Capítulo 3

Representación de datos textuales

Relacionados con el Tema 2 de la asignatura, la Guía de Aprendizaje establece estos resultados:

- Conocer los convenios de representación binaria, transmisión y almacenamiento de datos textuales, numéricos y multimedia.
- Conocer los principios de los algoritmos de detección de errores y de compresión.

En este capítulo veremos los convenios de representación de textos, en el capítulo 4 los de datos numéricos, y en el capítulo 5 los de datos multimedia. La detección de errores y la compresión se tratan en el capítulo 6. El capítulo 7 se dedica a los convenios de almacenamiento de los datos en ficheros.

Si vamos a tratar de «representación de textos» hay que empezar precisando qué es lo que queremos representar. Lo más sencillo es el *texto plano*: el formado por secuencias de **grafemas** (a los que, más familiarmente, llamamos *caracteres*), haciendo abstracción de su tamaño, color, tipo de letra, etcétera (que dan lugar a los **alógrafos** del grafema). Los grafemas son los caracteres visibles, pero el texto plano incluye también caracteres que no tienen una expresión gráfica y son importantes: espacio, tabulador, nueva línea, etcétera.

Empezaremos con la representación del texto plano. Para los distintos alógrafos los programas de procesamiento de textos utilizan convenios propios y muy variados (apartado 3.6), pero para el texto plano veremos que hay un número reducido de estándares. Y terminaremos el capítulo comentando una extensión de la representación textual muy importante en telemática: el hipertexto.

3.1. Codificación binaria

En un acto de comunicación se intercambian *mensajes* expresados en un lenguaje comprensible para los agentes que intervienen en el acto. Los mensajes de texto están formados por cadenas de caracteres. El conjunto de caracteres (o símbolos básicos) del lenguaje es lo que llamamos *alfabeto*. Para la comunicación entre personas los lenguajes occidentales utilizan un alfabeto latino. Pero si en el proceso de comunicación intervienen agentes artificiales los mensajes tienen que expresarse en su lenguaje, normalmente basado en un alfabeto binario.

La **codificación** es la traducción de un mensaje expresado en un determinado alfabeto, $\mathcal{A}$, al mismo mensaje expresado en otro alfabeto, $\mathcal{B}$. La traducción inversa se llama **descodificación** (o **decodificación**). Si los mensajes han de representarse en binario, entonces $\mathcal{B} = \{0,1\}$.

La codificación se realiza mediante un **código**, o correspondencia biunívoca entre los símbolos de los dos alfabetos. Como normalmente tienen diferente número de símbolos, a algunos símbolos de un alfabeto hay que hacerles corresponder una secuencia de símbolos del otro. Por ejemplo, si $\mathcal{A}$ = {a,b,c} y $\mathcal{B}$ = {0,1}, podríamos definir el código:

$\mathcal{A}$	$\mathcal{B}$
a	0
b	1
c	01

Ahora bien, tal código no sería útil, porque la descodificación es ambigua. Así, el mensaje codificado «011» puede corresponder a los mensajes originales «abb» o «cb». Si «a» se codifica como «00» el código ya no es ambiguo. Pero lo más sencillo es que el código sea de longitud fija, es decir, que el número de símbolos de $\mathcal{B}$ que corresponden a cada símbolo de $\mathcal{A}$ sea siempre el mismo. Podríamos establecer el código:

$\mathcal{A}$	$\mathcal{B}$
a	00
b	01
c	11

o bien:

$\mathcal{A}$	$\mathcal{B}$
a	01
b	00
c	10

o bien...

Los códigos BCD, ASCII y las normas ISO que veremos a continuación son de longitud fija. Pero UTF-8 (apartado 3.4) es de longitud variable. También veremos un ejemplo de codificación con longitud variable relacionado con la compresión en el apartado 6.5.

Códigos BCD

Codificar en binario un conjunto de m símbolos con un código de longitud fija requiere un número n de bits tal que $2^n \geq m$. Por tanto, para codificar los diez dígitos decimales hacen falta cuatro bits. Estos códigos se llaman **BCD** (decimal codificado en binario). El más común es el BCD «natural» (o BCD, a secas), en el que cada dígito decimal se representa por su equivalente en el sistema de numeración binario.

	BCD natural	**Aiken**	**exceso de 3**
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

Tabla 3.1: Tres códigos BCD.

