Tema 3: Representación de la información

Codificación de caracteres

Jose Luis Díaz

Curso 2011-2012

Contenidos

Esquema

Codificación de caracteres: problema

Un carácter es un elemento gráfico que se usa en el lenguaje escrito. La representación gráfica concreta del carácter, depende de la fuente tipográfica que se use. Esta "realización" se denomina *glifo*..

- En lugar de almacenar el dibujo (*glifo*) de cada carácter, se almacena simplemente un código numérico.
- Este código sirve como índice a una tabla que contiene el dibujo del carácter. .
- Inicialmente cada computador tenía su propia codificación.
- Para estandarizar el intercambio de información en Norteamérica se creó el ASCII

ASCII

b4 b3 b2 b1	b7 b6 b5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
		0	1	2	3	4	5	6	7
0 0 0 0	0			SP	0	@	P	`	p
0 0 0 1	1			!	1	A	Q	a	q
0 0 1 0	2			"	2	B	R	b	r
0 0 1 1	3			#	3	C	S	c	s
0 1 0 0	4			$	4	D	T	d	t
0 1 0 1	5			%	5	E	U	e	u
0 1 1 0	6			&	6	F	V	f	v
0 1 1 1	7			'	7	G	W	g	w
1 0 0 0	8			(	8	H	X	h	x
1 0 0 1	9			)	9	I	Y	i	y
1 0 1 0	10			*	:	J	Z	j	z
1 0 1 1	11			+	;	K	[	k	{
1 1 0 0	12			,	<	L	\	l	|
1 1 0 1	13			-	=	M	]	m	}
1 1 1 0	14			.	>	N	^	n	~
1 1 1 1	15			/	?	O	_	o	DEL

- Los primeros 32 códigos se reservan para caracteres de control.
- Sólo contiene caracteres del alfabeto inglés, dígitos y algunos símbolos.
- Le faltan caracteres de otros alfabetos, matemáticos, etc.
- No obstante tiene algunas propiedades interesantes.
  - Orden alfabético
  - 1 bit diferencia mayúsculas/minúsculas
  - Código de los dígitos

Esquema

Códigos de 8 bits

En un byte hay 8 bits, y por tanto permite almacenar 256 códigos.

Idea

El bit superior puede usarse de la siguiente forma:

- Si vale 0, los 7 bits inferiores usan la tabla ASCII
- Si vale 1, los 7 bits inferiores usan otra tabla

De este modo se pueden introducir 128 caracteres más.

Problemas

- Cada alfabeto requerirá su propia tabla de códigos.
- Las nuevas tablas deben ser estandarizadas.
- Incluso teniendo en cuenta lo anterior, hay alfabetos con más de 128 caracteres.

MS-DOS y las "*codepages*"

El sistema operativo MS-DOS usaba diferentes tablas , según el país en que se distribuía.

