In [1]:

```
Ce cours a été régénéré le 2018-10-11 10:22:54.476395
```

Les textes

De l'écrit aux nombres

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*).

On a assemblé les caractères qui étaient nécessaires pour écrire une langue en « alphabets » qui comprenaient lettres, chiffres, caractères de ponctuation et autres symboles techniques. Ces ensemble s'appellent des *jeux de caractères*.

Cette technique descend des techniques inventées par les Chinois (IX^{e} siècle) et les Européens (Gutenberg, XV^{e} siècle). Les jeux de caractères étaient alors physiquement des collections d'éléments mobiles (en bois, en terre cuite, en plomb...) qui étaient rangés dans une caisse (appellée *casse*) et servaient à transférer de l'encre sur du papier.

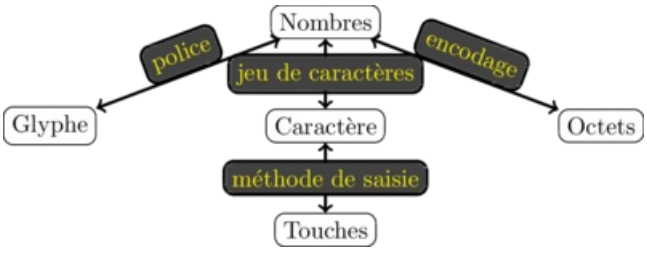
De nos jours, les caractères ne sont pas rangés dans une caisse, ils sont contenus dans une sorte de programme informatique spécialisé qu'on appelle une police (anglais : *font*) et qui correspond à la dénomination ancienne en imprimerie.

Les raisons historiques font qu'on a eu des jeux de caractères différents selon les régions et les fabricants. Depuis les années 2000, un jeu de caractères qui contient tous les autres s'impose lentement : le jeu de caractère Unicode (https://fr.wikipedia.org/wiki/Unicode).

Les glyphes sont les dessins des lettres, différents selon les polices. Une même police de caractère peut supporter plusieurs jeux de caractères (tant qu'il sait faire l'association caractère → glyphe), le dessin proprement dit n'est stocké qu'une seule fois.

Parce que les ordinateurs manient très bien les nombre, un texte est donc représenté par des nombres dont les valeurs sont fixées par le jeu de caractères. Toutefois, selon ces jeux de caractères, les nombres peuvent atteindre de grandes valeurs. Lorsque le nombre de caractères possibles ne dépasse pas 256, on utilise un octet pour chaque caractère ; mais sinon, on doit recourir à un *encodage* plus compliqué qui va permettre de transformer le nombre en un ou plusieurs octets. Cette table doit permettre évidemment d'aller dans un sens, et dans l'autre.

In [2]:



Les ligatures

Les ligatures sont une caractéristique de l'écriture qui vient de l'écriture manuscrite. Cela consiste à modifier l'écriture d'une paire (ou plus) de lettres en fonction des lettres voisines.

Certaines ligatures sont passées dans la langue au point que les lettres liées forment une nouvelle lettre à part entière (et où les lettres détachées existent indépendamment des lettres liées). On peut penser par exemple au œ en français, au ij en néerlandais, au ch espagnol (qui est trié différemment : il est entre le C et le D, et non pas entre CE et CI). Ces caractères ont en général leur place dans le jeu de caractères de façon autonome.

D'autres ligatures sont des ligatures dites esthétiques : la langue arabe par exemple modifie fortement le dessin des lettres en fonction des celles qui sont devant et derrières. La représentation de ces langues se fait en général par les caractères élémentaires, la tâche de la ligature étant reportée sur la police.

> **Exemple :** La lettre Œ (ligature linguistique) est différente de OE . La lettre fi représente deux caractères, avec affichage fi (ligature esthétique pour éviter le fi) où le f rencontre le i. En arabe ou sanskrit, la ligature est obligatoire mais esthétique : تونس est la forme liée de ت و ن س (se prononce Tounisse, c'est la ville de Tunis).

