Representación de los caracteres

José Ignacio Estévez Damas
10 de marzo de 2021

ÍNDICE

1. INTRODUCCIÓN

Las humanos nos comunicamos mediante el lenguaje, por lo que no es de extrañar que después de los datos numéricos, la información en forma de texto sea la siguiente en importancia. En estos apuntes vamos a realizar una introducción a la representación en los computadores, de información textual mediante caracteres.

Solemos entender por *carácter* un elemento básico de la escritura, que es la expresión gráfica del lenguaje. Antes de proseguir tenemos que introducir algunas definiciones para poder determinar el verdadero concepto de carácter en el ámbito de la informática. Cuando en el campo de los computadores se habla de carácter, en realidad suele tratarse, aunque no siempre, de un "glifo". Un glifo puede representarse mediante un símbolo gráfico que puede utilizarse para simbolizar algo en la escritura.

Por otra parte, los grafemas son el bloque mínimo en la representación gráfica del lenguaje, la escritura. Un glifo como "a" se asocia en castellano al sonido de la primera vocal, una letra que también es un grafema. Sin embargo no es lo mismo un glifo que un grafema. De hecho, no hay una relación uno a uno: por ejemplo, dos glifos juntos pueden dar lugar a un grafema,

es el caso en castellano de la “ch”. Y por otra parte, existen muchos glifos muy usados que no están asociados a grafemas, como por ejemplo el símbolo “$”. Los números también usan glifos para su representación textual.

Así, pues nos ocuparemos de caracteres que son glifos, denominados también **caracteres imprimibles**, entendiendo que están relacionados con los grafemas del lenguaje, pero no son lo mismo. Pero no todos los caracteres que se usan en informática son glifos. En el ámbito de los computadores hay que entender que el texto puede contener pequeñas órdenes o indicaciones a periféricos capaces de mostrar o imprimir el texto: pueden ir desde la inserción de un elemento de formato como un salto de línea o un tabulador hasta activar un sonido. Son los denominados **caracteres no imprimibles o caracteres de control**.

Aunque sabemos que es fundamental para el glifo tener una representación gráfica concreta de cara al usuario humano final (así lo puede identificar), no es nuestro objetivo aquí tratar este aspecto gráfico. Un glifo como la letra “a” puede tener muchas representaciones gráficas comprensibles por un humano, y un programa que tenga almacenado un código binario que entienda que este código representa el carácter “a”, puede optar por cualquiera de ellas. El principal problema no es la representación gráfica de los caracteres, la tipografía, sino la estandarización del significado asociado a los códigos binarios.

El problema que nos ocupa por lo tanto es la estandarización de la codificación binaria de un juego de caracteres, entendiendo como tal a un conjunto formado por glifos y órdenes de control. Se trata de un problema crítico en computación por dos razones:

- La representación interna en un computador es binaria. Necesariamente todo debe ser representado mediante cadenas de bits.
- Puesto que los humanos nos comunicamos y representamos nuestros conceptos mediante información textual es necesario que nos pongamos de acuerdo en un estándar de codificación del texto. En caso contrario, en la inmensa mayoría de aplicaciones, los programas no podrían procesar adecuadamente la información, ya que no sabrían dotar del significado correcto a los códigos binarios.

Comenzaremos estos apuntes con el conocido estándar ASCII, precursor de otros muchos y terminaremos con la estandarización más exitosa hasta la fecha: el sistema UNICODE y sus esquemas de serialización.

2. El estándar US-ASCII

Las siglas ASCII se corresponden con “American Standard Code for Information Exchange”. Esta codificación fue creada en 1963, y se fue completando hasta llegar a la versión denominada US-ASCII de 1967. El estándar US-ASCII usa 7 bits que como ya saben permiten obtener 128 códigos binarios diferentes. En [2] tenemos una descripción de un estándar equivalente a US-ASCII.

