

Estándares Para Codificación de Caracteres

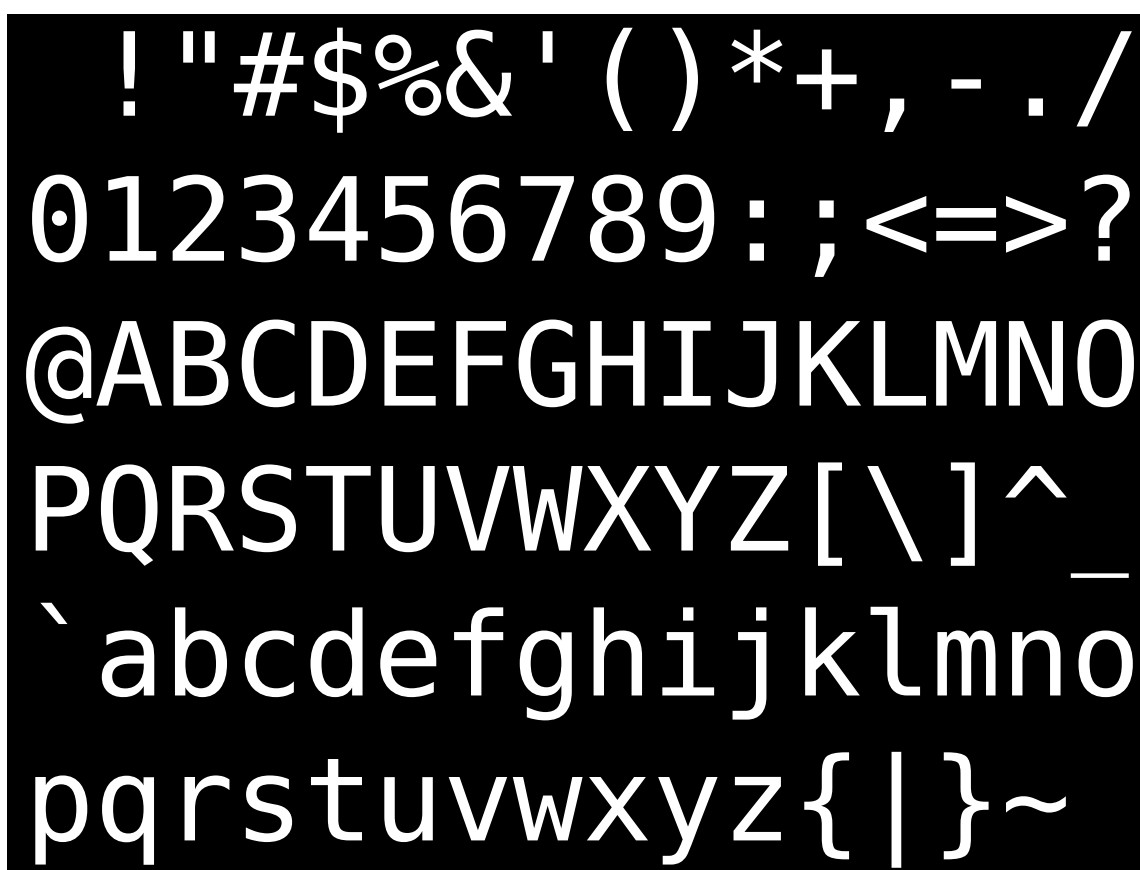
Índice general

1	ASCII	1
1.1	Vista general	2
1.2	Historia	2
1.3	Los caracteres de control ASCII	3
1.4	Caracteres imprimibles ASCII	4
1.5	Rasgos estructurales	4
1.6	Otros nombres para ASCII	4
1.7	Variantes de ASCII	5
1.8	Arte ASCII	5
1.9	Véase también	6
1.9.1	Variantes ASCII de ordenadores específicos	6
1.10	Referencias	6
1.10.1	Generales	6
1.10.2	Al pie	7
1.11	Enlaces externos	7
2	ASCII extendido	8
2.1	Conjunto de caracteres ASCII Extendido del IBM PC y sucesores	9
3	Unicode	10
3.1	Alcance del estándar	11
3.1.1	Relación con otros estándares	11
3.2	Repertorio de caracteres	11
3.2.1	Tipos de caracteres	12
3.2.2	Composición de caracteres y secuencias	12
3.2.3	Repertorio unificado chino, coreano y japonés	13
3.3	Elementos del estándar Unicode	13
3.3.1	Principios de diseño	13
3.3.2	Base de datos de caracteres	14
3.4	Tratamiento de la información	14
3.4.1	Formas de codificación	14
3.4.2	Esquemas de codificación	15
3.5	Historia	15

3.6	Véase también	15
3.7	Referencias	15
3.8	Enlaces externos	16
4	UTF-8	17
4.1	Historia	17
4.2	Descripción	17
4.2.1	Codificación de los caracteres	18
4.2.2	Errores de codificación	19
4.2.3	<i>Byte order mark</i> (BOM)	19
4.3	Derivaciones de UTF-8	19
4.3.1	CESU-8	19
4.3.2	UTF-8 modificado	20
4.4	Ventajas y desventajas	20
4.5	Referencias	20
4.6	Véase también	21
4.7	Enlaces externos	21
5	UTF-16	22
5.1	Historia	22
5.2	Descripción	22
5.2.1	Pares subrogados	23
5.2.2	Errores de codificación	23
5.2.3	Esquemas de codificación y BOM	24
5.3	Véase también	24
5.4	Referencias	24
6	ISO/IEC 8859	25
6.1	Tabla	25
6.2	Text and image sources, contributors, and licenses	26
6.2.1	Text	26
6.2.2	Images	26
6.2.3	Content license	27

Capítulo 1

ASCII



Hay 95 caracteres ASCII imprimibles, numerados del 32 al 126.

ASCII (acrónimo inglés de **American Standard Code for Information Interchange** — *Código Estándar Estadounidense para el Intercambio de Información*), pronunciado generalmente [áski] o [ásci] , es un código de caracteres basado en el alfabeto latino, tal como se usa en inglés moderno. Fue creado en 1963 por el Comité Estadounidense de Estándares (ASA, conocido desde 1969 como el Instituto Estadounidense de Estándares Nacionales, o ANSI) como una refundición o evolución de los conjuntos de códigos utilizados entonces en telegrafía. Más tarde, en 1967, se incluyeron las minúsculas, y se redefinieron algunos códigos de control para formar el código conocido como **US-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. A menudo se llama incorrectamente ASCII a otros códigos de caracteres de 8 bits, como el estándar ISO-8859-1, que es una extensión que utiliza 8 bits para proporcionar caracteres adicionales usados en idiomas distintos al inglés, como el español.

ASCII fue publicado como estándar por primera vez en 1967 y fue actualizado por última vez en 1986. En la actualidad define códigos para 32 caracteres no imprimibles, de los cuales la mayoría son **caracteres de control** que tienen efecto sobre cómo se procesa el texto, más otros 95 caracteres imprimibles que les siguen en la numeración (empezando por el carácter espacio).

Casi todos los sistemas informáticos actuales utilizan el código ASCII o una extensión compatible para representar textos y para el control de dispositivos que manejan texto como el teclado. No deben confundirse los códigos ALT+número de teclado con los códigos ASCII.

1.1 Vista general

Las computadoras solamente entienden números. El código ASCII es una representación numérica de un carácter como 'a' o '@'.^[1]

Como otros códigos de formato de representación de caracteres, el ASCII es un método para una correspondencia entre cadenas de bits y una serie de símbolos (alfanuméricos y otros), permitiendo de esta forma la comunicación entre dispositivos digitales así como su procesamiento y almacenamiento. El código de caracteres ASCII^[2] —o una extensión compatible (ver más abajo)— se usa casi en todos los ordenadores, especialmente con **ordenadores personales** y **estaciones de trabajo**. El nombre más apropiado para este código de caracteres es “US-ASCII”.^[3]

ASCII es, en sentido estricto, un código de siete **bits**, lo que significa que usa cadenas de bits representables con siete dígitos binarios (que van de 0 a 127 en base decimal) para representar información de caracteres. En el momento en el que se introdujo el código ASCII muchos ordenadores trabajaban con grupos de ocho bits (**bytes** u **octetos**), como la unidad mínima de información; donde el octavo bit se usaba habitualmente como **bit de paridad** con funciones de control de errores en líneas de comunicación u otras funciones específicas del dispositivo. Las máquinas que no usaban la comprobación de paridad asignaban al octavo bit el valor cero en la mayoría de los casos, aunque otros sistemas como las computadoras **Prime**, que ejecutaban **PRIMOS** ponían el octavo bit del código ASCII a uno. El código ASCII define una relación entre caracteres específicos y secuencias de bits; además de reservar unos cuantos códigos de control para el procesador de textos, y no define ningún mecanismo para describir la estructura o la apariencia del texto en un documento; estos asuntos están especificados por otros lenguajes como los **lenguajes de etiquetas**.

1.2 Historia

El código ASCII se desarrolló en el ámbito de la **telegrafía** y se usó por primera vez comercialmente como un código de teleimpresión impulsado por los servicios de datos de **Bell**. Bell había planeado usar un código de seis bits, derivado de **Fieldata**, que añadía puntuación y letras minúsculas al más antiguo código de teleimpresión **Baudot**, pero se le convenció para que se unieran al subcomité de la Agencia de Estándares Estadounidense (ASA), que había empezado a desarrollar el código ASCII. Baudot ayudó en la automatización del envío y recepción de mensajes telegráficos, y tomó muchas características del **código Morse**; sin embargo, a diferencia del código Morse, Baudot usó códigos de longitud constante. Comparado con los primeros códigos telegráficos, el código propuesto por Bell y ASA resultó en una reorganización más conveniente para ordenar listas (especialmente porque estaba ordenado alfabéticamente) y añadió características como la '**secuencia de escape**'. La Agencia de Estándares Estadounidense (ASA), que se convertiría más tarde en el Instituto Nacional Estadounidense de Estándares (**ANSI**), publicó por primera vez el código ASCII en 1963. El ASCII publicado en 1963 tenía una flecha apuntando hacia arriba (↑) en lugar del circunflejo (^) y una flecha apuntando hacia la izquierda en lugar del guión bajo (_). La versión de 1967 añadió las letras minúsculas, cambió los nombres de algunos códigos de control y cambió de lugar los dos códigos de control ACK y ESC de la zona de letras minúsculas a la zona de códigos de control. ASCII fue actualizado en consecuencia y publicado como ANSI X3.4-1968, ANSI X3.4-1977, y finalmente ANSI X3.4-1986. Otros órganos de estandarización han publicado códigos de caracteres que son idénticos a ASCII. Estos códigos de caracteres reciben a menudo el nombre de ASCII, a pesar de que ASCII se define estrictamente solamente por los estándares ASA/ANSI:

- La Asociación Europea de Fabricantes de Ordenadores (**ECMA**) publicó ediciones de su clon de ASCII, ECMA-6 en 1965, 1967, 1970, 1973, 1983, y 1991. La edición de 1991 es idéntica a ANSI X3.4-1986.^[4]
- La Organización Internacional de Estandarización (**ISO**) publicó su versión, ISO 646 (más tarde ISO/IEC 646) en 1967, 1972, 1983 y 1991. En particular, ISO 646:1972 estableció un conjunto de versiones específicas para cada país donde los caracteres de puntuación fueron reemplazados con caracteres no ingleses. ISO/IEC 646:1991 La International Reference Version es la misma que en el ANSI X3.4-1986.