Codepage 437 *(países latinos)*

	·0	·1	·2	·3	·4	·5	·6	·7	·8	·9	·A	·B	·C	·D	·E	·F
8·	Ç U+00C7	ü U+00FC	é U+00E9	â U+00E2	ä U+00E4	à U+00E0	å U+00E5	ç U+00E7	ê U+00EA	ë U+00EB	è U+00E8	ï U+00EF	î U+00EE	ì U+00EC	Ä U+00C4	Å U+00C5
9·	É U+00C9	æ U+00E6	Æ U+00C6	ô U+00F4	ö U+00F6	ò U+00F2	û U+00FB	ù U+00F9	ÿ U+00FF	Ö U+00D6	Ü U+00DC	¢ U+00A2	£ U+00A3	¥ U+00A5	₧ U+20A7	ƒ U+0192
A·	á U+00E1	í U+00ED	ó U+00F3	ú U+00FA	ñ U+00F1	Ñ U+00D1	ª U+00AA	º U+00BA	¿ U+00BF	⌐ U+2310	¬ U+00AC	½ U+00BD	¼ U+00BC	¡ U+00A1	« U+00AB	» U+00BB
B·	░ U+2591	▒ U+2592	▓ U+2593	│ U+2502	┤ U+2524	╡ U+2561	╢ U+2562	╖ U+2556	╕ U+2555	╣ U+2563	║ U+2551	╗ U+2557	╝ U+255D	╜ U+255C	╛ U+255B	┐ U+2510
C·	└ U+2514	┴ U+2534	┬ U+252C	├ U+251C	─ U+2500	┼ U+253C	╞ U+255E	╟ U+255F	╚ U+255A	╔ U+2554	╩ U+2569	╦ U+2566	╠ U+2560	═ U+2550	╬ U+256C	╧ U+2567
D·	╨ U+2568	╤ U+2564	╥ U+2565	╙ U+2559	╘ U+2558	╒ U+2552	╓ U+2553	╫ U+256B	╪ U+256A	┘ U+2518	┌ U+250C	█ U+2588	▄ U+2584	▌ U+258C	▐ U+2590	▀ U+2580
E·	α U+03B1	ß U+00DF	Γ U+0393	π U+03C0	Σ U+03A3	σ U+03C3	µ U+00B5	τ U+03C4	Φ U+03A6	Θ U+0398	Ω U+03A9	δ U+03B4	∞ U+221E	φ U+03C6	ε U+03B5	∩ U+2229
F·	≡ U+2261	± U+00B1	≥ U+2265	≤ U+2264	⌠ U+2320	⌡ U+2321	÷ U+00F7	≈ U+2248	° U+00B0	∙ U+2219	· U+00B7	√ U+221A	ⁿ U+207F	² U+00B2	■ U+25A0	U+00A0

MS-DOS y las "*codepages*"

Codepage 850 *(países latinos, mejorado)*

	·0	·1	·2	·3	·4	·5	·6	·7	·8	·9	·A	·B	·C	·D	·E	·F
8·	Ç U+00C7	ü U+00FC	é U+00E9	â U+00E2	ä U+00E4	à U+00E0	å U+00E5	ç U+00E7	ê U+00EA	ë U+00EB	è U+00E8	ï U+00EF	î U+00EE	ì U+00EC	Ä U+00C4	Å U+00C5
9·	É U+00C9	æ U+00E6	Æ U+00C6	ô U+00F4	ö U+00F6	ò U+00F2	û U+00FB	ù U+00F9	ÿ U+00FF	Ö U+00D6	Ü U+00DC	ø U+00F8	£ U+00A3	Ø U+00D8	× U+00D7	ƒ U+0192
A·	á U+00E1	í U+00ED	ó U+00F3	ú U+00FA	ñ U+00F1	Ñ U+00D1	a U+00AA	o U+00BA	¿ U+00BF	® U+00AE	¬ U+00AC	½ U+00BD	¼ U+00BC	¡ U+00A1	« U+00AB	» U+00BB
B·	▒ U+2591	▒ U+2592	▓ U+2593	│ U+2502	┤ U+2524	Á U+00C1	Â U+00C2	À U+00C0	© U+00A9	╣ U+2563	║ U+2551	╗ U+2557	╝ U+255D	¢ U+00A2	¥ U+00A5	┐ U+2510
C·	└ U+2514	┴ U+2534	┬ U+252C	├ U+251C	─ U+2500	┼ U+253C	ã U+00E3	Ã U+00C3	╚ U+255A	╔ U+2554	╩ U+2569	╦ U+2566	╠ U+2560	═ U+2550	╬ U+256C	¤ U+00A4
D·	ð U+00F0	Ð U+00D0	Ê U+00CA	Ë U+00CB	È U+00C8	ı U+0131	Í U+00CD	Î U+00CE	Ï U+00CF	┘ U+2518	┌ U+250C	█ U+2588	▄ U+2584	¦ U+00A6	Ì U+00CC	■ U+2580
E·	Ó U+00D3	ß U+00DF	Ô U+00D4	Ò U+00D2	õ U+00F5	Õ U+00D5	µ U+00B5	þ U+00FE	Þ U+00DE	Ú U+00DA	Û U+00DB	Ù U+00D9	ý U+00FD	Ý U+00DD	¯ U+00AF	´ U+00B4
F·	U+00AD	± U+00B1	‗ U+2017	¾ U+00BE	¶ U+00B6	§ U+00A7	÷ U+00F7	¸ U+00B8	° U+00B0	¨ U+00A8	· U+00B7	¹ U+00B9	³ U+00B3	² U+00B2	■ U+25A0	U+00A0

Incluye mayúsculas acentuadas, símbolos como ©, etc.