Jeux de caractères et encodage

Au début était un certain nombre de jeux de caractères sur 7 ou 8 bits par caractères. Un seul a vraiment survécu : l'ASCII (https://fr.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange).

Il en existe donc d'autres, principalement des jeux de caractères *régionaux* :

- Normes ISO-8859 (https://fr.wikipedia.org/wiki/ISO_8859) qui compte une quinzaine de jeux de caractères différents qui couvrent à eux tous les langues européennes, l'arabe, le grec, le russe, l'hébreu, le turc.
- L'Unicode (https://fr.wikipedia.org/wiki/Unicode) qui présente de nombreux caractères (http://hapax.qc.ca/Tableaux-10.0.htm)

La table ASCII

Le jeu de caractère ASCII comporte 128 caractères et pouvait être codé sur 7 bits par caractère. On l'utilise maintenant exclusivement sur 8 bits (avec des numéros inutilisés). Les caractères entre 0 et 31 sont des *caractères de contrôle* qui avaient des effets sur les terminaux mais n'avaient pas de forme, de 32 à 126 les *caractères affichables* (que vous connaissez).

N°	**?**	**N°**	**?**	**N°**	**?**	**N°**	**?**	**N°**	**?**	**N°**	**?**	**N°**	**?**	**N°**	**?**
00	NUL	01	SOH	02	STX	03	ETX	04	EOT	05	ENQ	06	ACK	07	BEL
08	BS	09	HT	0A	NL	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

In [3]:

Activité : La table ASCII

Quel caractère correspond au code 0x4e ? > _____

Quel caractère correspond au code 0x6e ? > _____

À quoi sert le caractère ESC ?

Votre proposition > _____

Quel est le nom qu'on donne pour le retour chariot ? > _____

Jeux de caractères et codage

Les jeux de caractères étant des ensembles numérotés de caractères, il faut pouvoir associer à chaque nombre-caractère une séquence d'octets qui permet de le reconnaître.

Le problème est simple si on a un nombre de caractères réduits (moins de 255). Mais si on en a plus, il faut trouver une association.

L'association simple en mettant plus d'octets pose rapidement le problème de la taille des textes (notamment si certains caractères sont très fréquents et d'autres rares, mais existants). Cette transformation s'appelle *l'encodage*.

Les encodages existants sont de plusieurs sortes :

- l'encodage banal qui écrit le numéro sur un octet, réservé aux jeux de caractères à moins de 255 caractères (ISO-8859, KOI8R). Inutilisable pour les langues asiatiques.
- les encodages banals sur plusieurs octets (UCS2, UCS4, Big5)
- les encodages à décalage où certaines suites d'octets fixent une valeur de base B et où les autres octets n fixent la valeur finale qui est $B + n$
- les encodages à nombre d'octets variables (UTF-8)

Ces problèmes d'encodage n'ont pas été évidents tout de suite, et du coup, l'encodage et le jeu de caractères n'ont pas été tout de suite des données bien intégrées. Les simples fichiers textes par exemple ne retiennent ni l'un ni l'autre, et les valeurs sont devinées par des algorithmes... qui peuvent échouer.

Cet oubli, ou la mauvaise coordination entre plusieurs logiciels (qui ignorent les spécifications données par les autres), a conduit à multiplié les exemples de mauvais encodages visibles par les utilisateurs. Ce phénomène a plusieurs noms, mais *mojibake* (le nom japonais) est un peu plus répandu : krakozyabry en russe par exemple. On dit parfois *hieroglyphes* en français.

Par exemple, deux personnes situées en France et en Russie correspondaient et R a donné à F son adresse pour se faire envoyer un paquet. Mais R a envoyé son adresse en KOI8R, et le logiciel de F l'a deviné (faussement) comme étant de l'ISO-8859-1... et a donc affiché les mauvais caractères. Le postier a réussi à faire la transformation inverse, ce qui témoigne à quel point ce genre de problèmes est courant.

