Introduction aux systèmes informatiques

Codage et Systèmes d'exploitation

Pôle ASR – Module M1101 – Semestre 1

Bruno BEAUFILS

⟨bruno.beaufils@univ-lille.fr⟩

https://beaufils.u-lille.fr

Université de Lille, IUT « A », Département informatique

Année 2020/2021

http://m1101.iutinfo.fr

Codage

Langage écrit

- Une langue est *souvent* écrite à partir de mots composés de *lettres*
- Les lettres sont définies dans un *alphabet*
  - liste de représentations graphiques (*glyphe*, *dessin*)
  - plusieurs casses possibles (capitale ou minuscule)

- Les phrases sont organisées grâce à la *ponctuation*

- lettres, ponctuations et chiffres sont représentés par des dessins

- Écrire du texte :
  - utiliser un sens d'écriture
  - notion de lignes et de position
  - déplacer le *stylo*

Écrire ⇔ Dessiner

Manipuler du texte ⇒ Coder ces symboles (caractères)

Langage écrit

- Une langue est *souvent* écrite à partir de mots composés de *lettres*
- Les lettres sont définies dans un *alphabet*
  - liste de représentations graphiques (*glyphe*, *dessin*)
  - plusieurs casses possibles (capitale ou minuscule)
- Les phrases sont organisées grâce à la *ponctuation*
- lettres, ponctuations et chiffres sont représentés par des dessins
- Écrire du texte :
  - utiliser un sens d'écriture
  - notion de lignes et de position
  - déplacer le *stylo*

Écrire ⇔ Dessiner

Manipuler du texte ⇒ Coder ces symboles (caractères)

Langage écrit

- Une langue est *souvent* écrite à partir de mots composés de *lettres*
- Les lettres sont définies dans un *alphabet*
  - liste de représentations graphiques (*glyphe*, *dessin*)
  - plusieurs casses possibles (capitale ou minuscule)
- Les phrases sont organisées grâce à la *ponctuation*
- lettres, ponctuations et chiffres sont représentés par des dessins
- Écrire du texte :
  - utiliser un sens d'écriture
  - notion de lignes et de position
  - déplacer le *stylo*

Écrire ⇔ Dessiner

Manipuler du texte ⇒ Coder ces symboles (caractères)

Langage écrit

- Une langue est *souvent* écrite à partir de mots composés de *lettres*
- Les lettres sont définies dans un *alphabet*
  - liste de représentations graphiques (*glyphe*, *dessin*)
  - plusieurs casses possibles (capitale ou minuscule)
- Les phrases sont organisées grâce à la *ponctuation*
- lettres, ponctuations et chiffres sont représentés par des dessins
- Écrire du texte :
  - utiliser un sens d'écriture
  - notion de lignes et de position
  - déplacer le *stylo*

Écrire ⇔ Dessiner

Manipuler du texte ⇒ Coder ces symboles (caractères)

Du texte au(x) glyphe(s)

- Les écrits sous forme d'images ne sont pas exploitables ;
- L'écriture est donc simplifiée pour ne retenir que les *caractères* les uns à la suite des autres (≠ *lettres*) ;