Estándares ISO 8859

La *International Organization for Standarization* (ISO) creó una serie de 15 estándares para codificación de caracteres con 8 bits, llamados ISO-8859-1 a ISO-8859-16.
De ellos, se usan en España (y países latinos):

- ISO-8859-1 (también llamado *latin1*)
- ISO-8859-15 (también llamado *latin9*), que es una actualización del anterior

Windows

Windows usa una variante de ISO-8859-1, denominada **Windows-1252** que cambia la codificación del euro y otros símbolos, pero mantiene la de los caracteres acentuados y la eñe.

ISO-8859-1 “Latin 1”

ISO-8859-1

	·0	·1	·2	·3	·4	·5	·6	·7	·8	·9	·A	·B	·C	·D	·E	·F
8·	U+0080	U+0081	U+0082	U+0083	U+0084	U+0085	U+0086	U+0087	U+0088	U+0089	U+008A	U+008B	U+008C	U+008D	U+008E	U+008F
9·	U+0090	U+0091	U+0092	U+0093	U+0094	U+0095	U+0096	U+0097	U+0098	U+0099	U+009A	U+009B	U+009C	U+009D	U+009E	U+009F
A·	U+00A0	¡ U+00A1	¢ U+00A2	£ U+00A3	¤ U+00A4	¥ U+00A5	¦ U+00A6	§ U+00A7	¨ U+00A8	© U+00A9	ª U+00AA	« U+00AB	¬ U+00AC	U+00AD	® U+00AE	¯ U+00AF
B·	° U+00B0	± U+00B1	² U+00B2	³ U+00B3	´ U+00B4	µ U+00B5	¶ U+00B6	· U+00B7	¸ U+00B8	¹ U+00B9	º U+00BA	» U+00BB	¼ U+00BC	½ U+00BD	¾ U+00BE	¿ U+00BF
C·	À U+00C0	Á U+00C1	Â U+00C2	Ã U+00C3	Ä U+00C4	Å U+00C5	Æ U+00C6	Ç U+00C7	È U+00C8	É U+00C9	Ê U+00CA	Ë U+00CB	Ì U+00CC	Í U+00CD	Î U+00CE	Ï U+00CF
D·	Ð U+00D0	Ñ U+00D1	Ò U+00D2	Ó U+00D3	Ô U+00D4	Õ U+00D5	Ö U+00D6	× U+00D7	Ø U+00D8	Ù U+00D9	Ú U+00DA	Û U+00DB	Ü U+00DC	Ý U+00DD	Þ U+00DE	ß U+00DF
E·	à U+00E0	á U+00E1	â U+00E2	ã U+00E3	ä U+00E4	å U+00E5	æ U+00E6	ç U+00E7	è U+00E8	é U+00E9	ê U+00EA	ë U+00EB	ì U+00EC	í U+00ED	î U+00EE	ï U+00EF
F·	ð U+00F0	ñ U+00F1	ò U+00F2	ó U+00F3	ô U+00F4	õ U+00F5	ö U+00F6	÷ U+00F7	ø U+00F8	ù U+00F9	ú U+00FA	û U+00FB	ü U+00FC	ý U+00FD	þ U+00FE	ÿ U+00FF

Ya obsoleto, se usa el ISO-8859-15 en su lugar

ISO-8859-15 “Latin 9”