USASCII code chart

<div><div><div>b7b6b5b4b3b2b1</div><div>Bits</div></div><div><div>→</div><div>→</div><div>→</div><div>→</div></div></div>					000	001	010	011	100	101	110	111
<div><div>Column</div><div>→</div></div>					0	1	2	3	4	5	6	7
<div><div>Row</div><div>↓</div></div>					0	1	2	3	4	5	6	7
0000	0	NUL	DLE	SP	0	@	P	\	p			
0001	1	SOH	DC1	!	1	A	Q	a	q			
0010	2	STX	DC2	"	2	B	R	b	r			
0011	3	ETX	DC3	#	3	C	S	c	s			
0100	4	EOT	DC4	\$	4	D	T	d	t			
0101	5	ENQ	NAK	%	5	E	U	e	u			
0110	6	ACK	SYN	&	6	F	V	f	v			
0111	7	BEL	ETB	'	7	G	W	g	w			
1000	8	BS	CAN	(8	H	X	h	x			
1001	9	HT	EM)	9	I	Y	i	y			
1010	10	LF	SUB	*	:	J	Z	j	z			
1011	11	VT	ESC	+	;	K	[k	{			
1100	12	FF	FS	,	<	L	\	l				
1101	13	CR	GS	-	=	M]	m	}			
1110	14	SO	RS	.	>	N	^	n	~			
1111	15	SI	US	/	?	O	_	o	DEL			

La carta de Código ASCII 1968 de los EE.UU. fue estructurada con dos columnas de caracteres de control, una columna con caracteres especiales, una columna con números, y cuatro columnas de letras

- La Unión Internacional de Telecomunicaciones (ITU) publicó su versión de ANSI X3.4-1986, Recomendación ITU T.50, en 1992. A principios de la **década de 1970** publicó una versión como Recomendación CCITT V.3.
- **DIN** publicó una versión de ASCII como el estándar DIN 66003 en 1974.
- El Grupo de Trabajo en Ingeniería de Internet (IETF) publicó una versión en 1969 como **RFC 20**, y estableció la versión estándar para Internet, basada en ANSI X3.4-1986, con la publicación de **RFC 1345** en 1992.
- La versión de **IBM** de ANSI X3.4-1986 se publicó en la literatura técnica de IBM como **página de códigos 367**.

El código ASCII también está incluido en su probable relevo, **Unicode**, constituyendo los primeros 128 caracteres (o los 'más bajos').

1.3 Los caracteres de control ASCII

El código ASCII reserva los primeros 32 códigos (numerados del 0 al 31 en decimal) para **caracteres de control**: códigos no pensados originalmente para representar información imprimible, sino para controlar dispositivos (como **impresoras**) que usaban ASCII. Por ejemplo, el carácter 10 representa la función “nueva línea” (line feed), que hace que una impresora avance el papel, y el carácter 27 representa la **tecla “escape”** que a menudo se encuentra en la esquina superior izquierda de los **teclados** comunes. El código 127 (los siete bits a uno), otro carácter especial, equivale a “suprimir” (“delete”). Aunque esta función se asemeja a otros caracteres de control, los diseñadores de ASCII idearon este código para poder “borrar” una sección de **papel perforado** (un medio de almacenamiento popular hasta la década de 1980) mediante la perforación de todos los agujeros posibles de una posición de carácter concreta, reemplazando cualquier información previa. Dado que el código 0 era ignorado, fue posible dejar huecos (regiones de agujeros) y más tarde hacer correcciones. Muchos de los caracteres de control ASCII servían para marcar paquetes de datos, o para controlar protocolos de transmisión de datos (por ejemplo ENquiry, con el significado: ¿hay alguna

estación por ahí?, ACKnowledge: recibido o ", Start Of Header: inicio de cabecera, Start of TeXt: inicio de texto, End of TeXt: final de texto, etc.). ESCape y SUBstitute permitían a un protocolo de comunicaciones, por ejemplo, marcar datos binarios para que contuviesen códigos con el mismo código que el carácter de protocolo, y que el receptor pudiese interpretarlos como datos en lugar de como caracteres propios del protocolo. Los diseñadores del código ASCII idearon los caracteres de separación para su uso en sistemas de cintas magnéticas. Dos de los caracteres de control de dispositivos, comúnmente llamados **XON** y **XOFF** generalmente ejercían funciones de caracteres de **control de flujo** para controlar el flujo hacia un dispositivo lento (como una impresora) desde un dispositivo rápido (como un ordenador), de forma que los datos no saturasen la capacidad de recepción del dispositivo lento y se perdiesen. Los primeros usuarios de ASCII adoptaron algunos de los códigos de control para representar “metainformación” como final-de-línea, principio/final de un elemento de datos, etc. Estas asignaciones a menudo entraban en conflicto, así que parte del esfuerzo de convertir datos de un formato a otro comporta hacer las conversiones correctas de metainformación. Por ejemplo, el carácter que representa el final-de-línea en ficheros de texto varía con el **sistema operativo**. Cuando se copian archivos de un sistema a otro, el sistema de conversión debe reconocer estos caracteres como marcas de final-de-línea y actuar en consecuencia. Actualmente los usuarios de ASCII usan menos los caracteres de control, (con algunas excepciones como “retorno de carro” o “nueva línea”). Los lenguajes modernos de etiquetas, los protocolos modernos de comunicación, el paso de dispositivos basados en texto a basados en gráficos, el declive de las teleimpresoras, las tarjetas perforadas y los papeles continuos han dejado obsoleta la mayoría de caracteres de control.

1.4 Caracteres imprimibles ASCII

El carácter 'espacio', designa al espacio entre palabras, y se produce normalmente por la barra espaciadora de un teclado. Los códigos del 33 al 126 se conocen como caracteres imprimibles, y representan letras, dígitos, signos de puntuación y varios símbolos. El ASCII de siete bits proporciona siete caracteres “nacionales” y, si la combinación concreta de hardware y software lo permite, puede utilizar combinaciones de teclas para simular otros caracteres internacionales: en estos casos un backspace puede preceder a un acento abierto o grave (en los estándares británico y estadounidense, pero sólo en estos estándares, se llama también “opening single quotation mark”), una **tilde** o una “marca de respiración”.

1.5 Rasgos estructurales

- Los dígitos del 0 al 9 se representan con sus valores prefijados con el valor 0011 en binario (esto significa que la conversión **BCD-ASCII** es una simple cuestión de tomar cada unidad bcd y prefijarla con 0011).
- Las cadenas de bits de las letras minúsculas y mayúsculas sólo difieren en un bit, simplificando de esta forma la conversión de uno a otro grupo.

1.6 Otros nombres para ASCII

La **RFC 1345** (publicada en junio de 1992) y el **registro IANA de códigos de caracteres**, reconocen los siguientes nombres alternativos para ASCII para su uso en Internet.

- ANSI_X3.4-1968 (nombre canónico)
- ANSI_X3.4-1986
- ASCII
- US-ASCII (nombre MIME recomendado)
- us
- ISO646-US
- ISO_646.irv:1991
- iso-ir-6

- IBM367
- cp367
- csASCII

De estos, sólo los nombres “US-ASCII” y “ASCII” se usan ampliamente. A menudo se encuentran en el parámetro de “código de caracteres” opcional en la cabecera Content-Type de algunos mensajes **MIME**, en el elemento equivalente “meta” de algunos documentos **HTML**, y en la parte de declaración de codificación de carácter de la cabecera de algunos documentos **XML**.

1.7 Variantes de ASCII

A medida que la tecnología informática se difundió a lo largo del mundo, se desarrollaron diferentes estándares y las empresas desarrollaron muchas variaciones del código ASCII para facilitar la escritura de lenguas diferentes al inglés que usaran alfabetos latinos. Se pueden encontrar algunas de esas variaciones clasificadas como “**ASCII Extendido**”, aunque en ocasiones el término se aplica erróneamente para cubrir todas las variantes, incluso las que no preservan el conjunto de códigos de caracteres original ASCII de siete bits. La **ISO 646** (1972), el primer intento de remediar el sesgo pro-inglés de la codificación de caracteres, creó problemas de compatibilidad, pues también era un código de caracteres de 7 bits. No especificó códigos adicionales, así que reasignó algunos específicamente para los nuevos lenguajes. De esta forma se volvió imposible saber en qué variante se encontraba codificado el texto, y, consecuentemente, los procesadores de texto podían tratar una sola variante. La tecnología mejoró y aportó medios para representar la información codificada en el octavo bit de cada byte, liberando este bit, lo que añadió otros 128 códigos de carácter adicionales que quedaron disponibles para nuevas asignaciones. Por ejemplo, **IBM** desarrolló páginas de código de 8 bits, como la **página de códigos 437**, que reemplazaba los caracteres de control con símbolos gráficos como sonrisas, y asignó otros caracteres gráficos adicionales a los 128 bytes superiores de la página de códigos. Algunos sistemas operativos como DOS, podían trabajar con esas páginas de código, y los fabricantes de **ordenadores personales** incluyeron soporte para dichas páginas en su hardware. Los estándares de ocho bits como **ISO 8859** y **Mac OS Roman** fueron desarrollados como verdaderas extensiones de ASCII, dejando los primeros 127 caracteres intactos y añadiendo únicamente valores adicionales por encima de los 7-bits. Esto permitió la representación de un abanico mayor de lenguajes, pero estos estándares continuaron sufriendo incompatibilidades y limitaciones. Todavía hoy, **ISO-8859-1** y su variante **Windows-1252** (a veces llamada erróneamente ISO-8859-1) y el código ASCII original de 7 bits son los códigos de carácter más comúnmente utilizados.