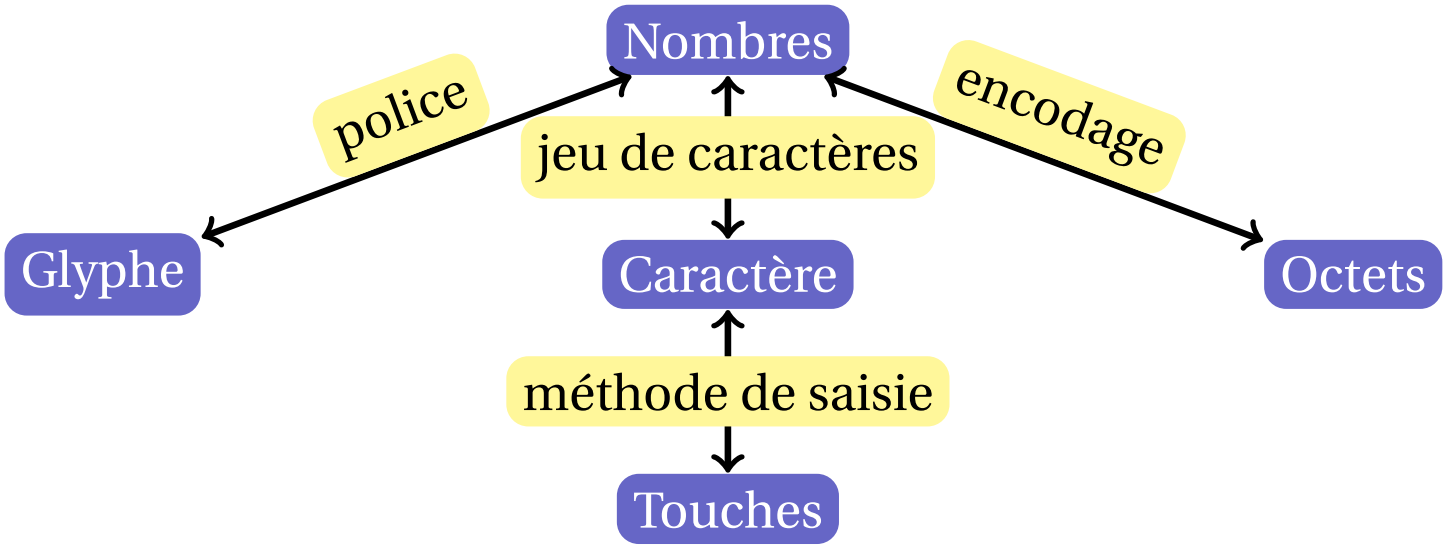
- Les glyphes sont les dessins des lettres, différents selon les polices
- Les polices supportent souvent plusieurs jeux de caractères. Le dessin n'y est stocké qu'une fois.

Différence de glyphes

La lettre $\mathscr{A}$ et A représentent le même caractère mais pas le même que A (α capitale).
De même le *a* de *Abba*, de Abba, ABBA ou Abba sont les mêmes caractères.

Du texte au(x) glyphe(s)

- Les écrits sous forme d'images ne sont pas exploitables ;
- L'écriture est donc simplifiée pour ne retenir que les *caractères* les uns à la suite des autres (≠ *lettres*) ;



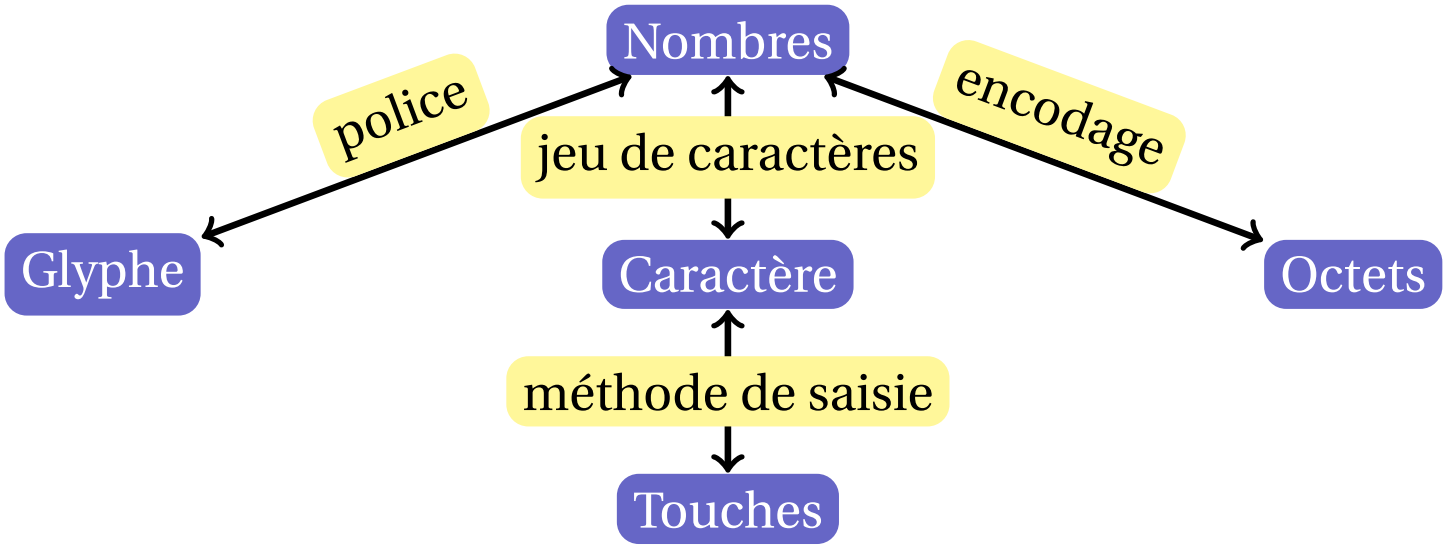
- Les glyphes sont les dessins des lettres, différents selon les polices
- Les polices supportent souvent plusieurs jeux de caractères. Le dessin n'y est stocké qu'une fois.