El sistema contiene glifos orientados a la representación textual de un único idioma, el inglés, además de símbolos para representar fórmulas matemáticas simples y otros como el

Cuadro 2.1: US-ASCII: caracteres no imprimibles

Cód. ASCII	F/C	Oct.	Hexa.	Nombre	Comb. teclas	Descripción
0	00/00	0	0	NUL	(Ctrl-@)	NULL
1	00/01	1	1	SOH	(Ctrl-A)	START OF HEADING
2	00/02	2	2	STX	(Ctrl-B)	START OF TEXT
3	00/03	3	3	ETX	(Ctrl-C)	END OF TEXT
4	00/04	4	4	EOT	(Ctrl-D)	END OF TRANSMISSION
5	00/05	5	5	ENQ	(Ctrl-E)	ENQUIRY
6	00/06	6	6	ACK	(Ctrl-F)	ACKNOWLEDGE
7	00/07	7	7	BEL	(Ctrl-G)	BELL (Beep)
8	00/08	10	8	BS	(Ctrl-H)	BACKSPACE
9	00/09	11	9	HT	(Ctrl-I)	HORIZONTAL TAB
10	00/10	12	0A	LF	(Ctrl-J)	LINE FEED
11	00/11	13	0B	VT	(Ctrl-K)	VERTICAL TAB
12	00/12	14	0C	FF	(Ctrl-L)	FORM FEED
13	00/13	15	0D	CR	(Ctrl-M)	CARRIAGE RETURN
14	00/14	16	0E	SO	(Ctrl-N)	SHIFT OUT
15	00/15	17	0F	SI	(Ctrl-O)	SHIFT IN
16	01/00	20	10	DLE	(Ctrl-P)	DATA LINK ESCAPE
17	01/01	21	11	DC1	(Ctrl-Q)	DEVICECONTROL 1 (XON)
18	01/02	22	12	DC2	(Ctrl-R)	DEVICECONTROL 2
19	01/03	23	13	DC3	(Ctrl-S)	DEVICECONTROL 3 (XOFF)
20	01/04	24	14	DC4	(Ctrl-T)	DEVICECONTROL 4
21	01/05	25	15	NAK	(Ctrl-U)	NEGATIVE ACKNOWLEDGE
22	01/06	26	16	SYN	(Ctrl-V)	SYNCHRONOUS IDLE
23	01/07	27	17	ETB	(Ctrl-W)	END OF TRANSMISSION BLOCK
24	01/08	30	18	CAN	(Ctrl-X)	CANCEL
25	01/09	31	19	EM	(Ctrl-Y)	END OF MEDIUM
26	01/10	32	1A	SUB	(Ctrl-Z)	SUBSTITUTE
27	01/11	33	1B	ESC	(Ctrl-[)	ESCAPE
28	01/12	34	1C	FS	(Ctrl-\)	FILE SEPARATOR
29	01/13	35	1D	GS	(Ctrl-])	GROUP SEPARATOR
30	01/14	36	1E	RS		RECORD SEPARATOR
31	01/15	37	1F	US		UNIT SEPARATOR
127	07/15	177	7F	RUB	(Ctrl-?)	RUBOUT (DELETE)

dolar, la libra esterlina o el símbolo “”. En la tabla 2.1 pueden ver los códigos y descripción de los caracteres no imprimibles, mientras que en la tabla 2.2 tienen los restantes imprimibles.

El estándar US-ASCII sigue plenamente en uso. Por ejemplo, las denominadas URLs (direcciones web), deben codificarse en ASCII. Además, los juegos de caracteres que han ido sustituyendo a ASCII mantienen su compatibilidad, de manera que la mayor parte de los códigos ASCII siguen manteniendo el mismo significado.