Unicode y Conjunto de Caracteres Universal (UCS) ISO/IEC 10646 definen un conjunto de caracteres mucho mayor, y sus diferentes formas de codificación han empezado a reemplazar ISO 8859 y ASCII rápidamente en muchos entornos. Mientras que ASCII básicamente usa códigos de 7-bits, Unicode y UCS usan “code points” o apuntadores relativamente abstractos: números positivos (incluyendo el cero) que asignan secuencias de 8 o más bits a caracteres. Para permitir la compatibilidad, Unicode y UCS asignan los primeros 128 apuntadores a los mismos caracteres que el código ASCII. De esta forma se puede pensar en ASCII como un subconjunto muy pequeño de Unicode y UCS. La popular codificación **UTF-8** recomienda el uso de uno a cuatro valores de 8 bits para cada apuntador, donde los primeros 128 valores apuntan a los mismos caracteres que ASCII. Otras codificaciones de caracteres como **UTF-16** se parece a ASCII en cómo representan los primeros 128 caracteres de Unicode, pero tienden a usar 16 a 32 bits por carácter, así que requieren de una conversión adecuada para que haya compatibilidad entre ambos códigos de carácter. La palabra *ASCIIbético* (o, más habitualmente, la palabra “inglesa” *ASCIIbetical*) describe la ordenación según el orden de los códigos ASCII en lugar del orden alfabético.^[5]

La abreviatura **ASCIIZ** o **ASCIZ** se refiere a una cadena de caracteres terminada en cero (del inglés “zero”). Es muy normal que el código ASCII sea embebido en otros sistemas de codificación más sofisticados y por esto debe tenerse claro cual es papel del código ASCII en la tabla o mapa de caracteres de un ordenador.

1.8 Arte ASCII

El código de caracteres ASCII es el soporte de una disciplina artística minoritaria, el **arte ASCII**, que consiste en la composición de imágenes mediante caracteres imprimibles ASCII. El efecto resultante ha sido comparado con el **puntillismo**, pues las imágenes producidas con esta técnica generalmente se aprecian con más detalle al ser vistas a

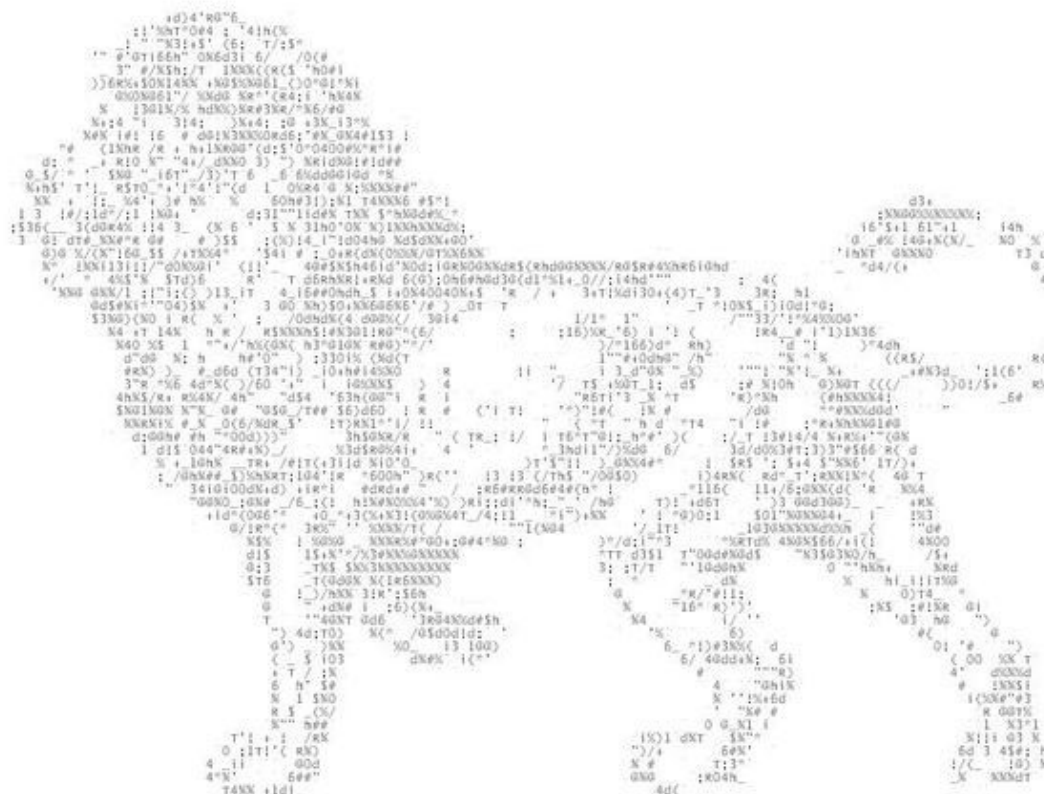


Imagen de león creada con arte ascii.

distancia. El arte ASCII empezó siendo un arte experimental, pero pronto se popularizó como recurso para representar imágenes en soportes incapaces de procesar gráficos, como teletipos, terminales, correos electrónicos o algunas impresoras.

Aunque se puede componer arte ASCII manualmente mediante un editor de textos, también se pueden convertir automáticamente imágenes y vídeos en ASCII mediante software, como la librería Aalib (de licencia libre), que ha alcanzado cierta popularidad. Aalib está soportada por algunos programas de diseño gráfico, juegos y reproductores de vídeo.

1.9 Véase también

1.9.1 Variantes ASCII de ordenadores específicos

- ATASCII
- PETSCII

1.10 Referencias

1.10.1 Generales

- Unicode.org Cuadro Unicode de la zona ASCII
- Tom Jennings (29 de octubre de 2004). Historia anotada de los códigos de caracteres. Consultado 17 de diciembre de 2005.

1.10.2 Al pie

- [1] *Nombres de Dominio Internacionalizados - Glosario*, Internet Corporation for Assigned Names and Numbers (ICANN). Consultado el 19 de noviembre de 2008.
- [2] Organización Internacional para la Estandarización (1 de diciembre de 1975). "El conjunto de caracteres de ISO 646". *Internet Assigned Numbers Authority Registry*. Versión estadounidense: . Accedido el 7 de agosto de 2005.
- [3] Internet Assigned Numbers Authority (28 de enero de 2005). "Códigos de caracteres". Accedido el 7 de agosto de 2005.
- [4] ECMA International (diciembre de 1991). *Standard ECMA-6: 7-bit Coded Character Set, 6th edition* Accedido el 17 de diciembre de 2005.
- [5] Jargon File. *ASCIIbetical*. Accedido el 17 de diciembre de 2005.

1.11 Enlaces externos

- Herramienta en línea que muestra los caracteres ASCII y sus conversiones a otros sistemas numéricos.
- Conversión desde y hacia decimal, octal, hexadecimal, binario (ASCII notación)
- El código ASCII Tabla con los códigos ASCII simple y completa.

Capítulo 2

ASCII extendido



Un error típico en un terminal con letras verdes limitado a caracteres ASCII

Se denomina **ASCII extendido** a cualquier juego de caracteres de 8 bits en el cual los códigos 32 a 126 (0x20 a 0x7E) coinciden con los caracteres imprimibles de **ASCII**, así como los caracteres comúnmente llamados “de espacio”, estos son los códigos de control de 8 a 13 (0x08 a 0x0D), ambos inclusive.

Las codificaciones de ASCII extendido utilizan además parte o la totalidad de los códigos superiores a 128 para codificar caracteres adicionales a los caracteres imprimibles ASCII.

Codificaciones “ASCII extendido” más comunes:

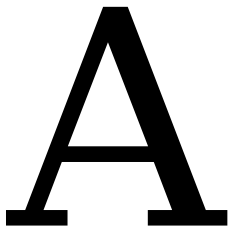
- Página de códigos 437 (usual en las versiones en inglés del IBM PC y MS-DOS)
- Página de códigos 850 (usual en las versiones de Europa occidental del IBM PC y MS-DOS)
- Latin-1 (ISO-8859-1) (típico de Unix y, con modificaciones, en Microsoft Windows y Macintosh)

2.1 Conjunto de caracteres ASCII Extendido del IBM PC y sucesores

Capítulo 3

Unicode

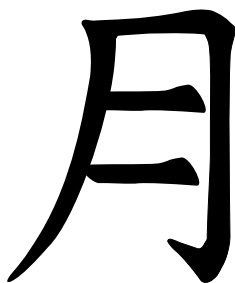
Ejemplos de caracteres Unicode



Carácter alfabético latino “A” (U+0041).



Sílaba devanagari “Aum” (ॐ) (U+0950).



Ideograma chino “yue” (月) (U+6708).

Unicode es un **estándar** de codificación de **caracteres** diseñado para facilitar el tratamiento informático, transmisión y visualización de textos de múltiples **lenguajes** y disciplinas técnicas, además de textos clásicos de **lenguas muertas**. El término Unicode proviene de los tres objetivos perseguidos: universalidad, uniformidad y unicidad.^[1]

Unicode especifica un nombre e identificador numérico único para cada carácter o símbolo, el *code point* (‘punto de código’), además de otras informaciones necesarias para su uso correcto: direccionalidad, mayúsculas y otros atributos. Unicode trata los caracteres alfabéticos, ideográficos y símbolos de forma equivalente, lo que significa que se pueden mezclar en un mismo texto sin la introducción de marcas o caracteres de control.^[2]

Este estándar es mantenido por el **Unicode Technical Committee** (UTC), integrado en el **Consortio Unicode**, del

que forman parte con distinto grado de implicación empresas como: Microsoft, Apple, Adobe, IBM, Oracle, SAP, Google o Yahoo, instituciones como la Universidad de Berkeley, y profesionales y académicos a título individual.^[3] El Unicode Consortium mantiene estrecha relación con ISO/IEC, con la que mantiene desde 1991 el acuerdo de sincronizar sus estándares que contienen los mismos caracteres y puntos de código.^[4]

El establecimiento de Unicode ha sido un ambicioso proyecto para reemplazar los esquemas de **codificación de caracteres** existentes, muchos de los cuales están muy limitados en tamaño y son incompatibles con entornos plurilingües. Unicode se ha vuelto el más extenso y completo esquema de codificación de caracteres, siendo el dominante en la **internacionalización** y adaptación local del **software informático**. El estándar ha sido implementado en un número considerable de tecnologías recientes, que incluyen XML, Java y **sistemas operativos** modernos.