Différence de glyphes

La lettre $\mathscr{A}$ et A représentent le même caractère mais pas le même que A (α capitale).
De même le *a* de *Abba*, de Abba, ABBA ou Abba sont les mêmes caractères.

Du texte au(x) glyphe(s)

- Les écrits sous forme d'images ne sont pas exploitables ;
- L'écriture est donc simplifiée pour ne retenir que les *caractères* les uns à la suite des autres (≠ *lettres*) ;



- Les glyphes sont les dessins des lettres, différents selon les polices
- Les polices supportent souvent plusieurs jeux de caractères. Le dessin n'y est stocké qu'une fois.

Différence de glyphes

La lettre $\mathscr{A}$ et A représentent le même caractère mais pas le même que A (α capitale).
De même le *a* de *Abba*, de Abba, ABBA ou Abba sont les mêmes caractères.

Qu'est-ce qu'un caractère ?

- Au début : lettres, chiffres, ponctuation simplifiée.
- Correspondait grossièrement à une touche de machine à écrire (+Capitale/Minuscule)
- Au fur et à mesure, de très nombreux caractères ont été rajoutés.
- Jeu de caractères universel : Unicode.

Quelques caractères dont vous ne connaissez peut-être pas les noms

C	Usuel	Français	Anglais
#	dièse	croisillon, octothorpe	hash, number sign
&	et	esperluette, et commercial	ampersand, and
|	ou, *païpe*	barre verticale	pipe
/	slash	barre oblique	slash
@	*arobasse*	arobase	at, at sign
\	backslash	contre-oblique	backslash
_	underscore	(blanc) souligné	underscore
[]	crochets	crochets	(square) brackets
{}	accolades	accolades	(curly) braces

Jeux de caractères

- Plusieurs jeux de caractères primitifs sur 7 ou 8 bits par caractère.

- Un seul a vraiment survécu : `US-ASCII`
  - 7 bits pour coder les caractères
  - 1 bit pour contrôler la parité

- Création de jeux de caractères nationaux
  - Normes ISO-8859-*
    - caractères 0 à 127 = US-ASCII
    - caractères 128 à 255 = caractères locaux
  - Autres méthodes :
    - KOI-8R (russe)
    - JIS (Japonais)
    - BIG5 (Chinois)
    - collections de caractères

- Universalisation : Unicode → plus de 100 000 caractères.

Le codage US-ASCII

American Standard Code for Information Interchange

- code standardisé par l'ANSI

  American National Standards Institute
- créé pour représenter du texte anglais
- utilise 7 bits
- inclut
  - lettres latines
  - chiffres arabes
  - caractères de ponctuation
  - caractères de contrôle

Le codage US-ASCII

- 32 caractères de *contrôle*,
- 96 caractères *affichables*;
- Unicode, ISO-8859 compatibles avec ASCII.

00 NUL	01 SOH	02 STX	03 ETX	04 EOT	05 ENQ	06 ACK	07 BEL
08 BS	09 HT	0A LF	0B VT	0C NP	0D CR	0E SO	0F SI
10 DLE	11 DC1	12 DC2	13 DC3	14 DC4	15 NAK	16 SYN	17 ETB
18 CAN	19 EM	1A SUB	1B ESC	1C FS	1D GS	1E RS	1F US
20 SP	21 !	22 "	23 #	24 $	25 %	26 &	27 ’
28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _
60 ‘	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C |	7D }	7E ~	7F DEL

Limite du codage sur un seul octet

- ASCII sur 8 bits avait un bit *inutilisé*.
- Langues asiatiques : pas suffisant.
- Codage à décalage : certaines séquences (non rencontrées habituellement) permettent de changer de « zone » de caractères.
- Certaines séquences déclenchent du codage où 1 caractère est codé par 2 octets.
- Rupture de l'égalité 1 octet = 1 caractère
- Autres codages : BIG5 est un codage à 2 octets par caractères pour le chinois.