Dado que US-ASCII está orientado sólo a la lengua inglesa, surgió inmediatamente la cuestión de cómo debía adaptarse a otros alfabetos, por ejemplo los alfabetos latinos de europa occidental. El organismo conocido como Intenational Organization of Standarization (ISO), diseñó una codificación basada en US-ASCII conocida como norma **ISO/IEC 646**, donde se establecen un conjunto de caracteres que pueden variar según la región. Estos caracteres configurables, proporcionan la posibilidad de definir juegos de caracteres basados en ASCII de 7 bits, adaptados a alfabetos particulares. Estos caracteres son: del 34 al 36, el 39, 44, 45, 47, 64, del 91 al 96 y del 123 al 126. En el caso del alfabeto español, existen dos variantes de ISO 646, conocidas ES ISO-IR 85 diseñada por IBM y esp ISO-IR diseñada por Olivetti, ambos estándares aprobados por la organización de estándares ECMA. En estas variantes de ASCII se incluyen caracteres como “ñ”, “¿”, la tilde o la diéresis.

Sin embargo, la restricción de 7 bits no era natural en computadores donde la información se almacenaba en bytes de 8 bits. Surge entonces el ISO 2022, que estableció como construir juegos de caracteres sobre la base de 8 bits a partir de otros de 7 bits. Esto fue necesario porque uno de los objetivos fundamentales al introducir este tipo de cambios es minimizar el impacto sobre los sistemas y software existente. A partir de esta fórmula de extensión a 8 bits, se crearon