Como cuatro bits permiten codificar dieciséis símbolos de los que sólo usamos diez, hay $V_{16,10} = 16!/6! \approx 2{,}9 \times 10^{10}$ códigos BCD diferentes. En la tabla 3.1 se muestran tres: el BCD «natural», u «8-4-2-1» (los números indican los pesos correspondientes a cada posición, es decir, que, por ejemplo, $1001 = 1\times8+0\times4+0\times1+1\times1 = 9$) y otros dos que presentan algunas ventajas para simplificar los algoritmos o los circuitos aritméticos: el código de Aiken, que es un código «2-4-2-1», y el llamado «exceso de 3».

3.2. Código ASCII y extensiones

En nuestra cultura se utilizan alrededor de cien grafemas para la comunicación escrita contando dígitos, letras y otros símbolos: ?, !, @, $, etc. Para su codificación binaria son necesarios al menos siete bits ($2^6 = 64; 2^7 = 128$). Para incluir letras con tilde y otros símbolos es preciso ampliar el código a ocho bits ($2^8 = 256$).

El **ASCII** (American Standard Code for Information Interchange) es una **norma**, o **estándar**, para representación de caracteres con *siete bits* que data de los años 1960 (ISO/IEC 646) y que hoy está universalmente adoptada. Las treinta y dos primeras codificaciones y la última no representan grafemas sino acciones de control. Por tanto, quedan $2^7 - 33 = 95$ configuraciones binarias con las que se codifican los dígitos decimales, las letras mayúsculas y minúsculas y algunos caracteres comunes (espacio en blanco, «.», «+», «–», «<», etc.), pero no letras con tilde, ni otros símbolos, como «¿», «¡», «ñ», «ç», «€», etc.

Hex.	Dec.		Hex.	Dec.		Hex.	Dec.		Hex.	Dec.		Hex.	Dec.		Hex.	Dec.	
20	032		30	048	0	40	064	@	50	080	P	60	096	‘	70	112	p
21	033	!	31	049	1	41	065	A	51	081	Q	61	097	a	71	113	q
22	034	"	32	050	2	42	066	B	52	082	R	62	098	b	72	114	r
23	035	#	33	051	3	43	067	C	53	083	S	63	099	c	73	115	s
24	036	$	34	052	4	44	068	D	54	084	T	64	100	d	74	116	t
25	037	%	35	053	5	45	069	E	55	085	U	65	101	e	75	117	u
26	038	&	36	054	6	46	070	F	56	086	V	66	102	f	76	118	v
27	039	’	37	055	7	47	071	G	57	087	W	67	103	g	77	119	w
28	040	(	38	056	8	48	072	H	58	088	X	68	104	h	78	120	x
29	041	)	39	057	9	49	073	I	59	089	Y	69	105	i	79	121	y
2A	042	*	3A	058	:	4A	074	J	5A	090	Z	6A	106	j	7A	122	z
2B	043	+	3B	059	;	4B	075	K	5B	091	[	6B	107	k	7B	123	{
2C	044	,	3C	060	<	4C	076	L	5C	092	\	6C	108	l	7C	124	|
2D	045	-	3D	061	=	4D	077	M	5D	093	]	6D	109	m	7D	125	}
2E	046	.	3E	062	>	4E	078	N	5E	094	^	6E	110	n	7E	126	~
2F	047	/	3F	063	?	4F	079	O	5F	095	_	6F	111	o	7F	127	DEL

Tabla 3.2: Código ASCII.

La tabla 3.2 muestra las codificaciones ASCII a partir de 0x20 expresadas en hexadecimal y en decimal. El decimal es útil por la forma que hay en algunos sistemas para introducir caracteres que no están en el teclado: Alt+<valor decimal>.

Algunos de los caracteres de control son:

0x00, NUL: carácter nulo (fin de cadena)

0x08, BS (backspace): retroceso

0x0A, LF (line feed): nueva línea

0x0D, CR (carriage return): retorno

0x1B, ESC (escape)

0x7F, DEL (delete): borrar

Extensiones

ASCII es «el estándar» de 7 bits, pero no es el único. Otro, también de 7 bits pero con cambios en los grafemas representados (introduce algunas letras con tilde), es el GSM 03.38, estándar para SMS en telefonía móvil.