ISO-8859-15

	·0	·1	·2	·3	·4	·5	·6	·7	·8	·9	·A	·B	·C	·D	·E	·F
8·	U+0080	U+0081	U+0082	U+0083	U+0084	U+0085	U+0086	U+0087	U+0088	U+0089	U+008A	U+008B	U+008C	U+008D	U+008E	U+008F
9·	U+0090	U+0091	U+0092	U+0093	U+0094	U+0095	U+0096	U+0097	U+0098	U+0099	U+009A	U+009B	U+009C	U+009D	U+009E	U+009F
A·	U+00A0	¡ U+00A1	¢ U+00A2	£ U+00A3	€ U+20AC	¥ U+00A5	Š U+0160	§ U+00A7	š U+0161	© U+00A9	ª U+00AA	« U+00AB	¬ U+00AC	U+00AD	® U+00AE	¯ U+00AF
B·	° U+00B0	± U+00B1	² U+00B2	³ U+00B3	Ž U+017D	µ U+00B5	¶ U+00B6	· U+00B7	ž U+017E	¹ U+00B9	º U+00BA	» U+00BB	Œ U+0152	œ U+0153	Ÿ U+0178	¿ U+00BF
C·	À U+00C0	Á U+00C1	Â U+00C2	Ã U+00C3	Ä U+00C4	Å U+00C5	Æ U+00C6	Ç U+00C7	È U+00C8	É U+00C9	Ê U+00CA	Ë U+00CB	Ì U+00CC	Í U+00CD	Î U+00CE	Ï U+00CF
D·	Ð U+00D0	Ñ U+00D1	Ò U+00D2	Ó U+00D3	Ô U+00D4	Õ U+00D5	Ö U+00D6	× U+00D7	Ø U+00D8	Ù U+00D9	Ú U+00DA	Û U+00DB	Ü U+00DC	Ý U+00DD	Þ U+00DE	ß U+00DF
E·	à U+00E0	á U+00E1	â U+00E2	ã U+00E3	ä U+00E4	å U+00E5	æ U+00E6	ç U+00E7	è U+00E8	é U+00E9	ê U+00EA	ë U+00EB	ì U+00EC	í U+00ED	î U+00EE	ï U+00EF
F·	ð U+00F0	ñ U+00F1	ò U+00F2	ó U+00F3	ô U+00F4	õ U+00F5	ö U+00F6	÷ U+00F7	ø U+00F8	ù U+00F9	ú U+00FA	û U+00FB	ü U+00FC	ý U+00FD	þ U+00FE	ÿ U+00FF

Incluye el símbolo del Euro y algún otro.

Windows-1252

Windows-1252

	·0	·1	·2	·3	·4	·5	·6	·7	·8	·9	·A	·B	·C	·D	·E	·F
8·	€ U+20AC	…	‚ U+201A	ƒ U+0192	„ U+201E	… U+2026	† U+2020	‡ U+2021	ˆ U+02C6	‰ U+2030	Š U+0160	‹ U+2039	Œ U+0152	…	Ž U+017D	…
9·	…	‘ U+2018	’ U+2019	“ U+201C	” U+201D	• U+2022	– U+2013	— U+2014	˜ U+02DC	™ U+2122	š U+0161	› U+203A	œ U+0153	…	ž U+017E	Ÿ U+0178
A·	U+00A0	¡ U+00A1	¢ U+00A2	£ U+00A3	¤ U+00A4	¥ U+00A5	¦ U+00A6	§ U+00A7	¨ U+00A8	© U+00A9	ª U+00AA	« U+00AB	¬ U+00AC	U+00AD	® U+00AE	¯ U+00AF
B·	° U+00B0	± U+00B1	2 U+00B2	3 U+00B3	´ U+00B4	µ U+00B5	¶ U+00B6	· U+00B7	¸ U+00B8	1 U+00B9	º U+00BA	» U+00BB	¼ U+00BC	½ U+00BD	¾ U+00BE	¿ U+00BF
C·	À U+00C0	Á U+00C1	Â U+00C2	Ã U+00C3	Ä U+00C4	Å U+00C5	Æ U+00C6	Ç U+00C7	È U+00C8	É U+00C9	Ê U+00CA	Ë U+00CB	Ì U+00CC	Í U+00CD	Î U+00CE	Ï U+00CF
D·	Ð U+00D0	Ñ U+00D1	Ò U+00D2	Ó U+00D3	Ô U+00D4	Õ U+00D5	Ö U+00D6	× U+00D7	Ø U+00D8	Ù U+00D9	Ú U+00DA	Û U+00DB	Ü U+00DC	Ý U+00DD	Þ U+00DE	ß U+00DF
E·	à U+00E0	á U+00E1	â U+00E2	ã U+00E3	ä U+00E4	å U+00E5	æ U+00E6	ç U+00E7	è U+00E8	é U+00E9	ê U+00EA	ë U+00EB	ì U+00EC	í U+00ED	î U+00EE	ï U+00EF
F·	ð U+00F0	ñ U+00F1	ò U+00F2	ó U+00F3	ô U+00F4	õ U+00F5	ö U+00F6	÷ U+00F7	ø U+00F8	ù U+00F9	ú U+00FA	û U+00FB	ü U+00FC	ý U+00FD	þ U+00FE	ÿ U+00FF

Es la codificación usada por los editores de Windows en los países de habla hispana, y en Europa Occidental. Se parece a ISO-8852-1, pero observar las dos primeras filas.