Unicode et UTF-8

- Unicode est une collection de plus de 100 000 caractères
  - UCS : *Universal Character Set*
  - ne spécifie pas la façon de représenter par une séquence d'octets
  - taille maximale est de 17×2^{16}
  - le code maximal 0x10FFFF
- UTF-8 est une façon de transformer un numéro en une séquence d'octets
  - UTF-8 : *UCS Transformation Format 8 bits*

Valeurs	Écriture binaire	Codage UTF-8 (binaire)	octets
0x0→0x7F	`abc defg`	`0abc defg`	1
0x80→0x7FF	`abc defg hijk`	`110a bcde 10fg hijk`	2
0x800→0xFFFF	`abcd efgh ijkl mnop`	`1110 abcd 10ef ghij 10kl mnop`	3
0x10000→0x1FFFFF	`a bcde fghi jklm nopq rstu`	`1111 0abc 10de fghi 10jk lmno 10pq rstu`	4

- D'autres encodages existent :
  - UCS-2 codage partiel sur 2 octets par caractères
    représente les 2^{16} premiers caractères
  - UTF-16 un codage plus simple qu'UTF-8 utilisant 2 ou 4 octets par caractères
    2 pour les premiers, 4 pour les autres (10 octets par paire de 2 octets).
- Avantage de l'UTF-8 : économise en place pour l'ASCII (1 octet par caractère)
- Inconvénient de l'UTF-8 : impossible de dire facilement combien il y a de caractères

Unicode et UTF-8

- Unicode est une collection de plus de 100 000 caractères
  - UCS : *Universal Character Set*
  - ne spécifie pas la façon de représenter par une séquence d'octets
  - taille maximale est de 17×2^{16}
  - le code maximal 0x10FFFF
- UTF-8 est une façon de transformer un numéro en une séquence d'octets
  - UTF-8 : *UCS Transformation Format 8 bits*

Valeurs	Écriture binaire	Codage UTF-8 (binaire)	octets
0x0→0x7F	`abc defg`	`0abc defg`	1
0x80→0x7FF	`abc defg hijk`	`110a bcde 10fg hijk`	2
0x800→0xFFFF	`abcd efgh ijkl mnop`	`1110 abcd 10ef ghij 10kl mnop`	3
0x10000→0x1FFFFF	`a bcde fghi jklm nopq rstu`	`1111 0abc 10de fghi 10jk lmno 10pq rstu`	4

- D'autres encodages existent :
  - UCS-2 codage partiel sur 2 octets par caractères
    représente les 2^{16} premiers caractères
  - UTF-16 un codage plus simple qu'UTF-8 utilisant 2 ou 4 octets par caractères
    2 pour les premiers, 4 pour les autres (10 octets par paire de 2 octets).
- Avantage de l'UTF-8 : économe en place pour l'ASCII (1 octet par caractère)
- Inconvénient de l'UTF-8 : impossible de dire facilement à quel octet est le n^e caractère.

Unicode et UTF-8

- Unicode est une collection de plus de 100 000 caractères
  - UCS : *Universal Character Set*
  - ne spécifie pas la façon de représenter par une séquence d'octets
  - taille maximale est de 17×2^{16}
  - le code maximal 0x10FFFF
- UTF-8 est une façon de transformer un numéro en une séquence d'octets
  - UTF-8 : *UCS Transformation Format 8 bits*

Valeurs	Écriture binaire	Codage UTF-8 (binaire)	octets
0x0→0x7F	`abc defg`	`0abc defg`	1
0x80→0x7FF	`abc defg hijk`	`110a bcde 10fg hijk`	2
0x800→0xFFFF	`abcd efgh ijkl mnop`	`1110 abcd 10ef ghij 10kl mnop`	3
0x10000→0x1FFFFF	`a bcde fghi jklm nopq rstu`	`1111 0abc 10de fghi 10jk lmno 10pq rstu`	4