La descripción completa del estándar y las tablas de caracteres están disponibles en la página web oficial de Unicode. La referencia completa se publica, además, en forma de libro impreso cada vez que se libera una nueva versión principal. La versión digital de este libro está disponible de forma gratuita. Las revisiones y adiciones se publican de forma independiente.

3.1 Alcance del estándar

Unicode incluye todos los caracteres de uso común en la actualidad. La versión 5.1 contenía 100 713 caracteres provenientes de alfabetos, sistemas ideográficos y colecciones de símbolos (matemáticos, técnicos, musicales, iconos...). La cifra crece con cada versión.

Unicode incluye sistemas de escritura modernos como: árabe, braille, copto, cirílico, griego, sinogramas (hanja coreano, hanzi chino y kanji japonés), silabarios japoneses (hiragana y katakana), hebreo y latino; escrituras históricas extintas, para propósitos académicos, como por ejemplo: cuneiforme, griego antiguo, lineal B micénico, fenicio y rúnico. Entre los caracteres no alfabéticos incluidos en Unicode se encuentran símbolos musicales y matemáticos, fichas de juegos como el dominó, flechas, iconos etc.

Además, Unicode incluye los **signos diacríticos** como caracteres independientes que pueden ser combinados con otros caracteres y dispone de versiones predefinidas de la mayoría de letras con símbolos diacríticos en uso en la actualidad, como las vocales acentuadas del español.

Unicode es un estándar en constante evolución y se agregan nuevos caracteres continuamente. Se han descartado ciertos alfabetos, propuestos por distintas razones, como por ejemplo el alfabeto klingon.^[5]

3.1.1 Relación con otros estándares

Como ya se ha indicado, Unicode está sincronizado con el estándar ISO/IEC conocido como UCS o juego de caracteres universal. Desde un punto de vista técnico, incluye o es compatible con codificaciones anteriores como ASCII7 o ISO 8859-1, los estándares nacionales ANSI Z39.64, KS X 1001, JIS X 0208, JIS X 0212, JIS X 0213, GB 2312, GB 18030, HKSCS, y CNS 11643, codificaciones particulares de fabricantes de software como Apple, Adobe, Microsoft, IBM, etc. Además, Unicode reserva espacio para fabricantes de software que pueden crear extensiones para su propio uso.^[6]

3.2 Repertorio de caracteres

El elemento básico del estándar Unicode es el carácter. Se considera un carácter al elemento más pequeño de un sistema de escritura con significado. El estándar Unicode codifica los caracteres esenciales —**grafemas**— definiéndolos de forma abstracta y deja la representación visual (tamaño, dimensión, fuente o estilo) al software que lo trate, como **procesadores de texto** o **navegadores web**. Se incluyen letras, signos diacríticos, caracteres de puntuación, ideogramas, caracteres silábicos, caracteres de control y otros símbolos. Los caracteres se agrupan en alfabetos o sistemas de escritura. Se considera que son diferentes los caracteres de alfabetos distintos, aunque compartan forma y significación.

Los caracteres se identifican mediante un **número o punto de código** y su nombre o descripción. Cuando se ha asignado un código a un carácter, se dice que dicho carácter está codificado. El espacio para códigos tiene 1 114 112 posiciones posibles (0x10FFFF). Los puntos de código se representan utilizando notación hexadecimal agregando el

prefijo U+. El valor hexadecimal se completa con ceros hasta 4 dígitos hexadecimales cuando es necesario; si es de longitud mayor que 4 dígitos no se agregan ceros.

3.2.1 Tipos de caracteres

Carácter	Código y nombre
Å	U+00C5 LATIN CAPITAL LETTER A WITH RING ABOVE
A + ◌̊	U+212B + U+030A LATIN CAPITAL LETTER + COMBINING RING ABOVE
Å	U+212B ANGSTROM SIGN

*Distintas versiones del carácter **angstrom**, como carácter (versión preferida), como carácter con signo diacrítico y como símbolo con forma de letra.*

Los bloques del espacio de códigos contienen puntos con la siguiente información:^[7]

- Caracteres gráficos: letras, signos diacríticos, cifras, caracteres de puntuación, símbolos y espacios.
- Caracteres de formato: caracteres invisibles que afectan al proceso del texto próximo. Ejemplos: U+2028 *salto de línea*, U+2029 *salto de párrafo*, U+00A0 *espacio duro*, etc.
- Códigos de control: 65 códigos definidos por compatibilidad con ISO/IEC 2022. Son los caracteres entre en los rangos [U+0000,U+001F], U+007F y [U+0080..U+009F]. Interpretarlos es responsabilidad de protocolos superiores.
- Caracteres privados: reservados para el uso fuera del estándar por fabricantes de software.
- Caracteres reservados: códigos reservados para su uso por Unicode. Son posiciones no asignadas.
- Puntos de código subrogados: Unicode reserva los puntos de código de U+D800 a U+DFFF para su uso como códigos subrogados en **UTF-16**, en la representación de caracteres suplementarios.
- No caracteres: son códigos reservados permanentemente para uso interno por Unicode. Los dos últimos puntos de cada plano U+FFFE y U+FFFF.
- Caracteres descartados: son caracteres que se retienen por compatibilidad con versiones anteriores, pero se debe evitar su uso.

3.2.2 Composición de caracteres y secuencias

Unicode incluye un mecanismo para formar caracteres y así extender el repertorio de compatibilidad con los símbolos existentes. Un carácter base se complementa con marcas: signos diacríticos, de puntuación o marcos. El tipo de cada carácter y sus atributos definen el papel que pueden jugar en una combinación. Por este motivo, puede haber varias opciones que representen el mismo carácter. Para facilitar la compatibilidad con codificaciones anteriores, se proporcionan caracteres precompuestos; en la definición de dichos caracteres se hace constar qué caracteres intervienen en la composición.

Un grupo de caracteres consecutivos, independientemente de su tipo, forma una secuencia. En caso de que varias secuencias representen el mismo conjunto de caracteres esenciales, el estándar no define una de ellas como 'correcta', sino que las considera equivalentes. Para poder identificar dichas equivalencias, Unicode define los mecanismos de *equivalencia canónica* y de *equivalencia de compatibilidad* basados en la obtención de formas normalizadas de las cadenas a comparar.



*Composición del carácter "ñ". La primera es un carácter independiente, la segunda una **n** más una tilde (virgulilla) combinable.*

3.2.3 Repertorio unificado chino, coreano y japonés

En el estándar Unicode, los ideogramas de Asia oriental (popularmente llamados «caracteres chinos») se denominan «ideogramas han». Estos ideogramas se desarrollaron en China y fueron adaptados por culturas próximas para su propio uso.^{[8][9]} Japón, Corea y Vietnam desarrollaron sus propios sistemas alfabéticos o silábicos para usar en combinación con los símbolos chinos: *hiragana* y *katakana* (en Japón), *hangul* (en Corea) y *yi* (en Vietnam). La evolución natural de los sistemas de escritura y los distintos momentos de entrada de los caracteres en las distintas culturas han marcado diferencias en los ideogramas utilizados. Unicode considera las distintas versiones de los ideogramas como variantes de un mismo carácter abstracto, es decir, como resultado de la aplicación de un *tipo de letra* diferente en cada caso y considera las variantes nacionales como pertenecientes a un mismo sistema de escritura. La versión original del estándar se desarrolló a partir de los estándares industriales existentes en los países afectados.

El organismo encargado de desarrollar el repertorio de caracteres es el Ideographic Rapporteur Group (IRG). IRG es un grupo de trabajo integrado en ISO/IEC JTC1/SC2/WG2, incluyendo a China, Hong Kong, Macao, Taipei Computer Association, Singapur, Japón, Corea del Sur, Corea del Norte, Vietnam y Estados Unidos de América.^[8]

La base de datos de caracteres CJK se denomina **UniHan** y contiene, además, información auxiliar sobre significado, conversiones, datos necesarios para utilizarlos en los diferentes lenguajes que los utilizan. A continuación se muestran los bloques que describen este repertorio. IRG define los caracteres de los tres grupos unificados, los siguientes dos grupos contienen caracteres para compatibilidad con estándares anteriores.

Secuencias de descripción ideográfica

Se admite que nunca se podrá finalizar la tarea de incluir ideogramas en el estándar debido, principalmente, a que la creación de nuevos ideogramas continúa. A fin de suplir eventuales carencias, Unicode ofrece un mecanismo que permite la representación de los símbolos que faltan denominado «secuencias de descripción ideográfica». Se basa en que en la práctica, la totalidad de los ideogramas se puede descomponer en piezas más pequeñas que, a su vez, son ideogramas. Aunque sea posible la representación de un símbolo mediante una secuencia, el estándar especifica que siempre que exista una versión codificada su uso debe ser preferente. No hay un método para la «descomposición canónica» de ideogramas ni algoritmos de equivalencia por lo que las operaciones sobre el texto, como búsqueda u ordenación, pueden fallar.

Unicode define 12 caracteres diferentes para la descripción de ideogramas representando distintas posibilidades de combinación espacial de otros caracteres han.

3.3 Elementos del estándar Unicode

3.3.1 Principios de diseño

El estándar fue diseñado con los siguientes objetivos:

- **Universalidad:** Un repertorio suficientemente amplio que albergue a todos los caracteres probables en el intercambio de texto multilingüe.

- **Eficiencia:** Las secuencias generadas deben ser fáciles de tratar.
- **No ambigüedad:** Un código dado siempre representa el mismo carácter.

3.3.2 Base de datos de caracteres

El conjunto de caracteres codificados por Unicode, es la UCD (unicode character database: base de datos de caracteres Unicode). Además de nombre y punto de código, incluye más información: alfabeto al que pertenece, nombre, clasificación, mayúsculas, orientación y otras formas de uso, variantes estandarizadas, reglas de combinación, etc.

Formalmente la base de datos se divide en **planos** y estos a su vez en **áreas** y **bloques**. Con excepciones, los caracteres codificados se agrupan en el espacio de códigos siguiendo categorías como alfabeto o sistema de escritura, de forma que caracteres relacionados se encuentren cerca en tablas de codificación.

Planos

Por conveniencia se ha dividido el espacio de códigos en grandes grupos denominados *planos*. Cada plano contiene un máximo de 65 535 caracteres. Dado un punto de código expresado en hexadecimal, los 4 últimos dígitos determinan la posición del carácter en el plano.