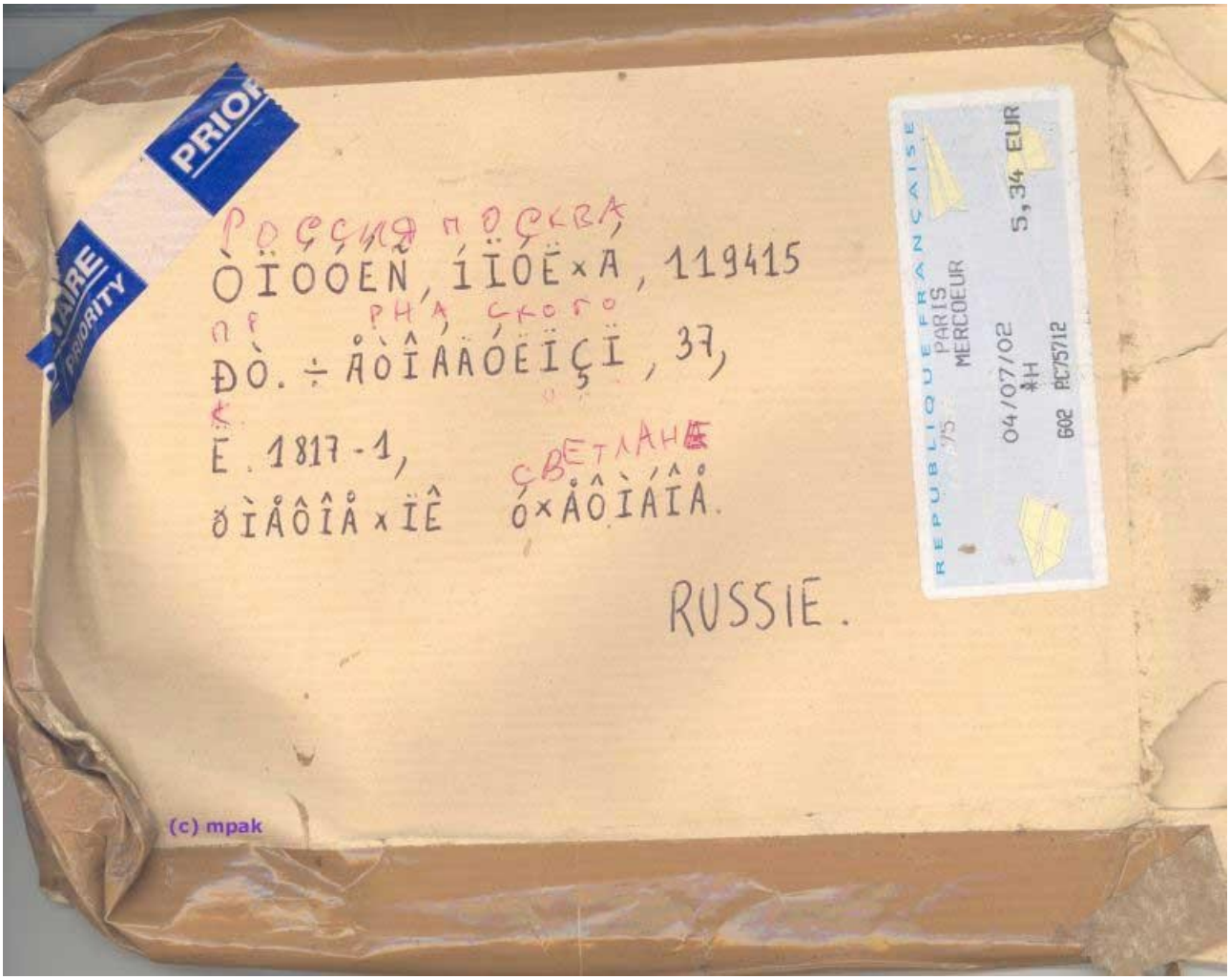
```
Россия Москва, 119415
пр.Вернадского, 37,
к.1817-1,
Плетневой Светлане
```

Les octets sont les suivants:

- **000** `cff2 d3d3 d1c9 ed20 d3cf d7cb 2cc1 3120`
- **010** `3931 3134 0a35 d2d0 f72e d2c5 c1ce d3c4`
- **020** `cfcb cfc7 202c 3733 0a2c 2ecb 3831 3731`
- **030** `312d 0a2c ccf0 d4c5 c5ce cfd7 20ca d7f3`
- **040** `d4c5 c1cc c5ce`

et en ISO-8859-1 ça s'affiche :

```
òÏÓÓÉÑ íÏÓË×Á, 119415
ÐÒ.÷ÅÒÎÁÄÓËÉÇÏ, 37,
Ë.1817-1,
ðÌÅÔÎÅ×ÏÊ ó×ÅÔÌÁÎÅ
```



In [4]:

Activité : Ligatures

Les filles actent selon leur cœur &
les garçons selon leur souffle.

Dans le texte ci-dessus, quelles sont les ligatures linguistiques que vous identifiez ?

Votre proposition > _____

Dans le texte ci-dessus, quelles sont les ligatures esthétiques que vous identifiez ?

Votre proposition > _____

In [5]:

Activité : Encodage et hieroglyphes

Est-il possible de représenter le texte précédent dans le jeu de caractères ASCII [oui/non] >

_____

Dans le jeu de caractères *latin9* (ISO-8859-15), il est possible de coder ce texte. Mathilde envoie ce texte à Ružar, un correspondant à Malte. Quand Ružar ouvre son mail, il lit : « Les filles actent selon leur c½ur & les garçons selon leur souffle. ». Expliquer pourquoi.

Votre proposition > _____

Unicode et UTF-8

Depuis les années 2000, les jeux de caractères qui ne sont ni ASCII, ni ISO-8850-1 se raréfient au profit d'Unicode, en particulier son encodage par UTF-8 et UTF-16. Unicode est une collection de plus de 100000 caractères qui ne spécifie pas la façon de le représenter par une séquence d'octets. La taille maximale est de 17×2^{16} et le code maximal 0x10FFFF. UTF-8 est une façon de transformer un numéro en une séquence d'octets. La norme qui correspond est formellement « ISO/CEI 10646:*année* », la dernière en vigueur est ISO/CEI 10646:2012, aussi appellée Unicode 6.1.

Dans l'encodage UTF-8, la conversion se fait à l'aide de cette table :

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

Il existe d'autres encodages pour Unicode : UCS-2 (partiel, ne couvre que 2^{16} caractères), UCS-4 qui sont des encodages à taille fixe multi-octets, UTF-16 (qui permet de représenter tout le premier plan de base en 2 octets, et le reste avec 4 octets).

Dans le jeu de caractère Unicode (c'est indépendant de l'encodage), il existe plusieurs façons de représenter les mêmes glyphes. Par exemple, on peut donner le caractère « é » et le caractère « e » et «◌́ » (qui est l'accent aigu combinant). Si on met ces deux derniers l'un derrière l'autre on obtient « é » qui sont deux caractères mais affichés avec un seul glyphe.

La notion de normalisation (combinaison maximale : on combine pour obtenir les caractères pré-combinés ou combinaison minimale : on combine pour obtenir le maximum de caractères indépendants) permet de comparer facilement deux textes et de savoir s'ils représentent la même chose. Toutefois, cela ne permet pas de distinguer des caractères différents qui par hasard auraient le même glyphe (A, A et A par exemple sont trois caractères différents: la première lettre des alphabets latin, cyrillique et grec).