Cuadro 2.2: Caracteres imprimibles US-ASCII

ASCII	F/C	Octal	Hexa.	Nombre	Descripción	ASCII	F/C	Octal	Hexa.	Nombre	Descripción
32	02/00	40	20	( )	SPACE	80	05/00	120	50	(P)	CAPITAL LETTER P
33	02/01	41	21	(!)	EXCLAMATIONMARK	81	05/01	121	51	(Q)	CAPITAL LETTER Q
34	02/02	42	22	(")	QUOTATION MARK	82	05/02	122	52	(R)	CAPITAL LETTER R
35	02/03	43	23	(#)	NUMBER SIGN	83	05/03	123	53	(S)	CAPITAL LETTER S
36	02/04	44	24	($)	DOLLAR SIGN	84	05/04	124	54	(T)	CAPITAL LETTER T
37	02/05	45	25	(%)	PERCENT SIGN	85	05/05	125	55	(U)	CAPITAL LETTER U
38	02/06	46	26	(&)	AMPERSAND	86	05/06	126	56	(V)	CAPITAL LETTER V
39	02/07	47	27	(')	APOSTROPHE	87	05/07	127	57	(W)	CAPITAL LETTER W
40	02/08	50	28	(()	LEFT PARENTHESIS	88	05/08	130	58	(X)	CAPITAL LETTER X
41	02/09	51	29	())	RIGHT PARENTHESIS	89	05/09	131	59	(Y)	CAPITAL LETTER Y
42	02/10	52	2A	(*)	ASTERISK	90	05/10	132	5A	(Z)	CAPITAL LETTER Z
43	02/11	53	2B	(+)	PLUS SIGN	91	05/11	133	5B	([)	LEFT SQUARE BRACKET
44	02/12	54	2C	(,)	COMMA	92	05/12	134	5C	(\)	REVERSE SOLIDUS (BACKSLASH)
45	02/13	55	2D	(-)	HYPHEN, MINUSSIGN	93	05/13	135	5D	(])	RIGHT SQUARE BRACKET
46	02/14	56	2E	(.)	PERIOD, FULL STOP	94	05/14	136	5E	(^)	CIRCUMFLEX ACCENT
47	02/15	57	2F	(/)	SOLIDUS, SLASH	95	05/15	137	5F	(_)	LOW LINE, UNDERLINE
48	03/00	60	30	(0)	DIGIT ZERO	96	06/00	140	60	(')	GRAVEACCENT
49	03/01	61	31	(1)	DIGIT ONE	97	06/01	141	61	(a)	SMALL LETTER a
50	03/02	62	32	(2)	DIGIT TWO	98	06/02	142	62	(b)	SMALL LETTER b
51	03/03	63	33	(3)	DIGIT THREE	99	06/03	143	63	(c)	SMALL LETTER c
52	03/04	64	34	(4)	DIGIT FOUR	100	06/04	144	64	(d)	SMALL LETTER d
53	03/05	65	35	(5)	DIGIT FIVE	101	06/05	145	65	(e)	SMALL LETTER e
54	03/06	66	36	(6)	DIGIT SIX	102	06/06	146	66	(f)	SMALL LETTER f
55	03/07	67	37	(7)	DIGIT SEVEN	103	06/07	147	67	(g)	SMALL LETTER g
56	03/08	70	38	(8)	DIGIT EIGHT	104	06/08	150	68	(h)	SMALL LETTER h
57	03/09	71	39	(9)	DIGIT NINE	105	06/09	151	69	(i)	SMALL LETTER i
58	03/10	72	3A	(:)	COLON	106	06/10	152	6A	(j)	SMALL LETTER j
59	03/11	73	3B	(;)	SEMICOLON	107	06/11	153	6B	(k)	SMALL LETTER k
60	03/12	74	3C	(<)	<SIGN, LEFT ANGLE BRACKET	108	06/12	154	6C	(l)	SMALL LETTER l
61	03/13	75	3D	(=)	EQUALS SIGN	109	06/13	155	6D	(m)	SMALL LETTER m
62	03/14	76	3E	(>)	>SIGN, RIGHT ANGLE BRACKET	110	06/14	156	6E	(n)	SMALL LETTER n
63	03/15	77	3F	(?)	QUESTION MARK	111	06/15	157	6F	(o)	SMALL LETTER o
64	04/00	100	40	(@)	COMMERCIAL AT SIGN	112	07/00	160	70	(p)	SMALL LETTER p
65	04/01	101	41	(A)	CAPITAL LETTER A	113	07/01	161	71	(q)	SMALL LETTER q
66	04/02	102	42	(B)	CAPITAL LETTER B	114	07/02	162	72	(r)	SMALL LETTER r
67	04/03	103	43	(C)	CAPITAL LETTER C	115	07/03	163	73	(s)	SMALL LETTER s
68	04/04	104	44	(D)	CAPITAL LETTER D	116	07/04	164	74	(t)	SMALL LETTER t
69	04/05	105	45	(E)	CAPITAL LETTER E	117	07/05	165	75	(u)	SMALL LETTER u
70	04/06	106	46	(F)	CAPITAL LETTER F	118	07/06	166	76	(v)	SMALL LETTER v
71	04/07	107	47	(G)	CAPITAL LETTER G	119	07/07	167	77	(w)	SMALL LETTER w
72	04/08	110	48	(H)	CAPITAL LETTER H	120	07/08	170	78	(x)	SMALL LETTER x
73	04/09	111	49	(I)	CAPITAL LETTER I	121	07/09	171	79	(y)	SMALL LETTER y
74	04/10	112	4A	(J)	CAPITAL LETTER J	122	07/10	172	7A	(z)	SMALL LETTER z
75	04/11	113	4B	(K)	CAPITAL LETTER K	123	07/11	173	7B	({)	LEFT CURLY BRACKET, LEFT BRACE
76	04/12	114	4C	(L)	CAPITAL LETTER L	124	07/12	174	7C	(|)	VERTICAL LINE, VERTICAL BAR
77	04/13	115	4D	(M)	CAPITAL LETTER M	125	07/13	175	7D	(})	RIGHT CURLY BRACKET, RIGHT BRACE
78	04/14	116	4E	(N)	CAPITAL LETTER N	126	07/14	176	7E	(~)	TILDE
79	04/15	117	4F	(O)	CAPITAL LETTER O						

juegos de caracteres para Japón, Corea y China, pero especialmente destaca el **ISO 8859**.

La internalización del juego de caracteres hizo que durante los años ochenta y buena parte de los noventa surgieran gran cantidad de tablas de código. Por ejemplo, las denominadas "code pages" de IBM/Microsoft que se usaban en sistemas bajo MS-DOS y Windows: IBM CP037, IBM CP437, y los Microsoft CP 1250 hasta Microsoft CP 1257. Todos estas tablas de códigos fueron soportadas por los sistemas Windows de la familia NT. El sistema de "code pages" siempre fue muy problemático porque cada aplicación podía elegir una codificación en la que trabajar.