- Plano básico multilingüe: BMP o plano 0. Contiene la mayor parte de los alfabetos modernos, incluyendo los caracteres más comunes del sistema CJK, otros caracteres históricos o poco habituales y 64 reservadas para uso privado.
- Plano suplementario multilingüe: SMP o plano 1. Alfabetos históricos de menor uso y sistemas de uso técnico u otros usos.
- Plano suplementario ideográfico: SIP o plano 2. Contiene los caracteres del sistema CJK que no se incluyen en el plano 0. La mayoría son caracteres muy raros o de interés histórico.
- Plano de propósito especial: SSP o plano 14. Área para caracteres de control que no se han introducido en el plano 0.
- Planos de uso privado: planos 15 y 16. Reservados para uso privado por fabricantes de software.

Áreas y bloques

Los distintos planos se dividen en áreas de direccionamiento en función de los tipos generales que incluyen. Esta división es convencional, no reglada y puede variar con el tiempo. Las áreas se dividen, a su vez, en bloques. Los bloques están definidos normativamente y son rangos consecutivos del espacio de códigos. Los bloques se utilizan para formar las tablas impresas de caracteres pero no deben tomarse como definiciones de grupos significativos de caracteres.

3.4 Tratamiento de la información

3.4.1 Formas de codificación

Los puntos de código de Unicode se identifican por un número entero. Según su arquitectura, un ordenador utilizará unidades de 8, 16 o 32 bits para representar dichos enteros. Las *formas de codificación* de Unicode reglamentan la forma en que los puntos de código se transformarán en unidades tratables por el computador.

Unicode define tres formas de codificación bajo el nombre **UTF** (Unicode transformation format: formato de transformación Unicode).^[10]

- **UTF-8:** codificación orientada a byte con símbolos de longitud variable.
- **UTF-16:** codificación de 16 bits de longitud variable optimizada para la representación del plano básico multilingüe (BMP).

- **UTF-32**: codificación de 32 bits de longitud fija, y la más sencilla de las tres.

Las *formas de codificación* se limitan a describir el modo en que se representan los puntos de código en formato inteligible por la máquina. A partir de las 3 formas identificadas se definen 7 esquemas de codificación.

3.4.2 Esquemas de codificación

Los *esquemas de codificación* tratan de la forma en que se serializa la información codificada.^[10] La seguridad en los intercambios de información entre sistemas heterogéneos requiere la implementación de sistemas que permitan determinar el orden correcto de los bits y bytes y garantizar que la reconstrucción de la información es correcta. Una diferencia fundamental entre **procesadores** es el orden de disposición de los bytes en palabras de 16 y 32 bits, lo que se denomina *endianness*. Los esquemas de codificación deben garantizar que los extremos de una comunicación saben cómo interpretar la información recibida. A partir de las 3 formas de codificación se definen 7 esquemas. A pesar de que comparten nombres, no debe confundirse esquemas y formas de codificación.

Unicode define una marca especial, la **marca de orden de bytes** (BOM, *Byte Order Mark*), al inicio de un fichero o una comunicación para hacer explícita la ordenación de bytes. Cuando un protocolo superior especifica el orden de bytes, la marca no es necesaria y puede omitirse dando lugar a los esquemas de la lista anterior con sufijo *BE* o *LE*. En los esquemas UTF-16 y UTF-32, que admiten BOM, si este no se especifica se asume que la ordenación de bytes es *big-endian*.

La unidad de codificación en UTF-8 es el byte por lo que no necesita una indicación de orden de byte. El estándar ni requiere ni recomienda la utilización de BOM, pero lo admite como marca de que el texto es Unicode o como resultado de la conversión de otros esquemas.

3.5 Historia

El proyecto Unicode se inició a finales de 1987, tras conversaciones entre Joe Becker, Lee Collins y Mark Davis (ingenieros de las empresas **Apple** y **Xerox**).^[11] Como resultado de su colaboración, en agosto de 1988 se publicó el primer borrador de Unicode bajo el nombre de Unicode88.^[12] Esta primera versión, con códigos de 16 bits, se publicó asumiendo que solo se codificarían los caracteres necesarios para el uso moderno.

Durante el año 1989 el trabajo continuó con la adición de colaboradores de otras compañías como **Microsoft** o **Sun Microsystems**. El **Consortio Unicode** se formó el 3 de febrero de 1991, y en octubre de 1991 se publicó la primera versión del estándar. La segunda versión, incluyendo **escritura ideográfica han** se publicó en junio de 1992. A continuación se muestra una tabla con las distintas versiones del Estándar Unicode con sus adiciones o modificaciones más importantes.

3.6 Véase también

- Sistema de escritura
- Historia de la escritura

3.7 Referencias

- [1] «Resumen histórico». Unicode, Inc. Consultado el 21 de mayo de 2009.
- [2] «About the Unicode Standard». Unicode, Inc. Consultado el 21 de mayo de 2009.
- [3] «The Unicode Consortium Members». Unicode, Inc. Consultado el 15 de mayo de 2012.
- [4] The Unicode Consortium (octubre de 2006). «Appendix C. Relationship to ISO/IEC10646». En Julie D. Allen, Joe Becker (et al.). *Unicode 5.0 standard* (en inglés). Addison-Wesley. ISBN 0-321-48091-0.
- [5] «Archive of Notices of Non-Approval». Unicode, Inc. Consultado el 21 de mayo de 2009.

- [6] The Unicode Consortium (octubre de 2006). Julie D. Allen, Joe Becker (et al.), ed. *Unicode 5.0 standard* (en inglés). Addison-Wesley. ISBN 0-321-48091-0.
- [7] The Unicode Consortium (octubre de 2006). «16. Special Areas and Format Characters». En Julie D. Allen, Joe Becker (et al.). *Unicode 5.0 standard* (en inglés). Addison-Wesley. ISBN 0-321-48091-0.
- [8] «On the Encoding of Latin, Greek, Cyrillic, and Han».
- [9] «12. East Asian Scripts». *Unicode 5.0 Standard*.
- [10] The Unicode Consortium (octubre de 2006). «2.5 Encoding Forms». En Julie D. Allen, Joe Becker (et al.). *Unicode 5.0 standard* (en inglés). Addison-Wesley. ISBN 0-321-48091-0.
- [11] «Chronology of Unicode Version 1.0».
- [12] Becker, Joseph D. (10 de septiembre). *Unicode 88* (en inglés). Unicode Consortium. p. 10. Consultado el 29 de mayo de 2009.
- [13] The Unicode Consortium, Joan Aliprand, et al. (enero de 2000). «Appendix D. Changes from Unicode Version 2.0». *The Unicode Standard. Version 3.0 standard* (en inglés). Addison-Wesley. ISBN 0-201-61633-5.
- [14] The Unicode Consortium (octubre de 2006). «Appendix D. Changes from previous versions». En Julie D. Allen, Joe Becker (et al.). *Unicode 5.0 standard* (en inglés). Addison-Wesley. ISBN 0-321-48091-0.
- [15] Archivo de datos de Unicode 5.1
- [16] Unicode 5.2.0
- [17] Unicode 6.0.0

3.8 Enlaces externos

- Unicode Consortium, en el sitio web Unicode.org.
- Historia de la unificación han, en el sitio web Unicode.org.
- Catálogo de sistemas de escritura y hojas de caracteres, en el sitio web Unicode.org.

Capítulo 4

UTF-8

UTF-8 (8-bit *Unicode Transformation Format*) es un formato de codificación de caracteres **Unicode** e **ISO 10646** utilizando símbolos de longitud variable. UTF-8 fue creado por **Robert C. Pike** y **Kenneth L. Thompson**. Está definido como estándar por la **RFC 3629** de la *Internet Engineering Task Force* (IETF).^[1] Actualmente es una de las tres posibilidades de codificación reconocidas por Unicode y lenguajes web, o cuatro en **ISO 10646**.

Sus características principales son:

- Es capaz de representar cualquier carácter Unicode.
- Usa símbolos de longitud variable (de 1 a 4 bytes por carácter Unicode).
- Incluye la especificación **US-ASCII** de 7 bits, por lo que cualquier mensaje ASCII se representa sin cambios.
- Incluye sincronía. Es posible determinar el inicio de cada símbolo sin reiniciar la lectura desde el principio de la comunicación.
- No superposición. Los conjuntos de valores que puede tomar cada byte de un carácter multibyte, son disjuntos, por lo que no es posible confundirlos entre sí.

Estas características lo hacen atractivo en la codificación de correos electrónicos y páginas web.^{[2][3]} El **IETF** requiere que todos los protocolos de **Internet** indiquen qué **codificación** utilizan para los textos y que UTF-8 sea una de las codificaciones contempladas.^[4] El *Internet Mail Consortium* (IMC) recomienda que todos los programas de correo electrónico sean capaces de crear y mostrar mensajes codificados utilizando UTF-8.^[5]

4.1 Historia

UTF-8 fue ideado por **Kenneth L. Thompson** bajo los criterios de diseño de **Robert C. Pike** el 2 de septiembre de 1992. Ambos lo implementaron e implantaron en su sistema operativo *Plan 9 from Bell Labs*. Posteriormente fue oficialmente presentado en la conferencia **USENIX** en **San Diego** en enero de 1993. Fue promovido a estándar con el patrocinio de *X/Open Joint Internationalization Group* (XOJIG) y durante el proceso recibió diferentes nombres como FSS/UTF y UTF-2.^[1]

4.2 Descripción

UTF-8 divide los caracteres Unicode en varios grupos, en función del número de bytes necesarios para codificarlos. El número de bytes depende exclusivamente del código de carácter asignado por Unicode y del número de bytes necesario para representarlo. La distribución de caracteres es la siguiente:

- Caracteres codificados con un byte: Los incluidos en US-ASCII, un total de 128 caracteres.

- Caracteres codificados con dos bytes: Un total de 1920 caracteres. Este grupo incluye los caracteres **romances** más signos diacríticos, y los alfabetos **griego**, **cirílico**, **copto**, **armenio**, **hebreo**, **árabe**, **siríaco** y **Thaana** entre otros.
- Caracteres codificados con tres bytes: Caracteres del plano básico multilingüe de Unicode, que unido al grupo anterior, incluye la práctica totalidad de caracteres de uso común, entre ellos los caracteres del grupo **CJK**: Chino, japonés y coreano.
- Caracteres codificados con cuatro bytes: Caracteres del plano suplementario multilingüe. Símbolos matemáticos y alfabetos clásicos para uso principalmente académico: **Lineal B** silábico e ideográfico, alfabeto persa, fenicio... Y el plano suplementario ideográfico: caracteres **Han** de uso poco común.