- D'autres encodages existent :
  - UCS-2 codage partiel sur 2 octets par caractères
    représente les 2^{16} premiers caractères
  - UTF-16 un codage plus simple qu'UTF-8 utilisant 2 ou 4 octets par caractères
    2 pour les premiers, 4 pour les autres (10 octets par paire de 2 octets).
- Avantage de l'UTF-8 : économe en place pour l'ASCII (1 octet par caractère)
- Inconvénient de l'UTF-8 : impossible de dire facilement à quel octet est le n^e caractère.

Unicode et UTF-8

- Unicode est une collection de plus de 100 000 caractères
  - UCS : *Universal Character Set*
  - ne spécifie pas la façon de représenter par une séquence d'octets
  - taille maximale est de 17×2^{16}
  - le code maximal 0x10FFFF
- UTF-8 est une façon de transformer un numéro en une séquence d'octets
  - UTF-8 : *UCS Transformation Format 8 bits*

Valeurs	Écriture binaire	Codage UTF-8 (binaire)	octets
0x0→0x7F	`abc defg`	`0abc defg`	1
0x80→0x7FF	`abc defg hijk`	`110a bcde 10fg hijk`	2
0x800→0xFFFF	`abcd efgh ijkl mnop`	`1110 abcd 10ef ghij 10kl mnop`	3
0x10000→0x1FFFFF	`a bcde fghi jklm nopq rstu`	`1111 0abc 10de fghi 10jk lmno 10pq rstu`	4

- D'autres encodages existent :
  - UCS-2 codage partiel sur 2 octets par caractères
    représente les 2^{16} premiers caractères
  - UTF-16 un codage plus simple qu'UTF-8 utilisant 2 ou 4 octets par caractères
    2 pour les premiers, 4 pour les autres (10 octets par paire de 2 octets).
- Avantage de l'UTF-8 : économe en place pour l'ASCII (1 octet par caractère)
- Inconvénient de l'UTF-8 : impossible de dire facilement à quel octet est le n^e caractère.

Les chaînes avec longueur spécifiée

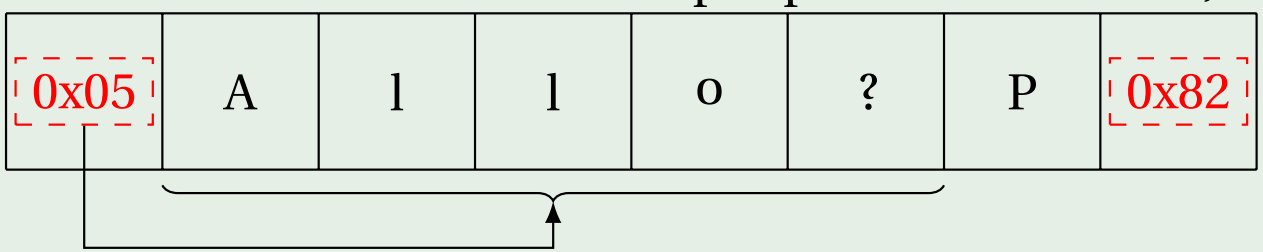
- chaînes de caractères = listes ordonnées de caractères
- En mémoire elle occupe plusieurs positions consécutives
- On désigne souvent la chaîne par la première position occupée

Certains langages résolvent le problème de savoir où la chaîne s'arrête en stockant aussi la longueur.

- Inconvénient : problème avec certains codages/jeux de caractères pour trouver le n^e élément d'une chaîne (et en particulier, la longueur en nombre de caractères).
- Avantage : le calcul de la place mémoire occupée est instantané.

Exemple

On stocke ici la chaîne « Allo ? » (le P et la valeur 0x82 sont des éléments qui sont dans la mémoire mais ne font pas partie de la chaîne).



Les chaînes avec longueur spécifiée

- chaînes de caractères = listes ordonnées de caractères
- En mémoire elle occupe plusieurs positions consécutives
- On désigne souvent la chaîne par la première position occupée

Certains langages résolvent le problème de savoir où la chaîne s'arrête en stockant aussi la longueur.