Por este motivo, desde Windows XP y Windows Server 2003 los sistemas Windows empezaron a soportar también codificaciones basadas en UNICODE (Ver más abajo). Actualmente Microsoft recomienda usar esquemas de codificación basados en UNICODE como UTF-8 o UTF-16 (Ver más abajo).

3. El estándar ISO 8859

Antes de la llegada de UNICODE se popularizó el estándar ISO 8859. Este estándar agrupa un conjunto de juegos de caracteres donde se utilizan códigos binarios de 8 bits y se mantiene la compatibilidad con US-ASCII. Esta es la lista de juegos de caracteres originados por ISO 8859:

- **ISO/IEC 8859-1**. Conocido también como **Latin-1**. Lenguas de Europa Occidental.Incluye la mayoría de lenguas de europa incluyendo el español. Su revisión es ISO/IEC 8859-9.
- **ISO/IEC 8859-2**. Conocido también como **Latin-2**. Lenguas de Europa Central.
- **ISO/IEC 8859-3**. Conocido también como **Latin-3**. Lenguas del sur de Europa como el Turco. En este caso se usa preferentemente ISO/IEC 8859-9.
- **ISO/IEC 8859-4**. Conocido también como **Latin-4**. Lenguas del norte de europa.
- **ISO/IEC 8859-5**. Conocido también como **Latin-Cyrillic**.
- **ISO/IEC 8859-6**. Conocido también como **Latin-Arabic**.
- **ISO/IEC 8859-7**. Conocido también como **Latin-Greek**.
- **ISO/IEC 8859-8**. Conocido también como **Latin-Hebrew**.
- **ISO/IEC 8859-9**. Conocido también como **Latin-5**. Turco.
- **ISO/IEC 8859-10**. Conocido también como **Latin-6**. Lenguas nórdicas.
- **ISO/IEC 8859-11**. Conocido también como **Latin-Thai**.
- **ISO/IEC 8859-13**. Conocido también como **Latin-7**. Lenguas bálticas.
- **ISO/IEC 8859-14**. Conocido también como **Latin-8**. Incluye lenguas celtas.

- **ISO/IEC 8859-15**. Conocido también como **Latin-9**. Es muy importante porque reemplaza a Latin-1, introduce el símbolo del euro y algunos otros caracteres.
- **ISO/IEC 8859-16**. Conocido también como **Latin-10**.

En el mundo occidental se usa el ISO 8859-1 (Lain-1) [1] o su revisión ISO 8859-15 (Latin-9). Según el documento del estándar (en inglés) está diseñado para las siguientes lenguas: "Albanian, Basque, Breton, Catalan, Danish, Dutch, English, Faroese, Finnish, French (with restrictions, see Annex A.1, Notes), Frisian, Galician, German, Greenlandic, Icelandic, Irish Gaelic (new orthography), Italian, Latin, Luxemburgish, Norwegian, Portuguese, Rhaeto-Romanic, Scottish Gaelic, Spanish and Swedish".

Los responsables del estándar ISO/IEC8859 trabajan conjuntamente con Unicode para disminuir los problemas de aquellos dispositivos o software que usen ambos estándares. En este sentido, es importante saber que los primeros 256 caracteres de Unicode tienen la codificación ISO 8859 Latin-1.

La estrategia de base de las codificaciones ISO 8859, esto es, considerar diferentes tablas presenta hoy en día problemas importante:

- Pueden existir textos que requieran de caracteres de varias tablas.
- No es inmediato averiguar qué esquema de codificación se está usando en un documento.
- Ni siquiera con el gran número de tablas ISO 8859 se pueden representar todos los caracteres de todas las lenguas conocidas.

Para solucionar estos problemas, el consorcio UNICODE propuso una única tabla de codificación para todos los lenguajes existentes o que han existido.

4. El sistema UNICODE y la serialización

El sistema UNICODE [3] consiste un tabla "maestra" que asocia a cada carácter un "código " o identificador numérico multi-byte y una descripción. El sistema UNICODE tiene una contrapartida en el estándar ISO 10646-1, que define el denominado "Universal Character Set" conocido como UCS.