In [6]:

Le caractère LATIN SMALL LETTER B (bloc Basic Latin) a pour numéro 00062 : : b

(https://www.fileformat.info/info/unicode/char/0062/brow

Par combien d'octets va-t-il être représenté ? > _____

Par quels octets va-t-il être représenté (majuscules et espace tous les deux caractères) ? >

_____

Le caractère GREEK CAPITAL LETTER PSI (bloc Greek and Coptic) a pour numéro 003A8 : : Ψ

(https://www.fileformat.info/info/unicode/char/03a8/brow

Par combien d'octets va-t-il être représenté ? > _____

Par quels octets va-t-il être représenté (majuscules et espace tous les deux caractères) ? >

_____

Le caractère CJK UNIFIED IDEOGRAPH-4433 (bloc CJK Unified Ideographs Extension A) a pour numéro 04433 : : 腳

(https://www.fileformat.info/info/unicode/char/4433/brow

Par combien d'octets va-t-il être représenté ? > _____

Par quels octets va-t-il être représenté (majuscules et espace tous les deux caractères) ? >

_____

Le caractère MATHEMATICAL BOLD CAPITAL M (bloc Mathematical Alphanumeric Symbols) a pour numéro 1D40C : : 𝐌

(https://www.fileformat.info/info/unicode/char/1d40c/brov

Par combien d'octets va-t-il être représenté ? > _____

Par quels octets va-t-il être représenté (majuscules et espace tous les deux caractères) ? >

_____

Le caractère représenté par D5 89 est : Ձ

(https://www.fileformat.info/info/unicode/char/0549/brow

Quel est son numéro en hexadécimal ? > _____

La chaîne de caractères est représentée par EB B7 9A EB 8C A6 ED 85 B0 en UTF-8 : Café

Combien de caractères sont représentés ? > _____

La chaîne de caractères est représentée par 43 CC A7 61 en UTF-8 : 

Combien de caractères sont représentés ? > _____

La chaîne de caractères est représentée par C3 87 61 en UTF-8 : 

Combien de caractères sont représentés ? > _____

In [7]:

Activité : Taille de texte et Unicode

La plupart des langages recouraient à des jeux de caractères régionaux avant l'arrivée d'Unicode. Ceux-ci prenaient un octet par caractère, sauf pour les langues asiatiques (chinois par exemple : deux octets par caractère).

Le passage à Unicode a créé une augmentation de la taille des textes selon le nombre de caractères hors de la grille ASCII. Pour différents langages, calculer le taux moyen d'augmentation des textes en fonction de la répartion.

Il est possible par exemple de calculer la longueur en octets d'un texte en moyenne en UTF-8 et de le comparer à la taille dans le jeu de caractères régional.

- Anglais
  - 100% des caractères occupent 1 octets
- Français
  - 97% des caractères occupent 1 octets
  - 3% des caractères occupent 2 octets
- Russe (codage régional KOI8R, 1 o/caractère)
  - 5% des caractères occupent 1 octets
  - 95% des caractères occupent 2 octets
- Chinois (codage régional BIG5, 2 o/caractère)
  - 1% des caractères occupent 1 octets
  - 99% des caractères occupent 3 octets

Quelle est la taille d´un texte de 1000 caractères en Anglais ? > _____

Quel rapport d´augmentation en Anglais ? > _____

Quelle est la taille d´un texte de 1000 caractères en Français ? > _____

Quel rapport d´augmentation en Français ? > _____

Quelle est la taille d´un texte de 1000 caractères en Russe ? > _____

Quel rapport d´augmentation en Russe ? > _____

Quelle est la taille d´un texte de 1000 caractères en Chinois ? > _____

Quel rapport d´augmentation en Chinois ? > _____

Chaînes de caractères

Les chaînes de caractères sont des listes ordonnées de caractères.

Lorsqu'une chaîne de caractères est stockée en mémoire, elle occupe plusieurs positions consécutives dans la mémoire. On désigne souvent la chaîne par la première position occupée.

Codage longueur-données

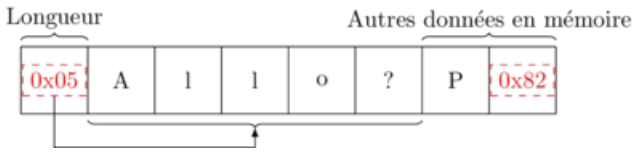
Certains langages résolvent le problème de savoir où la chaîne s'arrête en stockant aussi la longueur.