Esquema

Unicode

- Unicode es un estándar que pretende unificar todas las codificaciones de caracteres en una sola.
- Unicode reserva espacio para $2^{20} + 2^{16}$ códigos diferentes.
- Cada código es un número denominado *codepoint* y representa casi siempre un carácter.
- Se ha diseñado para que sea compatible con ASCII y con ISO-8859-1

Organización de Unicode

- Los códigos se organizan por "planos". Cada plano contiene 2^{16} códigos, y hay 17 planos.
  - El plano 0 se denomina "*Basic Multilingual Plane*" (BMP) y contiene todos los alfabetos en uso en la actualidad.
  - El plano 1 contiene otros alfabetos antiguos, símbolos matemáticos y símbolos musicales.
  - Los restantes planos aún están siendo definidos.
- Un código Unicode suele escribirse con el prefijo `U-` seguido del valor numérico en hexadecimal
  - Las cuatro últimas cifras (hex) del código nos dan la localización del carácter dentro del plano, y las cifras superiores el número de plano.
  - Si el número de plano es `0000`, suele omitirse, y escribirse el prefijo `U+` en lugar de `U-`

Unicode, ASCII y latin1

- Los 128 primeros códigos de Unicode coinciden con ASCII.
- Los 128 códigos siguientes coinciden con ISO-8859-1.
- Por tanto los códigos `U+0000` hasta `U+00FF` coinciden con el estándar ISO-8859-1.
- La conversión entre Unicode e ISO-8859-1 es trivial.

Ejemplos de códigos Unicode

Código Unicode	Carácter	Nombre
U+0041	A	Letra A Latina Mayúscula
U+00D1	Ñ	Letra Ñ Latina Mayúscula
U+03B4	δ	Letra Delta Griega Minúscula
U+20AC	€	Símbolo del Euro
U+0259	ə	Letra "Schwa" del IPA
U+13A3	Ꭳ	Letra O Cherokee
U+263A	☺	Smiley
U+6F22	漢	Carácter chino "Han"
U+3042	あ	Letra A del alfabeto hiragana (japonés)
U+FEB6	ش	Letra Sheen en forma final (árabe)
U+2815	⠕	Letra O en Código Braille
U-00010330	𐌰	Letra A en Gótico
U-0001D4DB	𝓛	Letra L caligráfica matemática (transformada de Laplace)
U-0001D571	𝕱	Letra F gótica matemática (transformada de Fourier)
U-0001D160	♪	Nota Musical Corchea

Codificaciones

El estándar no especifica cómo almacenar en un computador un código Unicode.
Posibilidades:

- ¿Usar 8 bits para cada código?
  - Sólo cabrían los primeros 256 códigos.
- ¿Usar 16 bits para cada código?
  - Sólo cabe el plano 0, por otro lado el más común.
- ¿Usar 32 bits para cada código?
  - Caben todos los planos existentes.
  - Pero el desperdicio de espacio es muy grande.
- ¿Otras ideas?

UCS-4 y UCS-2

- La codificación más obvia (y más derrochadora), consiste en usar 32 bits para cada código.
  - Esta forma de codificar Unicode se denomina UCS-4, o a veces UTF-32.
  - En la práctica no es usada.
- Otra codificación obvia es usar 16 bits para cada código
  - Esto sólo permite codificar el plano 0.
  - Los restantes planos no pueden ser codificados.
  - Esta forma de codificar parcialmente Unicode se denomina UCS-2 pues usa 2 bytes por carácter.
  - No se recomienda esta codificación, pues no da cabida al Unicode completo.
  - No obstante, es la que usa Java.
  - A menudo se confunde con UTF-16 (que veremos después), pero UTF-16 sí que permite codificar el Unicode completo.