- Inconvénient : problème avec certains codages/jeux de caractères pour trouver le n^e élément d'une chaîne (et en particulier, la longueur en nombre de caractères).
- Avantage : le calcul de la place mémoire occupée est instantané.

Est-ce que la longueur est en caractères ou en octets ?

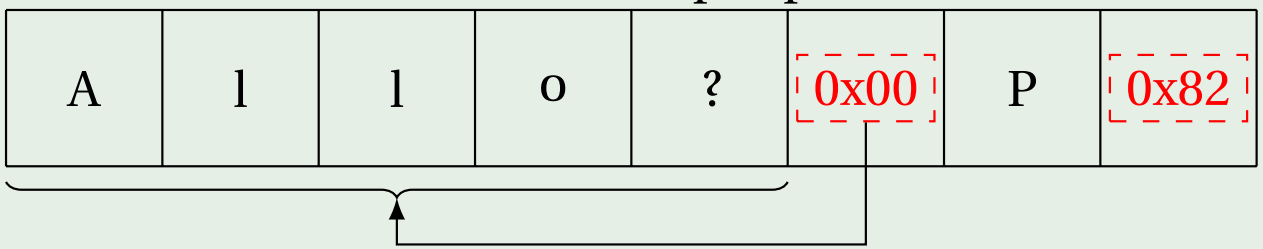
En **octets**, le plus souvent, ou les deux. Le plus important est de savoir trouver la fin de la chaîne (pour pouvoir la copier).

Les chaînes avec marqueurs de fin

Une autre possibilité est de marquer la fin de la chaîne avec un octet particulier ou une séquence d'octets particulière. C'est le cas du langage C (et de beaucoup d'autres langages dérivés) qui utilise le caractère nul.

Exemple

On stocke ici la chaîne « Allo ? » (le P et la valeur 0x82 sont des éléments qui sont dans la mémoire mais ne font pas partie de la chaîne).



Est-ce que le marqueur fait partie de la chaîne ?

En pratique, oui. Mais il ne fait pas partie du texte codé par la chaîne.
À l'intérieur d'un langage il n'y a en général qu'une seule sorte de chaîne.

Le problème de l'échappement

La fin de chaîne

- Quand la longueur n'est pas spécifiée à côté d'une chaîne, la fin de la chaîne est forcément indiquée par une séquence spécifique de bits.
- S'il existe une séquence spécifique invalide dans le codage pour la représentation de caractères, alors on peut la choisir comme représentant la fin de chaîne.
- Sinon, il faut choisir un caractère qui va coder la fin de la chaîne

Comment coder une chaîne qui comporte ce caractère ?

Les séquences signifiantes

Parfois, on veut pouvoir utiliser dans des chaînes des séquences qui ont un sens spécial. Par exemple, on pourrait vouloir que `0x0F03` représente le caractère ་ qu'on ne peut pas rentrer facilement au clavier. Mais dans ce cas, comment écrire la chaîne `0x0F03` (comme par exemple pour la phrase « Si on met `0x0F03` dans une chaîne on obtient le caractère ་ » ?

Il faut donc utiliser une procédure d'échappement !

Les séquences d'échappement

- On utilise une séquence (parfois codante) d'échappement qui permet de modifier le sens des caractères qui suivent
- Si la séquence d'échappement est codante, on doit prévoir au moins une combinaison qui permet de redonner le caractère d'échappement
- Avoir des chaînes interprétables complique énormément les opérations élémentaires, comme calculer le nombre de caractères dans la chaîne, ou savoir si un caractère est présent dans la chaîne.

On se retrouve souvent à « empiler » les modes d'échappement identiques ou différents.

Les séquences d'échappement

Exemple

En langage C et dérivés, le caractère \ est utilisé pour introduire des séquences d'échappement.

\0 ..... le caractère nul.
\n ..... le caractère 10 (NL).
\t ..... le caractère 9 (TAB).
\xxx ..... le caractère de numéro octal xxx.
\Uxxxx ..... le caractère unicode de numéro hexadécimal xxxx.

Ce caractère unicode peut représenter plusieurs octets. Le codage choisi dépend du compilateur et du type de la chaîne.