Una propiedad importante de la codificación es que los bits más significativos del primer byte de una secuencia multi-byte determinan la longitud de la secuencia. Estos bits más significativos 110 para secuencias de dos bytes; 1110 para secuencias de tres bytes, etc. Estos bits además proporcionan la información de sincronía que permite identificar el inicio de un símbolo.

4.2.1 Codificación de los caracteres

La tabla siguiente muestra la forma en que se codifican los caracteres. Los valores fijos al principio de cada byte garantizan el cumplimiento del principio de no superposición, pues son distintos en función de la posición del byte en la cadena. Se incluye también la codificación UTF-16 para ver la diferencia con una codificación de número fijo de bytes.

Siguiendo el esquema anterior, sería posible incrementar el tamaño máximo del símbolo de 4 a 6 bytes. La definición de UTF-8 dada por Unicode no admite esta posibilidad que sí es admitida por ISO/IEC.^[6]

CODIFICACIÓN CON UTF-8

$$\tilde{n} \text{ (Unicode)} = \text{U+00F1}$$

$$\begin{aligned}
 &= \overset{0}{0} \overset{0}{0} \overset{F}{00} \overset{1}{1} \overset{1}{1} \overset{1}{000} 1 \\
 &\quad \swarrow \quad \searrow \\
 &110x \text{ xxxx} \quad 10xx \text{ xxxx} \\
 &\quad \swarrow \quad \searrow \\
 \tilde{n} \text{ (UTF-8)} &= 1100 \ 0011 \ 1011 \ 0001 = \text{C3B1}
 \end{aligned}$$

Ejemplo: Codificación del carácter Unicode ñ.

Veamos, a modo de ejemplo, cómo se codifica en UTF-8 el carácter eñe ('ñ'), que se representa en Unicode como 0x00F1:

- Su valor se sitúa en el rango de 0x0080 a 0x07FF. Una consulta a la tabla permite ver que debe ser codificado usando 2 bytes, con el formato *110xxxxx 10xxxxxx*.
- El valor **hexadecimal** 0x00F1 es equivalente al **binario** (0000-0)000-1111-0001 (los primeros 5 bits se ignoran, ya que no son necesarios para representar valores en el rango especificado).
- Los 11 bits requeridos se sitúan ordenados en la posición marcada por las equis: *11000011 10110001*.

- El resultado final son dos bytes con los valores hexadecimales 0xC3 0xB1. Ese es el código de la letra ñe en UTF-8.

Para recuperar el punto de código original se realiza el proceso inverso, descomponiendo las secuencias de bits en sus componentes y tomando solo los bits necesarios.

4.2.2 Errores de codificación

Las normas de codificación establecen, por lo tanto, límites a las cadenas que se pueden formar. Según la norma, un intérprete de cadenas debe rechazar como inválidos, y no tratar de interpretar, las caracteres mal formados. Un intérprete de cadenas UTF-8 puede cancelar el proceso señalando un error, omitir los caracteres mal formados o reemplazarlos por un carácter U+FFFD (*REPLACEMENT CHARACTER*).

Los siguientes son errores de codificación:

- Secuencias truncadas, cuando un carácter de inicio multi-byte no está seguido por suficientes bytes.
- Bytes de datos (comenzados por 10) sin el correspondiente inicio de carácter.
- Caracteres anómalamente largos: Por ejemplo, representar con 2 bytes un carácter del rango ASCII de un byte. Los bytes 0xC0, 0xC1 no se admiten.
- Bytes de inicio de carácter que especifican un largo anómalo de 5 o 6 bytes. Los bytes 0xF8 a 0xFD no se admiten.
- Valores fuera del rango Unicode: Los bytes 0xF5 y 0xF7 no se admiten.
- Caracteres inválidos. Los caracteres en el rango de pares subrogados de UTF-16, con código de 0xD800 a 0xDFFF, no son caracteres reales y no deben codificarse en UTF-8.

4.2.3 Byte order mark (BOM)

Cuando se sitúa al inicio de una cadena UTF-8, un carácter 0xFEFF, codificado en UTF-8 como 0xEF,0xBB,0xBF, se denomina *Byte Order Mark* (BOM) e identifica el contenido como una cadena de caracteres Unicode. Cuando este carácter se encuentra en otro lugar de la cadena debe ser interpretado con su significado original Unicode (ZWNBSP). Al ser UTF-8 una codificación en la que la unidad de información es el byte, no tiene la utilidad que sí tiene en UTF-16 y UTF-32 de identificar el orden de bytes en una palabra (*endianness*).

La especificación no recomienda o desaconseja la utilización de BOM, aunque sí desaconseja eliminarlo si existe como medida de seguridad, previniendo errores en aplicaciones de firma digital, etc. También advierte que debe ser eliminado en operaciones de concatenación para impedir que se mantenga en posiciones no iniciales.

4.3 Derivaciones de UTF-8

Las siguientes normas de codificación presentan diferencias con la especificación UTF-8 y son, por lo tanto, incompatibles con ella.

4.3.1 CESU-8

Esta implementación realiza una traducción directa de la cadena de caracteres representada con UTF-16 en lugar de codificar los puntos de código Unicode. El resultado es codificaciones diferentes para caracteres Unicode con código superior a 0xFFFF.^[1] Oracle, a partir de la versión 8, implementa CESU-8 con el alias *UTF8* y, a partir de la versión 9, UTF-8 estándar con otro alias.^[7] Java y Tcl utilizan esta codificación.^[cita requerida]

4.3.2 UTF-8 modificado

Con UTF-8 modificado, el carácter *nulo* se codifica como 0xC080 en lugar de 0x00. De esta forma un texto que contenga el carácter nulo no contendrá el byte 0x00 y, por lo tanto, no se truncará en lenguajes como C que consideran 0x00 un final de cadena.

Todas las implementaciones conocidas de UTF-8 modificado cumplen, además, con CESU-8^[cita requerida].

4.4 Ventajas y desventajas

Ventajas

- UTF-8 permite codificar cualquier carácter Unicode.^[1]
- Es compatible con US-ASCII, la codificación del repertorio de 7 bits es directa.
- Fácil identificación. Es posible identificar claramente una muestra de datos como UTF-8 mediante un sencillo algoritmo. La probabilidad de una identificación correcta aumenta con el tamaño de la muestra.^[1]
- UTF-8 ahorrará espacio de almacenamiento para textos en caracteres latinos, donde los caracteres incluidos en US-ASCII son comunes, cuando se compara con otros formatos como UTF-16.^[8]
- Una secuencia de bytes para un carácter jamás será parte de una secuencia más larga de otro carácter por contener información de sincronización.

Desventajas

- UTF-8 utiliza símbolos de longitud variable; eso significa que diferentes caracteres pueden codificarse con distinto número de bytes. Es necesario recorrer la cadena desde el inicio para encontrar el carácter que ocupa una determinada posición.
- 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.^[8]
- UTF-8 ofrece peor rendimiento que UTF-16 y UTF-32 en cuanto a coste de computación,^[8] por ejemplo en operaciones de ordenación.

4.5 Referencias

- [1] F. Yergeau (Noviembre 2003). «RFC 3629 - UTF-8, un formato de transformación de ISO 10646». Internet Society. Consultado el 20/05/2009.
- [2] «Moving to Unicode 5.1». Official Google Blog. 5 de mayo de 2008. Consultado el 20-05-2009.
- [3] Usage of character encodings for websites
- [4] H. Alvestrand (Enero 1998). «RFC 2277 - Política oficial de IETF sobre juegos de caracteres e idiomas». Internet Engineering Task Force. Consultado el 20/05/2009.
- [5] «Utilización de Caracteres Internacionales en el Correo de Internet». Internet Mail Consortium. 1 de agosto de 1998. Consultado el 20/5/2008.
- [6] The Unicode Consortium (octubre de 2006). «2.5 Encoding Forms». En Julie D. Allen, Joe Becker (et al.). *Unicode 5.0 standard* (en inglés). Addison-Wesley. ISBN 0-321-48091-0.
- [7] Simon Law (Mayo de 2005). «Globalization Support. Oracle Unicode database support.». Oracle Corporation. Consultado el 20/05/2009.
- [8] The Unicode Consortium (octubre de 2006). Julie D. Allen, Joe Becker (et al.), ed. *Unicode 5.0 standard* (en inglés). Addison-Wesley. ISBN 0-321-48091-0.

4.6 Véase también

- El estándar Unicode
- UTF-16
- UTF-32

4.7 Enlaces externos

- RFC 3629. Estándar UTF-8 (en inglés).
- Hello World Presentación de UTF-8 en *USENIX winter 1993* por Rob C. Pike y Ken Thompson (en inglés).
- Diseño de UTF-8 comentado por Robert C. Pike (en inglés).

Capítulo 5

UTF-16

UTF-16 que significa en **ISO/IEC 10646:2003**, “UCS Transformation Format for 16 Planes of Group 00.” es una forma de codificación de caracteres UCS y Unicode utilizando símbolos de longitud variable. Se halla oficialmente definido en el Anexo C de la norma ISO/IEC 10646:2003. También está descrita en el **Estándar Unicode** (versión 3.0 o superior), al igual que en la **RFC 2781** de la **IETF**.

Sus características principales son:

- Es capaz de representar cualquier carácter Unicode.
- Utiliza símbolos de longitud variable: 1 o 2 **palabras** de 16 bits por carácter Unicode (2 o 4 bytes). La unidad de información es la **palabra** de 16 bits.
- Está optimizado para representar caracteres en el *plano básico multilingüe* o BMP; caracteres en el rango U+0000 a U+FFFF. El BMP contiene la gran mayoría de caracteres y sistemas de escritura en uso en la actualidad. Cuando se limita al plano básico multilingüe, UTF-16 puede ser considerado una forma de codificación con símbolos de tamaño fijo (16 bits).
- No superposición: Los símbolos de 1 palabra (16 bits) utilizan un subconjunto de valores que no puede utilizarse en símbolos de 2 palabras (32 bits).