UTF-16

- El *Unicode Transformation Format* de 16 bits (UTF-16) permite codificar todos los planos Unicode, usando secuencias de 16 bits.
- Para ello usa un truco, que le permite usar un solo dato de 16 bits para los casos más comunes, y 2 datos de 16 bits para los casos más raros.

Caso más común

Si el código a codificar está por debajo de `U+10000` (Plano 0) la codificación UTF-16 coincide con UCS-32.

Ejemplo

El código Unicode de la 'A' es `U+0041`, y se codificará con 16 bits como `0000 0000 0100 0001`
El código Unicode del '€' es `U+20AC`, y se codificará con 16 bits como `0010 0000 1010 1100`

UTF-16

Restantes planos

Si el código a codificar está por encima de `U+FFFF`, la codificación usará dos datos de 16 bits, que se obtienen como sigue:

- Restar `10000h` al código Unicode.
  El resultado siempre cabrá en 20 bits.
  Llamemos `yyyyyyyyyy xxxxxxxxxx` a estos 20 bits.
- Generar un dato $D1$ con los 16 bits: `110110yyyyyyyyyy`
- Generar un dato $D2$ con los 16 bits: `110111xxxxxxxxxx`
- La secuencia $D1, D2$ es la codificación UTF-16 del código Unicode original.

Ejemplo

El código Unicode de una nota corchea es `U-0001D160` ¿Cuál es su codificación UTF-16?.

UTF-16: Observaciones

- Todas las codificaciones UTF-16 que requieren dos datos son fácilmente identificables:
  - El primer dato siempre está en el rango `D800`–`DBFF`
  - El segundo dato siempre está en el rango `DC00`–`DFFF`
- El estándar Unicode no define ningún carácter en estos rangos (denominados “rangos surrogados”) para posibilitar esta forma de codificación.
- Lo anterior permite detectar fácilmente errores en la codificación.

UTF-16: Almacenamiento y transmisión

La codificación UTF-16 usa datos de 16 bits. Sin embargo el almacenamiento en ficheros y la transmisión por sockets ocurre a nivel de byte.

Problema

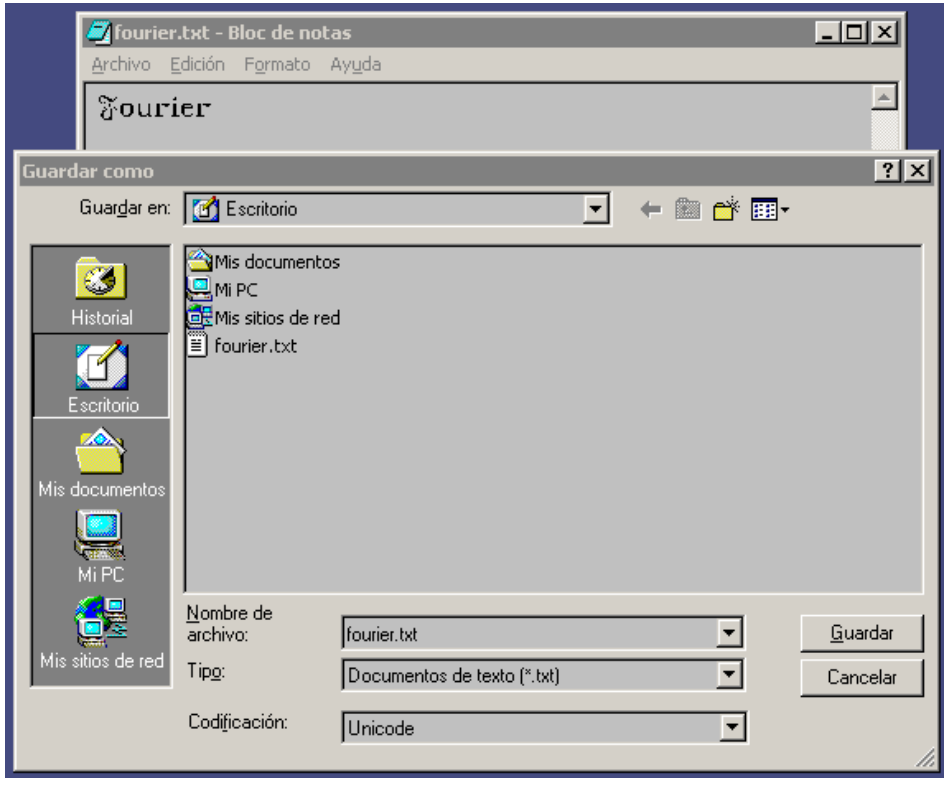
El dato de 16 bits debe partirse en dos bytes para poder ser almacenado o enviado. ¿Qué *endianity* usar?

