Codificación de caracteres

La codificación de caracteres es el método que permite convertir un carácter de un lenguaje natural en un símbolo de otro sistema de representación, como un número o una secuencia de pulsos eléctricos en un sistema electrónico, aplicando normas o reglas de codificación.

Por ejemplo: Código morse

0Ndt0TA19bQ

¿Por qué es necesario codificar caracteres?

No siempre el medio de almacenamiento o de transmisión permite que la información se envíe o guarde tal cual es, esto nos obliga a tener que adaptar la información al medio.

En el caso de las computadoras sólo entienden números por lo que buscamos representar los caracteres como números. Ejemplos: EBCDIC, ASCII, UTF-8, UTF-16, UTF-32, entre otros.

Tipos de codificación

Código ASCII

El código ASCII utiliza 7 bits para representar los caracteres, aunque inicialmente empleaba un bit adicional (bit de paridad) que se usaba para detectar errores en la transmisión.

Tipos de caracteres:

- Caracteres de control
- Caracteres imprimibles

ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

Código Extendido

A medida que la tecnología informática se difundió a lo largo del mundo, se desarrollaron diferentes estándares y las empresas desarrollaron variaciones del código ASCII para facilitar la escritura de lenguas diferentes al inglés.

Así surge el código ASCII extendido que utiliza 8 bits para poder representar más caracteres.

Extended ASCII characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ò	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

Unicode

Fue diseñado poder representar textos de numerosos idiomas y reemplazar los esquemas de codificación de caracteres ya existentes, por las limitaciones de tamaño que estos tenían.

Define tres formas de codificación bajo el nombre UTF (Unicode transformation format): UTF-8, UTF-16 y UTF-32

character	encoding	bits
A	UTF-8	01000001
A	UTF-16	00000000 01000001
A	UTF-32	00000000 00000000 00000000 01000001
あ	UTF-8	11100011 10000001 10000010
あ	UTF-16	00110000 01000010
あ	UTF-32	00000000 00000000 00110000 01000010

UTF-8

Número de bytes	Bits para el código de caracteres	Primer código	Último código	Byte 1	Byte 2	Byte 3	Byte 4
1	7 (ASCII)	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Ventajas:

- Permite codificar cualquier carácter Unicode.
- Es compatible con US-ASCII, la codificación del repertorio de 7 bits es directa.
- UTF-8 ahorrará espacio de almacenamiento para caracteres comunes

Desventajas:

- Utiliza símbolos de longitud variable; eso significa que diferentes caracteres pueden codificarse con distinto número de bytes.
- Los caracteres ideográficos usan 3 bytes en UTF-8, pero sólo 2 en UTF-16. Así, los textos chinos, japoneses o coreanos ocupan más espacio cuando se representan en UTF-8
- UTF-8 ofrece peor rendimiento que UTF-16 y UTF-32 en cuanto a coste de computación, por ejemplo en operaciones de ordenación.

Comparación

	ASCII	ASCII-E	UTF-8	UTF-16	UTF-32
Tamaño	7 bits	8 bits	8 a 32 bits	16 a 32 bits	32 bits
Caracteres	128	256	Todos	Todos	Todos
Endianness (Formato de almacenamiento de datos de más de un byte)	-	-	Orientado al byte, no tiene problemas.	Si	Si
Admite BOM (Byte Order Mask). Opcional, se suele colocar al inicio para dejar explícita la ordenación de bytes. Si no se aclara se asume big-endian	-	-	Si (no se recomienda)	Si. No se usa en UTF-16BE ni UTF-16LE	Si. No se usa en UTF-32BE ni UTF-32LE

UTF-8 optimiza el uso del espacio sacrificando el rendimiento del procesador ya que debe trabajar mas para determinar la longitud de cada símbolo. UTF-32 optimiza el rendimiento del procesador sacrificando el uso del espacio en memoria/disco.

Tablas comparativas con ejemplos concretos

	Texto	Bytes	Texto	Bytes	Texto	Bytes	Texto	Bytes
ISO_8859_1	Hola	4	áéíóú	5	??????	6	????	4
US_ASCII	Hola	4	?????	5	??????	6	????	4
UTF-8	Hola	4	áéíóú	10	Šťŕĭńġ	12	先秦兩漢	12
UTF-16	Hola	10	áéíóú	12	Šťŕĭńġ	14	先秦兩漢	10

Representación de una cadena en memoria

Existen dos formas de representar una cadena en memoría:

1. **Pascal (P-Strings o cadenas de pascal)**: Se destina 1 byte para almacenar la longitud. Desventaja: La longitud máxima de la cadena depende del tamaño del campo destinado a la longitud.

[Longitud]	'H'	'O'	'L'	'A'
04	48	4F	4C	41

2. **C**: La cadena termina cuando se aparece el carácter nulo. Ventaja: puedo tener cadenas de longitud infinita. Desventaja: Si quiero saber la longitud del texto tengo que contar hasta encontrar el caracter nulo.

'H'	'O'	'L'	'A'	NULL
48	4F	4C	41	00

Diferentes tipos de saltos de linea CRLF (\r\n) VS LF (\n)

4i1MqlplzIk

Volver

(158)

— *Fernando Velcic*