Problème avec certains codages/jeux de caractères pour trouver le énième élément d'une chaîne (et en particulier, la longueur en nombre de caractères).

Avantage: le calcul de la place mémoire occupée est instantané.

> **Remarque :** *Est-ce que la longueur est en caractères ou en octets ?* En octets, parfois les deux : le plus important est de savoir trouver la fin de la chaîne (pour pouvoir la copier d'un endroit à un autre de la mémoire. \end{block}
>
> **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).

In [8]:



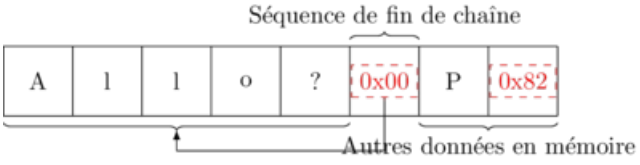
Caractère terminal

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.

À l'intérieur d'un langage il n'y a en général qu'une seule sorte de chaîne.

> **Remarque :** *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.
>
> **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).

In [9]:



L'échappement

Premier problème, le choix de la séquence de fin.

1. 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.
2. 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.
3. Sinon, il faut choisir un caractère qui va coder la fin de la chaîne.
4. Comment coder une chaîne qui comporte ce caractère ?

Deuxième problème, 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 !

\item On utilise une séquence (parfois codante) d'échappement qui permet de modifier le sens des caractères qui suivent \item Si la séquence d'échappement est codante, on doit prévoir au moins une combinaison qui permet de redonner le caractère d'échappement \item 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. \item[\dialogwarning] On se retrouve souvent à «~empiler~» les modes d'échappement identiques ou différents. \end{itemize}

> **Exemple :** En langage C et d'autres (Python par exemple), le caractère \ est utilisé pour introduire des séquences d'échappement. `\0` est le caractère nul, `\n` est le caractère de nouvelle ligne, `\t` est le caractère de tabulation, `\xxx` est le caractère de numéro octal *xxx*...

In [10]:

Activité : Échappements en C

Dessinez la structure en mémoire ds chaînes C suivantes, et montrez comment elles s'affichent sur un terminal.

- `Toto`

0	**1**	**2**	**3**	**4**
T	o	t	o	\0

Affichage:
`Toto`

- `Bonjour␣!\n`

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
B	o	n	j	o	u	r	␣	!	\n	\0

Affichage:
`Bonjour !`

- `Un␣philosophe\n\tparle\ndu␣temps.\n`

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
U	n	␣	p	h	i	l	o	s	o	p	h	e	\n	\t	p	a	r	l	e	\n	d	u	␣	t	e	m	p	s	.	\n	\0

Affichage:
```
Un philosophe
    parle
du temps.
```

- `Cr\303\250me`

0	**1**	**2**	**3**	**4**	**5**	**6**
C	r	\303	\250	m	e	\0

Affichage:
`Crème`

Attention, l´affichage ci-dessus dépend du terminal ! Ici on a choisi UTF-8

`In [11]:`

Activité : Échappements en C — la vengeance

Une bizarrerie historique du C/C++ fait que certaines séquences sont remplacées avant que le programme ne soit exécuté par d'autres caractères:

Trigraphe	**Résultat**
??(	[
??)	]
??<	{
??>	}
??=	#
??/	\
??'	^
??!	|
??-	~

Dessinez la structure en mémoire ds chaînes C suivantes telles qu'elles sont dans la mémoire du programme exécuté, et montrez comment elles s'affichent sur un terminal.

- `Hello??!`

0	**1**	**2**	**3**	**4**	**5**	**6**	
H	e	l	l	o			\0

Affichage:
`Hello|`

- `Bye??/n`

0	**1**	**2**	**3**	**4**
B	y	e	\n	\0

Affichage:
Bye

Quels sont les problèmes qui pourraient arriver avec ce système tel qu'il est décrit ?

Votre proposition > _____

`In [12]:`

`Mode interactif`