
Unicode

AB – v2.18 (04/09/2021)

Historique

ASCII

- Représentation des caractères sur 7 bits
- Caractères 0 à 31 et 127 non imprimables ou affichables
- 1 caractère = 1 octet

ASCII Code Chart																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Extensions

- Utilisation du 8^e bit
- Les 128 nouveaux caractères dépendent de la page de code utilisées :
 - 437
 - 850 (Europe occidentale)
 - Pages normalisées désormais par l'ISO :
 - ISO 8859-1 (ou Latin 1)
 - ISO 8859-15
- On a toujours 1 caractère = 1 octet

Exemple de différences entre les pages de code 437 et 850

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8x	Ç	ü	é	â	ä	à	á	ç	ê	ë	è	ï	î	í	Ä	Å
9x	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₣	ƒ
Ax	á	í	ó	ú	ñ	Ñ	ª	º	¿	¬	½	¼	¡	«	»	
Bx	⌘	⌘	⌘													
Cx	L	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
Dx	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
Ex	α	β	Γ	Π	Σ	σ	μ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
Fx	≡	±	≥	≤			÷	≈	°	.	.	√	n	²	■	NBSP

Page de code 437

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8x	Ç	ü	é	â	ä	à	á	ç	ê	ë	è	ï	î	í	Ä	Å
9x	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	ø	£	Ø	×	ƒ
Ax	á	í	ó	ú	ñ	Ñ	ª	º	¿	®	¬	½	¼	¡	«	»
Bx	⌘	⌘	⌘			Á	Â	À	©	⌘	⌘	⌘	⌘	¢	¥	₣
Cx	L	⊥	⊥	⊥	⊥	ã	Ä	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
Dx	ø	Ð	Ê	Ë	È	Ì	Í	Î	⌘	⌘	⊥	⊥	⊥	⊥	⊥	⊥
Ex	Ó	β	Ô	Ò	ö	Õ	μ	þ	þ	Ú	Û	Ü	ý	Ý	—	·
Fx	SHY	±	=	¾	¶	§	÷	,	°	”	.	¹	³	²	■	NBSP

Page de code 850

Norme Unicode

Unicode

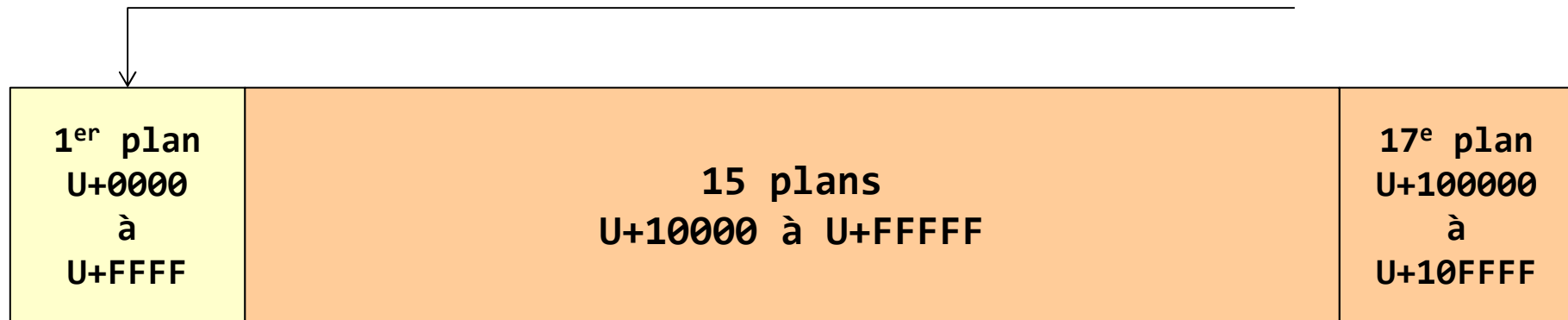
- **Standard** informatique développé par le « Unicode Consortium » à partir de 1991 en remplacement des pages de code
- Compatible avec la **norme** ISO-10646 qui définit l'UCS (*Universal Character Set*)
- Permet de représenter n'importe quel caractère de tous les systèmes d'écriture (109 000 caractères couvrant 93 écritures)
- Unicode sépare la définition du jeu de caractères et son encodage