En UNICODE cada carácter tiene un identificador numérico asociado denomina "punto de código". Dicho identificador numérico se suele expresar en hexadecimal precedido por "U+". Por ejemplo, "U+0F1" es un punto de código.

Una característica importante de UNICODE es que un carácter puede construirse concatenando varios puntos de código. Pensemos por ejemplo en la letra "ñ" formada por "n" y por "~". Este glifo puede considerarse compuesto por otros dos, y UNICODE permite expresarlo así aunque exista un punto de código particular para la "ñ". Una consecuencia de esto es que un mismo carácter pueda codificarse de varios modos diferentes.

Una cadena UNICODE es una secuencia de puntos de código. Sin embargo, para representar la secuencia de puntos de código en un ordenador hay que escribirla en forma binaria. En los

Carácter	Código
H	01001000
o	01101111
l	01101100
a	01100001

Cuadro 4.1: Codificación ISO8859 de los caracteres de la palabra “Hola”.

estándares revisados en estos apuntes como US-ASCII o ISO 8859 esta traducción a binario es inmediata: simplemente se concatena el código binario de cada carácter. Por ejemplo, veamos en ISO8859 la cadena “Hola”. La tabla 4.1 muestra los códigos binarios de cada carácter.

Así, pues la cadena “Hola” se traduce por la secuencia de bits:

01001000011011110110110001100001

Esta secuencia se puede almacenar en la memoria de un computador con un byte por celda en grupos de 8 bits (1 byte por celda). Los lenguajes de programación suelen utilizar por defecto esta forma de representación para guardar cadenas de caracteres en memoria, aunque en sistemas como Windows se ofrece al programador una variedad de tipos de caracteres que permiten codificar la cadena en otras codificaciones como UTF-16 y que se describe más abajo.

La creación de una secuencia de bits a partir de unos datos se denomina serialización. Como hemos visto la serialización de US-ASCII o ISO 8859 es un proceso inmediato. Sin embargo, UNICODE admite diferentes esquemas de serialización que no son tan simples, siendo los más comunes los denominados UTF-16 y UTF-8 que describiremos después.

El sistema de puntos de código de UNICODE ya va por 101000 caracteres. La tabla UNICODE se divide en partes denominados planos. El denominado **plano básico multilingüe** va de los códigos U+0000 a U+FFFF, es decir 65536 caracteres que se corresponden con lo necesario para expresar todas las lenguas modernas (en realidad son menos de 65536 porque como veremos parte del rango no se puede usar). El resto de planos se utiliza para soportar alfabetos antiguos, caracteres raros, caracteres de control nuevos y otros caracteres especiales.

Como Unicode comenzó por el plano básico multilingües, se usaba inicialmente una traducción directa de cada punto de código a 16 bits bajo la representación de binario natural. Actualmente no son suficientes los 16 bits por lo que los esquemas de serialización son más complicados. Veamos por ejemplo, el esquema UTF-16:

- Los puntos de código entre U+0000 y U+FFFF (plano básico multilingüe) se traduce de forma directa a una palabra de 16 bits.
- Para el resto de planos, cada punto de código requiere 20 bits que se reparten entre dos palabras de 16 bits (pares subrogados). La primera palabra del par subrogado tiene como bits más significativos siempre 110110, mientras que la segunda tiene 110111,
- El primer paso para codificar en UTF-16 un punto de código no perteneciente al plano básico multilingüe es restar el valor 0×10000 y expresar el resultado en una celda de 20

Rango del punto de código	Formato de los bytes
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

Cuadro 4.2: Reglas de composición de los bytes UTF-8 para los puntos de código UNICODE.

bits.