5.1 Historia

UTF-16 es la evolución de **UCS-2**, presente en el estándar Unicode hasta la versión 1.1. En UCS-2 cada punto de código se representa por su valor, lo que limitaba su uso al *plano básico multilingüe*. En la versión 2.0 del Estándar Unicode, la decisión de ampliar el espacio de códigos por encima del código FFFF supuso la necesidad de incluir un nuevo formato que diese soporte a los nuevos planos, 15 y 16, de uso privado. Sin embargo, el estándar no definió todavía ningún punto de código haciendo uso de este mecanismo hasta la versión 3.1.^{[1][2]}

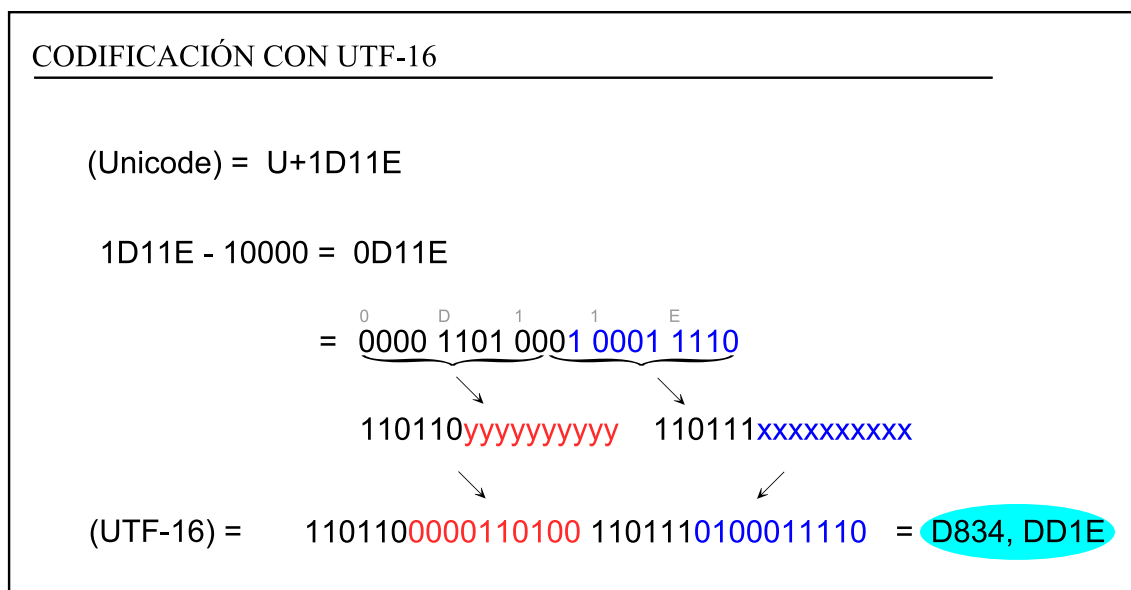
5.2 Descripción

En UTF-16 cada punto de código entre U+0000 y U+FFFF se codifica, sin cambios, utilizando 16 bits. Este rango se corresponde con el *plano básico multilingüe* de Unicode, por lo que la gran mayoría de los caracteres de uso común se codifican con 16 bits.

Los caracteres de los planos adicionales, se codifican mediante 32 bits. La codificación no se hace de forma directa, es decir, el código final no es el valor del punto de código. UTF-16 define un formato de transformación para estos casos denominado *pares subrogados*.

A la hora de valorar el espacio de almacenamiento requerido por un texto utilizando esta codificación, puede asumirse que los caracteres no incluidos en el plano básico son muy poco frecuentes y por lo tanto cada carácter utilizará 16 bits. Esta afirmación es válida también para el caso de las escrituras **CJK** (chino, japonés y coreano).

5.2.1 Pares subrogados



Ejemplo: Codificación del carácter *Unicode* U+1D11E, *Clave de sol*. El carácter está fuera del plano básico (BMP) y por lo tanto requiere el uso de pares subrogados.

Unicode e ISO/IEC han limitado el espacio asignable a los códigos hasta 10FFFF, por lo que se requiere un máximo de 21 bits para la representación de cualquier carácter. Si se usan dos palabras para representar códigos por encima de FFFF, se tiene un exceso de bits. Aprovechando esta circunstancia, se diseñó el sistema de pares subrogados para representar caracteres fuera del plano básico.

Se define un par subrogado como dos palabras de 16 bits donde:

- La primera palabra de 16 bits se denomina subrogado alto o subrogado inicial en terminología Unicode o *elementos RC de la zona media-alta* en terminología ISO/IEC. Toman valores en el rango [D800,DBFF].
- La segunda palabra de 16 bits se denomina subrogado bajo o subrogado final en terminología Unicode o *elementos RC de la zona media-baja* en terminología ISO/IEC. Toman valores en el rango [DC00,DFFF].
- Se cumple el principio de no superposición, los conjuntos valores de subrogados altos, bajos y códigos no subrogados son completamente disjuntos.

Para poder cumplir el principio de no superposición, el conjunto de valores utilizados como pares subrogados ha sido estandarizado y reservado, por lo que no puede ser utilizado para otros fines.

El sistema de pares subrogados se añadió en la versión 2.0 de Unicode, ISO 10646:xxxx, y por su diseño minimiza los conflictos que se pudieran presentar con implementaciones anteriores que no hagan uso de ellos.

5.2.2 Errores de codificación

Las normas de codificación establecen, por lo tanto, límites a las cadenas que se pueden formar. Según la norma, un intérprete de cadenas debe rechazar como inválidas, y no tratar de interpretar, las cadenas mal formadas.

- Un código subrogado inicial siempre debe ir seguido de un código subrogado final.
- Un código subrogado final siempre debe ir precedido por un código subrogado inicial.
- No se admite la codificación de caracteres inválidos. Los puntos de código U+FFFE, U+FFFF, y el rango U+FD00 a U+FDFF son puntos de códigos especiales que no representan caracteres y no deben codificarse en UTF-16. Tampoco se admiten los valores reservados fuera del plano básico.

5.2.3 Esquemas de codificación y BOM

A partir del formato de transformación UTF-16 se definen 3 esquemas de codificación. A pesar de que comparten nombres, no debe confundirse esquemas y formas de codificación. Los *esquemas de codificación* tratan de la forma en que se serializa la información codificada.^[3] La seguridad en los intercambios de información entre sistemas heterogéneos requiere la implementación de sistemas que permitan determinar el orden correcto de los bits y bytes y garantizar que la reconstrucción de la información es correcta. Una diferencia fundamental entre *procesadores* es el orden de disposición de los bytes en palabras de 16 y 32 bits, lo que se denomina *endianness*. Los esquemas de codificación deben garantizar que los extremos de una comunicación saben cómo interpretar la información recibida.

Unicode define una marca especial, *Byte order mark* o **BOM**, al inicio de un fichero o una comunicación para hacer explícita la ordenación de bytes. Esta marca es el carácter con punto de código U+FEFF. Cuando se encuentra en una posición inicial, puede ser interpretado como BOM dependiendo del contexto de la comunicación. En cualquier otra posición mantiene su semántica original como carácter *ZERO WIDTH NON-BREAKING SPACE*. Cuando un protocolo superior especifica el orden de bytes, la marca no es necesaria y puede omitirse dando lugar a los esquemas de la lista anterior con sufijo *BE* o *LE*. En el caso del esquema UTF-16, que admite BOM, si este no se especifica se asume que la ordenación de bytes es *big-endian*.

5.3 Véase también

- Unicode
- UTF-8
- UTF-32

5.4 Referencias

- [1] The Unicode Consortium, Joan Aliprand, et al. (enero de 2000). «Appendix D. Changes from Unicode Version 2.0». *The Unicode Standard. Version 3.0 standard* (en inglés). Addison-Wesley. ISBN 0-201-61633-5.
- [2] The Unicode Consortium (octubre de 2006). «Appendix D. Changes from previous versions». En Julie D. Allen, Joe Becker (et al.). *Unicode 5.0 standard* (en inglés). Addison-Wesley. ISBN 0-321-48091-0.
- [3] The Unicode Consortium (octubre de 2006). «2.5 Encoding Forms». En Julie D. Allen, Joe Becker (et al.). *Unicode 5.0 standard* (en inglés). Addison-Wesley. ISBN 0-321-48091-0.

Capítulo 6

ISO/IEC 8859

ISO/IEC 8859 es un conjunto ISO y la IEC estándar de 8 bits para codificaciones de caracteres para su uso en computadoras. La norma se divide en los números, publicado por separado, tales como ISO/IEC 8859-1, ISO/IEC 8859-2, etc, cada uno de los cuales puede ser informal a que se refiere como una norma en sí misma. En la actualidad hay 15 partes a partir de 2006, sin contar el abandonado ISO/ IEC 8859-12 estándar.