Jeu de caractères codés (*Coded Character Set*)

Caractère : É (Lettre majuscule latine e accent aigu)

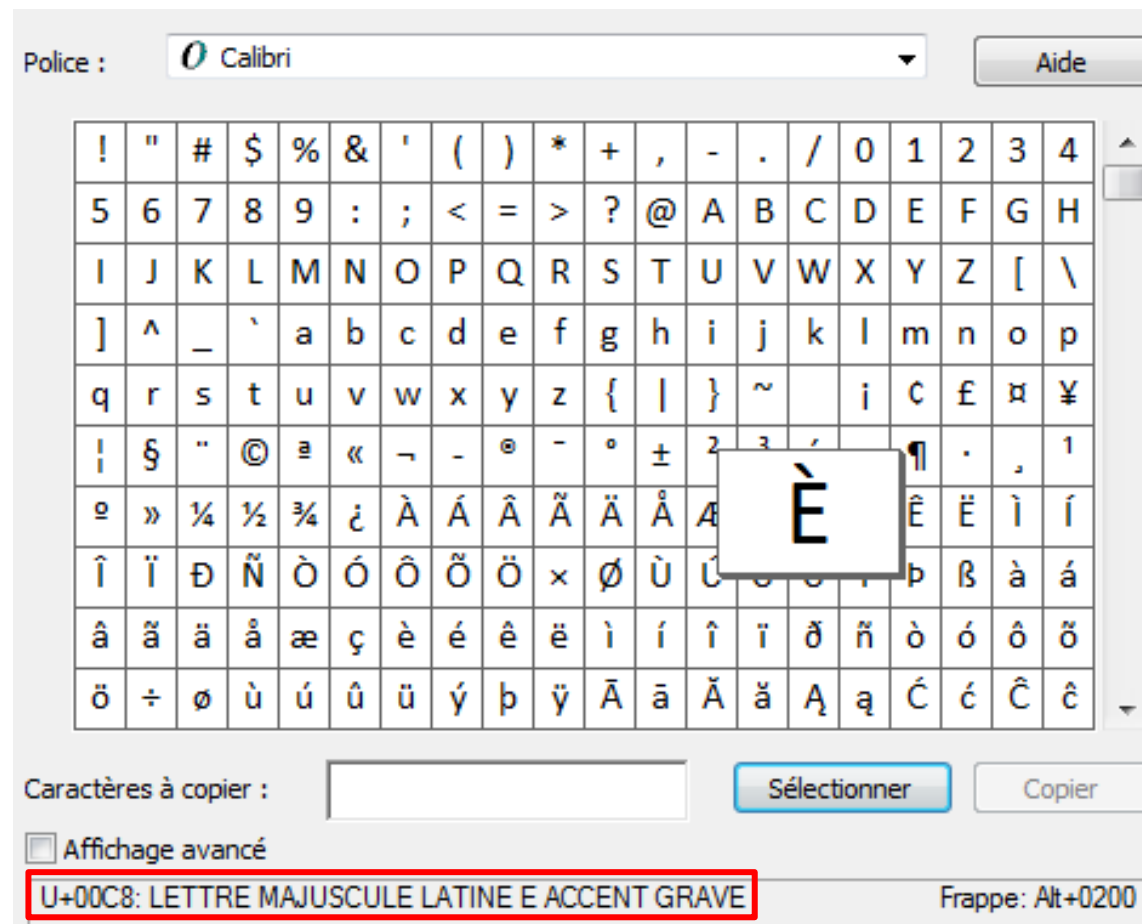


Point de code (*code point*) : de 0 à 0x10FFFF (17x65535) → U+00C9



Plan multilingue de base - *Basic Multilingual Plane (BMP)*
(principaux caractères)

Table des caractères Windows (charmap.exe)



Encodage



Formalisme de codage des caractères

(Character Encoding Form)

- Codage du numéro de caractère (U+XXXXX) en octets
- La taille d'un caractère codé dépend de l'encodage choisi (de 8 à 21 bits par caractère)

UTF-X / UCS-Y

- **UCS : *Universal Character Set***
 - UCS-Y : nombre d'octets pour le codage d'un caractère (taille fixe)
 - UCS-2, UCS-4
- **UTF : *Unicode Transformation Format***
 - UTF-X : nombre minimal de bits pour le codage d'un caractère (taille variable)
 - UTF-8, UTF-16, UTF-32

