ASCII:
ils sont codés sous la forme $0XXX\ XXXX_{UTF\text{-}8}$ que l'on retrouve dans les caractères UTF-8 codés sur un octet.
Par contre les 128 valeurs suivantes n'ont pas de sens en UTF-8:
les caractères ajoutés par Latin-1, codés sous la forme 1XXX XXXX, ne sont pas transposables directement en UTF-8.

Codages des caractères

Exemples de caractères

UTF-8 est un codage à taille variable occupant de 1 à 4 octets

Définition du nombre d'octets utilisés

Code Point (Hexadécimal)	Caractère	Valeur scalaire (décimal)	Valeur scalaire (binaire naturel)	Codage en UTF-8 (binaire)	Codage en UTF-8 (Hexadécimal)
U+0041	A	65	1000001	01000001	41
U+00F4	ô	244	11 110100	11000011 10110100	C3 B4
U+20AC	€	8364	0010 000010 101100	11100010 10000010 10101100	

On peut remarquer que les octets sont remplis de droite à gauche.
Des 0 sont insérés dans le premier octet si nécessaire (en orange).

La table UTF-8 est consultable à cette page.