6.1 Tabla

6.2 Text and image sources, contributors, and licenses

6.2.1 Text

- **ASCII** *Fuente:* <http://es.wikipedia.org/wiki/ASCII?oldid=81326195> *Colaboradores:* PACO, Pit, Jmieres, Joseaperez, Moriel, Pablo.cl, Sauron, JorgeGG, ManuelGR, Rafael Soriano, Robbot, Angus, Zwobot, Comae, Dodo, Ascánder, Sms, Rsg, Tano4595, Murphy era un optimista, Barcex, Daniel G., Robotito, Jsl, Troodon, Mandramas, Suruena, Loco085, 142857, Quistnix, Renabot, LeonardoRob0t, Alexan, Petronas, Orgullo Moore, Airunp, JMPerez, Yrithinnd, Taichi, Emijrp, Rembiapo pohiyete (bot), Magister Mathematicae, Robot-Quistnix, Francosrodriguez, Platonides, Superzerocool, Chobot, Yrbot, Baifito, Seanver, BOT-Superzerocool, Oscar ., FlaBot, Vitamine, BOTijo, .Sergio, YurikBot, Icvav, GermanX, Sasquatch21, Beto29, Yanqui perdido, KnightRider, Gothmog, Jesuja, Ganon, Txo, Banfield, Dove, Maldoror, Er Komandante, KErosEnE, Paintman, Mirkovich, Aleator, Jstitch, BOTpolicia, Qwertyytrewqwwerty, Hawking, CEM-bot, Jorgelrm, -jem-, Ignacio Icke, GuiXu, Eli22, Nokeer, Gafotas, Montgomery, FrancoGG, Thijs!bot, RoyFocker, Gilberto IV, Knacr, Isha, Mpeinadopa, JAnDbot, Walterzum, Kved, Jalcaire, DerHexer, Mansoncc, Argeektect, Homo logos, Don Depresor, Muro de Aguas, Gaius iulius caesar, TXiKiBoT, MarquitoX, Gustronico, Elisardojm, Netito777, KanTagoff, Pólux, Manuel Trujillo Berges, Cinevoro, Zeist Antilles, Snakeyes, Technopat, Galandil, Penarc, Raystorm, Josell2, Matdrones, Liquid-aim-bot, Synthebot, BlackBeast, Lucien leGrey, AlleborgoBot, Muro Bot, SieBot, Mushii, Diego2891, Loveless, Cobalttempest, Cousteau, Drinibot, Bigsus-bot, BOTarate, Manwë, Greek, BuenaGente, Tirithel, Mutari, Jarisleif, Javierito92, Cambio, HUB, Kikobot, Estirabot, Eduardosalg, Arpotrek, Leonpolanco, Pan con queso, Alejandrocaro35, Botito777, Alexbot, Darkicebot, BodhisattvaBot, SilvononBot, UA31, AVBOT, DayL6, Dagane, Rayquazados, MarcoAurelio, Speedplus, Diegusjaimes, Davidgutierrezalvarez, CarsracBot, Capmo, Arjuno3, InflaBOT, Saloca, Madalberta, Luckas-bot, FariBOT, Super lol, Caf96, Billingham, Vivaelcelta, Peregrin08, Mono .lck, Nixón, SuperBraulio13, Manuelt15, Xqbot, Jkbw, Rubinbot, Dossier2, Josue arias silva, Lrainge, Botarel, Catalin586, AstaBOTH15, Kizar, DixonDBot, Robot8A, PatruBOT, Fran89, Mr.Ajedrez, Gustavo Girardelli, TjBot, Humbefa, Tarawa1943, Miss Manzana, EmausBot, Bachi 2805, Savh, AVIADOR, ChessBOT, Sergio Andres Segovia, JackieBot, Rubpe19, El Ayudante, Emiduronte, Chopinzone, Jatosfera, Antonorsi, SaeedVilla, Santanás va de retro, KLBOT2, Vagobot, BOT8A, Tximitx, Mega-buses, Elvisor, Creosota, DLeandroc, Helmy oved, Legobot, Chmarkine, FFraenz, Fernanda yepes, BenjaBot, Jade323, Trollerloler123 y Anónimos: 477
- **ASCII extendido** *Fuente:* <http://es.wikipedia.org/wiki/ASCII%20extendido?oldid=64408535> *Colaboradores:* Angus, Tano4595, Daniel G., Chlewey, Hari Seldon, Taichi, Rembiapo pohiyete (bot), BOT-Superzerocool, GermanX, Gothmog, Eloy, Gizmo II, Ignacio Icke, Thijs!bot, BotOni, Rei-bot, Dusan, AlleborgoBot, PaintBot, HUB, DiegoFb, Peregrin08, ZéroBot, MerlIwBot, KLBOT2 y Anónimos: 7
- **Unicode** *Fuente:* <http://es.wikipedia.org/wiki/Unicode?oldid=80218394> *Colaboradores:* Macar, Mac, Gwolf, Joseaperez, Sabbut, Moriel, JorgeGG, Mxn, ManuelGR, Aparejador, Zwobot, Interwiki, Rosarino, Faustito, Sms, Avm, Barcex, Daniel G., Mandramas, Almorca, Kordas, Benjavalero, FAR, Digigalos, Boticario, ICrash, Taichi, Andrés Cortina, Rembiapo pohiyete (bot), Marco Regueira, RobotQuistnix, Chobot, Yrbot, BOT-Superzerocool, Jamuki, FlaBot, YurikBot, Martingala, Carutsu, Eskimbot, Diegorodriguez, Maldoror, Chlewbob, Smrolando, Erufailon, Paintman, Ál, CEM-bot, AndresDominguez, D hanbun, Alexav8, Baiji, Dorio, Thijs!bot, Coiduras, Ñuño Martínez, Knacr, PhJ, Botones, JAnDbot, Juandesant, Rafa3040, Muro de Aguas, Flafus, TXiKiBoT, Xosema, R2D2!, Qu3tzalc0atl5, Pólux, LauraFarina, Fertejol, AlnoktaBOT, Cinevoro, VolkovBot, Technopat, Matdrones, Elabra sanchez, Synthebot, AlleborgoBot, Muro Bot, Racso, SieBot, Ctrl Z, PaintBot, Loveless, Obelix83, Bigsus-bot, RicardoPineda, LTB, PipepBot, Idleloop, StarBOT, Camilo Sanchez, Tosin2627, Eduardosalg, Pepbudes, Darkicebot, SilvononBot, Camilo, Albano Barcelona Caballero, Ente X, AVBOT, LucienBOT, JOe-LoFish, Diegusjaimes, MelancholieBot, Luckas-bot, MystBot, Jotterbot, Vic Fede, LyingB, Chuletadechanchon, FedericoF, AIXD, Xqbot, Jkbw, Bot0811, Botarel, MAfotBOT, TobeBot, Halfdrag, RedBot, Kizar, AnselmiJuan, PatruBOT, KamikazeBot, GrouchoBot, EmausBot, HRoestBot, Martindiego, Rufflos, MerlIwBot, UAwiki, LlamaAl, Albertao, Elvisor, Helmy oved, YFdyh-bot, ProfesorFavalli, JPaestpreonJeolhlna, Addbot, Dalbrok, Digmin3 y Anónimos: 103
- **UTF-8** *Fuente:* <http://es.wikipedia.org/wiki/UTF-8?oldid=79953018> *Colaboradores:* Gwolf, Moriel, ManuelGR, Zwobot, Javier Carro, Dodo, Ascánder, Sms, Netgrino, Truor, Cookie, Miernik, Daniel G., Rizox, Albertomendez, Mandramas, Jelko, 142857, Renabot, Ronaldo16, Juan Antonio Herguera Torres, Natrix, Rembiapo pohiyete (bot), Marco Regueira, Orgullobot, RobotQuistnix, Platonides, Chobot, Caierbot, Yrbot, FlaBot, YurikBot, Mortadelo2005, Martingala, Jgaray, LoquBot, Lobillo, Maldoror, CEM-bot, Alexav8, Coz, Ramonido, Botones, JAnDbot, Dvssolidaridad, Kved, Rafa3040, Juke, ColdWind, Millars, Humberto, Jotego, Emilioar 2000, Muro Bot, SieBot, Ensada, Bigsus-bot, LTB, Xqno, HUB, BodhisattvaBot, AVBOT, FerranJorba, Amrayo, MarcoAurelio, Luckas-bot, Xarxamedina, DSisyphBot, ArthurBot, Xqbot, Jkbw, Enter19, Halfdrag, RedBot, PatruBOT, Johnbot, Ralgisbot, Addbot, Chmarkine, Giovanni Alfredo Garcilano Diaz y Anónimos: 71
- **UTF-16** *Fuente:* <http://es.wikipedia.org/wiki/UTF-16?oldid=64568951> *Colaboradores:* Galio, Jasev, Mandramas, FAR, JMPerez, Marco Regueira, Orgullobot, RobotQuistnix, Yrbot, FlaBot, .Sergio, GermanX, No sé qué nick poner, CEM-bot, Thijs!bot, PabloCastellano, Botones, AlleborgoBot, Muro Bot, SieBot, Mixetmalo, Bigsus-bot, LTB, Alecs.bot, Luckas-bot, ArthurBot, Hprmedina, KamikazeBot, WikitanvirBot, Addbot y Anónimos: 10
- **ISO/IEC 8859** *Fuente:* <http://es.wikipedia.org/wiki/ISO/IEC%208859?oldid=70047198> *Colaboradores:* Escarbot, TXiKiBoT, SieBot, Diego2891, Diegusjaimes, Pbtogourou, Xqbot, Rubinbot, Dossier2, Jerowiki, KLBOT2, BOTito y Anónimos: 2

6.2.2 Images

- **Archivo:074_-_yue4_-_moon.svg** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/9/9e/074_-_yue4_-_moon.svg *Licencia:* CC BY-SA 2.5 *Colaboradores:* Trabajo propio *Artista original:* User:Magna
- **Archivo:ASCII_Code_Chart-Quick_ref_card.png** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/e/e0/ASCII_Code_Chart-Quick_ref_card.png *Licencia:* Public domain *Colaboradores:* See above description *Artista original:* Namazu-tron
- **Archivo:ASCII_full.svg** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/4/42/ASCII_full.svg *Licencia:* CC-BY-SA-3.0 *Colaboradores:* Own work. Created in Inkscape 0.44.1 *Artista original:* Arite
- **Archivo:Ambox_outdated_serious.svg** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/8/8f/Ambox_outdated_serious.svg *Licencia:* Public domain *Colaboradores:* Trabajo propio *Artista original:* penubag, Tkgd2007 made the clock
- **Archivo:Angstrom_unicode_sample.svg** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/1/11/Angstrom_unicode_sample.svg *Licencia:* Public domain *Colaboradores:* Trabajo propio *Artista original:* Marco Regueira

- **Archivo:Aum.svg** *Fuente:* <http://upload.wikimedia.org/wikipedia/commons/4/4d/Aum.svg> *Licencia:* Public domain *Colaboradores:* ? *Artista original:* ?
- **Archivo:Codificación_UTF-16.svg** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/4/4e/Codificaci%C3%B3n_UTF-16.svg *Licencia:* Public domain *Colaboradores:* Trabajo propio *Artista original:* Marco Regueira
- **Archivo:Codificación_UTF-8.svg** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/2/2c/Codificaci%C3%B3n_UTF-8.svg *Licencia:* Public domain *Colaboradores:* Trabajo propio *Artista original:* Marco Regueira
- **Archivo:Composicion_nh.svg** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/3/3f/Composicion_nh.svg *Licencia:* Public domain *Colaboradores:* Trabajo propio *Artista original:* Marco Regueira
- **Archivo:Cscr-featured.svg** *Fuente:* <http://upload.wikimedia.org/wikipedia/commons/e/e7/Cscr-featured.svg> *Licencia:* LGPL *Colaboradores:* Wikipedia until June, 2006 *Artista original:* Wikimedia users ClockworkSoul, CyberSkull, Optimager, White Cat, Erina, AzaToth, Pbroks13.
- **Archivo:Dumb_terminal_virus.png** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/2/25/Dumb_terminal_virus.png *Licencia:* Public domain *Colaboradores:* ? *Artista original:* ?
- **Archivo:Letter_A.svg** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/5/54/Letter_A.svg *Licencia:* CC BY-SA 3.0 *Colaboradores:* Trabajo propio *Artista original:* Neo@NHNG
- **Archivo:León_ASCII.JPG** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/2/24/Le%C3%B3n_ASCII.JPG *Licencia:* CC BY-SA 3.0 *Colaboradores:* Trabajo propio *Artista original:* G

6.2.3 Content license

- Creative Commons Attribution-Share Alike 3.0