UTF-8

RFC 3629 (11/2003)

- La taille d'un caractère encodé est variable (de 1 à 4 octets) :
 - réduction de la taille occupée (surtout pour caractères occidentaux)
 - coût pour le traitement de chaînes de caractères
- Compatible avec l'ASCII sur les caractères 0-127
- Tous les caractères du BMP (*Basic Multilingual Plane*) sont représentés au maximum sur 3 octets

UTF-8

Char. number range (hexadecimal)	UTF-8 octet sequence (binary)
-----+-----	
0000 0000 - 0000 007F	0xxxxxxx
0000 0080 - 0000 07FF	110xxxxx 10xxxxxx
0000 0800 - 0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
0001 0000 - 0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

A : U+0041 -> 1000001 -> 01000001
É : U+00E9 -> 00011 101001 -> 11000011 10101001
山 : U+5C71 -> 0101 110001 110001 -> 11100101 10110001 10110001
☺ : U+1F60A -> 11111 011000 001010
-> 11110000 10011111 10011000 10001010
(F0 9F 98 8A)

UTF-16

- Codage d'un caractère sur **16** ou **32** bits
- Les caractères du BMP (U+0000 à U+D7FF et U+E000 à U+FFFD) sont tous représentés sur **16 bits**
- Les autres plans (*supplementary planes*), qui contiennent les *supplementary characters*, sont représentés sur **2x16 bits** au moyen d'une *surrogate pair*
- Dans le BMP, les points de code U+D800 à U+DFFF sont réservés à l'encodage des *supplementary characters* :
 - U+D800 à U+DBFF : *high surrogates*
 - U+DC00 à U+DFFF : *low surrogates*

} *Surrogate pair*
(2x 10 bits)

UTF-16 : principe d'encodage

- Caractères hors du BMP :
 - U+10 000 à U+10 FFFF → 16 x (0xFFFF) soit 20 bits
- 2 x (10 bits) → 2 x (0 à 0x3FF) soit X et Y
 - Fort : 0xD800 + X → 0xD800 à 0xDBFF (10 bits)
 - Faible : 0xDC00 + Y → 0xDC00 à 0xDFFF (10 bits)
- Rappels :
 - $2^{10} = 1024 = 0x400$
 - U+D800 à U+DBFF : *high surrogates* (10 bits)
 - U+DC00 à U+DFFF : *low surrogates* (10 bits)

UTF-16

- Dans BMP :

- A : U+0041 -> 00 41

- É : U+00E9 -> 00 E9

- 山 : U+5C71 -> 5C 71

- Hors BMP :

- ☺ : U+1F60A -> F60A -> 1111011000001010

0000111101 (3D) 1000001010 (20A)

D800+3D = D83D

DC00+20A = DE0A

D8 3D DE 0A

Police de caractère

- Représentation d'un point de code par un glyphe
- En général, les polices n'affichent qu'une partie des points de code
- Si le glyphe du caractère n'est pas disponible dans la police, le caractère de substitution est affiché (U+FFFD)
 - 

Mécanisme de sérialisation des caractères (*Character Encoding Scheme*)

- Transformation des octets résultant de l'encodage (suites d'unités de codage) en suite d'octets
- Hormis en UTF-8, l'ordre est important
- Définition explicite :
 - charset=UTF-8
 - charset=UTF-16BE
- *Byte order mark* (BOM)

Codage	BOM
UTF-8	EF BB BF
UTF-16 Big Endian	FE FF
UTF-16 Little Endian	FF FE
UTF-32 Big Endian	00 00 FE FF
UTF-32 Little Endian	FF FE 00 00

Mise en œuvre

- Le codage UCS-2/UTF-16 est utilisé nativement par les systèmes « modernes » (Windows NT, Java, Qt, OS/X) mais peut-être limité à 16 bits
- Grand nombre d'octets NULL dans les textes contenant les caractères ASCII
- L'encodage des plans supplémentaires (hors BMP) est souvent problématique
 - UTF-16 est supporté depuis Windows 2000
 - Cf. API Uniscribe