- Los 10 bits más significativos del resultado son los bits menos significativos de la primera palabra del par subrogado, mientras que los 10 bits menos significativos del resultado de la resta son los bits menos significativos de la segunda palabra del par subrogado.
- De este modo, dada la parte fija establecida para cada palabra subrogada, el rango de valores de la primera palabra del par está entre $0 \times D800$ y $0 \times DBFF$, mientras que el rango de la segunda palabra es desde $0 \times DC00$ a $0 \times DFFF$.
- Los rangos anteriores no se utilizan para puntos de código en el plano básico multilingüe.
- De este modo, en UTF-16, cada palabra de 16 bits puede clasificarse en: perteneciente al plano básico multilingüe, primera palabra del par subrogado o segunda palabra del par subrogado. Sólo hay que revisar los 6 bits más significativos para determinar en qué categoría estamos.
- Si la palabra se identifica como perteneciente al plano básico multilingüe, se obtiene el punto de código directamente decodificando bajo la interpretación de binario natural.
- Si en cambio la palabra se identifica como primera palabra del par subrogado hay que tomar la siguiente palabra que deberá ser la segunda palabra del par subrogado y decodificar la secuencia de 20 bits extraída para identificar al punto de código.

Por contra, en el esquema de serialización UTF-8, la unidad de codificación es el byte. La codificación UTF-8 se basa en la tabla 4.2.

Por ejemplo, los primeros 128 puntos de código UNICODE, coincidentes con US-ASCII, se codifican en un byte cuyo bit más significativo es el 0. El resto de bits es la codificación directa del punto de código en binario natural con 7 bits. En general el proceso de codificación es sencillo:

- Determinar a partir del punto de código la fila de la tabla 4.2 en la que identificamos su rango. Observen que los rangos son mutuamente exclusivos, no se solapan.
- Una vez identificada la fila podemos escribir parte de los bytes necesarios, concretamente los bits más significativos de cada byte, según lo mostrado en la segunda columna.

- Entonces expresamos el punto de código en binario natural y vamos copiando los bits de menos significativo a más significativo en las “x” de los bytes codificados, empezando también de menos significativo a más significativo.

Se puede ver que la decodificación también es sencilla. Dado un byte que se supone es el más significativo de un conjunto que identifica un punto de código debemos primero determinar la fila de la tabla 4.2 a la que pertenece. Para ello basta mirar sus bits más significativos: si el más significativo es un 0 estamos en la primera fila, si los más significativos son 110 estamos en la segunda, 1110 es la tercera y 11110 es la segunda. A partir de ahí sabemos también cuantos bytes conforman el código: 1, 2, 3 o 4. Leemos los necesarios y simplemente extraemos en orden los bits que codifican el punto de código.

Puesto que los sistemas de codificación basados en UTF-8 se basan en el byte como elemento básico, y los primeros 128 puntos de código coinciden con la tabla US-ASCII, un documento de texto codificado con US-ASCII se interpretará correctamente por una aplicación que use UNICODE / UTF-8 como esquema de codificación serialización de caracteres.

Observen que para la parte de ISO 8859 que no coincide con ASCII se usan en UTF-8 dos bytes. Como estos caracteres no son muy frecuentes en lenguas occidentales este aumento de números de bytes, supone un porcentaje de aumento de los ficheros que puede rondar el2 %. Sin embargo en otras lenguas como el ruso el aumento puede ser del 100 %.

Los sistemas como Linux usan por defecto UTF-8. Las aplicaciones Windows han usado UTF-16 desde Windows 2000. Recientemente, en 2019, Windows 10 ha empezado a soportar UTF-8 y Microsoft recientemente ha recomendado el uso de UTF-8 para aplicaciones UWP de modo que sean compatibles con otras aplicaciones de entornos tipo Unix.

REFERENCIAS

[1] Final text of dis 8859-1, 8-bit single-byte coded graphic character sets – part 1: Latin alphabet no.1. `http://www.open-std.org/JTC1/SC2/WG3/docs/n411.pdf`. Accessed: 03-03-2016.

[2] Standard ecma 6. 7-bit coded character set. `http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-006.pdf`. Accessed: 03-03-2016.

[3] The Unicode Consortium. The unicode standard. version 8.0. core specification. `http://www.unicode.org/versions/Unicode8.0.0/`.