La solución propuesta es definir tres variantes de la codificación:

- UTF-16BE (*Big Endian*).
- UTF-16LE (*Little Endian*)
- UTF-16 (sin especificar). Al principio del fichero o del flujo de datos se envía el código `U+FEFF`. Según se reciba como `FE`, `FF` o como `FF`, `FE`, se sabe la *endianity* de lo que sigue.

Ejemplo

En una máquina con arquitectura *little-endian*, y usando un editor de texto que guarda los documentos en UTF-16 escribimos la cadena "𝔉ourier" y guardamos.



¿Cuál es la secuencia de bytes almacenada en el fichero?

UTF-8

- UTF-8 es otra forma de codificar *cualquier carácter Unicode* como una secuencia de datos de 8 bits.
  - Si el carácter Unicode es menor de `U+0080`, se codificará con un solo byte (8 bits).
  - Los restantes casos requieren 2 ó más bytes, según el carácter en cuestión.
  - El caso peor requiere 4 bytes.

- Si el texto se compone sólo de ASCII puro, cada carácter se codificará en 1 byte (máximo ahorro de espacio).
- Los caracteres ISO-8859 requerirán 2 bytes.
- Es la codificación más habitual para las páginas Web.

UTF-8: Codificación

La codificación se guía por la siguiente tabla:

Códigos Unicode	Bits	UTF-8	Bits/byte
U+0000 a U+007F	7	0xxxxxxx	7
U+0080 a U+07FF	11	110xxxxx	5
		10xxxxxx	6
U+0800 a U+FFFF	16	1110xxxx	4
		10xxxxxx	6
		10xxxxxx	6
mayor de U-00010000	21	11110xxx	3
		10xxxxxx	6
		10xxxxxx	6
		10xxxxxx	6

UTF-8: Propiedades

- Si el primer bit del byte es 0, el byte directamente codifica un carácter con 7 bits.
- Si es distinto de cero:
  - El número de "1" seguidos indica cuántos bytes ocupa la codificación del carácter. El caso `10...` seria un error.
  - Tras este primer byte, deben venir otros que obligatoriamente empiezan por `10`. Si no, es un error.

Ejemplo

Si se lee la siguiente secuencia de bytes de un fichero supuestamente codificado con UTF-8:

```
41 6F E7 84 A6 70 C6 71 20 84 61
```

¿Cuántos caracteres se pueden decodificar correctamente antes de detectar un error? ¿En qué byte se detecta el error?

UTF-8: Ejemplos

¿Cuál es la codificación UTF-8 del carácter hebreo Aleph (א), si su código Unicode es el U+FB21?

- Caso entre `U+0800` y `U+FFFF`, requiere 16 bits: `1111101100100001`
- Los 16 bits se han de dividir en grupos de 4, 6 y 6, y resultan: `1111 101100 100001`
- Al primer grupo se le pone delante `1110`, y a los restantes se les pone `10`, lo que resulta en los tres bytes: `11101111`, `10101100`, `10100001`
- Resultado: `EF AC A1`

UTF-8: Ejemplos

- Codificar la cadena “Eñe” en UTF-8.
- Si esa cadena se interpreta erróneamente como ISO-8859-1 ¿Qué texto se mostrará en pantalla?
- Más ejemplos:

Código Unicode	Carácter	UTF-16BE	UTF-16LE	UTF-8
U+0041	A	00 41	41 00	41
U+00D1	Ñ	00 D1	D1 00	C3 91
U+20AC	€	20 AC	AC 20	E2 82 AC
U+263A	☺	26 3A	3A 26	E2 98 BA
U+3042	あ	30 42	42 30	E3 81 82
U-0001D4DB	𝓛	D8 35 DC DB	35 D8 DB DC	F0 9D 93 9B
U-0001D571	𝕱	D8 35 DD 71	35 D8 71 DD	F0 9D 95 B1