Programmation sous Windows

Définition des types C/C++

- **char** : caractère sur 8 bits (1 octet)
- **wchar_t** : caractère Unicode (2 octets)

int printf(const char **format* [, *argument*]...);

int wprintf(const wchar_t **format* [, *argument*]...);

- char ascii = "Ceci est du texte";
- wchar_t unicode = L"Ceci est du texte";

Définition des types Windows

- typedef char **CHAR**;
- typedef wchar_t **WCHAR**;

- typedef CHAR ***PCHAR**;
- typedef CHAR ***PSTR**;
- typedef CHAR ***LPSTR**;

- typedef WCHAR ***PWCHAR**;
- typedef WCHAR ***PWSTR**;
- typedef WCHAR ***LPWSTR**;

Définition des types Windows

- `typedef CONST CHAR *PCSTR;`
- `typedef CONST CHAR *LPCSTR;`
- `typedef CONST WCHAR *PCWSTR;`
- `typedef CONST WCHAR *LPCWSTR;`

Fonctions Windows

```
int
WINAPI
MessageBoxA(
    _In_opt_ HWND hWnd,
    _In_opt_ LPCSTR lpText,
    _In_opt_ LPCSTR lpCaption,
    _In_ UINT uType
);
```

```
int
WINAPI
MessageBoxW(
    _In_opt_ HWND hWnd,
    _In_opt_ LPCWSTR lpText,
    _In_opt_ LPCWSTR lpCaption,
    _In_ UINT uType
);
```

TCHAR

```
#ifndef UNICODE
    typedef WCHAR TCHAR;
    typedef LPWSTR LPTSTR;
    typedef LPWSTR PTSTR;
    typedef LPCWSTR LPCTSTR;
    typedef LPCWSTR PCTSTR;
#else
    typedef unsigned char TCHAR;
    typedef LPSTR LPTSTR;
    typedef LPSTR PTSTR;
    typedef LPCSTR LPCTSTR;
    typedef LPCSTR PCTSTR;
#endif

typedef TCHAR *PTCHAR;
```

Déclarations

Fonctions Windows

```
#ifndef UNICODE
    #define MessageBox  MessageBoxW
#else
    #define MessageBox  MessageBoxA
#endif
```

```
#ifndef UNICODE
    #define __TEXT(quote)  L##quote
#else
    #define __TEXT(quote)  quote
#endif
#define TEXT(quote)  __TEXT(quote)
```

API de conversion

- `MultiByteToWideChar` : `LPSTR -> LPWSTR`
- `WideCharToMultiByte` : `LPWSTR -> LPSTR`

- `printf("%s", "test")`
- `wprintf(L"%s", L"test")`

- `printf("%S", L"test")`
 - À éviter (échec si présence de caractères non ASCII)
- `wprintf(L"%S", "test")`

Fun facts

Gestion par le CRT

- Le CRT gère uniquement les caractères avec un encodage UCS-2, sans idée du contexte :
 - `wcslen(L"Effet")` = ?
 - Réponse : 4
 - U+FB00 : LATIN SMALL LIGATURE FF (ff ≠ ff)
 - `wcslen(L"😊")` = ?
 - Réponse : 2
 - 😊 : U+1F60A -> D8 3D DE 0A

Changement de contexte

- U+FE0E : VARIATION SELECTOR-15 -> Text style
- U+FE0F : VARIATION SELECTOR-16 -> Emoji style

- U+1F3FB : EMOJI MODIFIER FITZPATRICK TYPE-1-2
- U+1F3FC : EMOJI MODIFIER FITZPATRICK TYPE-3
- U+1F3FD : EMOJI MODIFIER FITZPATRICK TYPE-4
- U+1F3FE : EMOJI MODIFIER FITZPATRICK TYPE-5
- U+1F3FF : EMOJI MODIFIER FITZPATRICK TYPE-6



- U+200D : ZERO WIDTH JOINER (ZWJ)
- U+2640 : MALE SIGN (♂)
- U+2642 : FEMALE SIGN (♀)

- U+1F3CC(🏏) U+FE0F U+200D U+2642(♂) U+FE0F = 🏏♂
- U+1F3CC(🏏) U+1F3FE(🏏) U+200D U+2640 (♀) U+FE0F = 🏏♀

wcslen() ?