La mayoría de las extensiones se han hecho añadiendo un bit (es decir, utilizando un byte para cada carácter), lo que permite ciento veintiocho nuevas configuraciones, conservando las codificaciones ASCII cuando este bit es 0. Se han desarrollado literalmente miles de códigos. Si no se lo cree usted,

pruebe el programa `iconv`, que traduce un texto de un código a otro. Escribiendo «`iconv -l`» se obtiene una lista de todos los códigos que conoce.

Un contemporáneo de ASCII, de ocho bits pero no compatible, que aún se utiliza en algunos grandes ordenadores *(mainframes)* es el EBCDIC (Extended Binary Coded Decimal Interchange Code).

Las extensiones a ocho bits más utilizadas son las definidas por las normas ISO 8859.

3.3. Códigos ISO

La ISO (International Organization for Standardization) y la IEC (International Electrotechnical Commission) han desarrollado unas normas llamadas «ISO 8859-X», donde «X» es un número comprendido entre 1 y 16 para adaptar el juego de caracteres extendido a diferentes necesidades:

- ISO 8859-1 (o «Latin-1»), para Europa occidental
- ISO 8859-2 (o «Latin-2»), para Europa central
- ISO 8859-3 (o «Latin-3»), para Europa del sur (Turquía, Malta y Esperanto)
- ...
- ISO 8859-15 (o «Latin-9»), revisión de 8859-1 en la que se sustituyen ocho símbolos poco usados por € y por algunos caracteres de francés, finés y estonio: Ž, Š, š, ž, Œ, œ, Ÿ
- ISO 8859-16 (o «Latin-10»), para Europa del sudeste (Albania, Croacia, etc.)

Todas ellas comparten las 128 primeras codificaciones con ASCII. La tabla 3.3 resume la norma ISO Latin-9. Las treinta y dos primeras codificaciones por encima de ASCII (0x80 a 0x8F) son también caracteres de control dependientes de la aplicación, y no se definen en la norma. «NBSP» (0xA0) significa «non-breaking space» (espacio inseparable) y su interpretación depende de la aplicación. En HTML, por ejemplo, se utilizan varios «` `» (o «` `», o «` `») seguidos cuando se quiere que el navegador presente todos los espacios sin colapsarlos en uno solo.

Hex.	Dec.		Hex.	Dec.		Hex.	Dec.		Hex.	Dec.		Hex.	Dec.		Hex.	Dec.	
A0	160	NBSP	B0	176	°	C0	192	À	D0	208	Đ	E0	224	à	F0	240	ð
A1	161	¡	B1	177	±	C1	193	Á	D1	209	Ñ	E1	225	á	F1	241	ñ
A2	162	¢	B2	178	²	C2	194	Â	D2	210	Ò	E2	226	â	F2	242	ò
A3	163	£	B3	179	³	C3	195	Ã	D3	211	Ó	E3	227	ã	F3	243	ó
A4	164	€	B4	180	Ž	C4	196	Ä	D4	212	Ô	E4	228	ä	F4	244	ô
A5	165	¥	B5	181	μ	C5	197	Å	D5	213	Õ	E5	229	å	F5	245	õ
A6	166	Š	B6	182	¶	C6	198	Æ	D6	214	Ö	E6	230	æ	F6	246	ö
A7	167	§	B7	183	·	C7	199	Ç	D7	215	×	E7	231	ç	F7	247	÷
A8	168	š	B8	184	ž	C8	200	È	D8	216	Ø	E8	232	è	F8	248	ø
A9	169	©	B9	185	¹	C9	201	É	D9	217	Ù	E9	233	é	F9	249	ù
AA	170	ª	BA	186	º	CA	202	Ê	DA	218	Ú	EA	234	ê	FA	250	ú
AB	171	«	BB	187	»	CB	203	Ë	DB	219	Û	EB	235	ë	FB	251	û
AC	172	¬	BC	188	Œ	CC	204	Ì	DC	220	Ü	EC	236	ì	FC	252	ü
AD	173	-	BD	189	œ	CD	205	Í	DD	221	Ý	ED	237	í	FD	253	ý
AE	174	®	BE	190	Ÿ	CE	206	Î	DE	222	Þ	EE	238	î	FE	254	þ
AF	175	¯	BF	191	¿	CF	207	Ï	DF	223	ß	EF	239	ï	FF	255	ÿ

Tabla 3.3: Códigos ISO 8859-15 que extienden ASCII.

Estos estándares son aún muy utilizados, pero paulatinamente están siendo sustituidos por Unicode. La ISO disolvió en 2004 los comités que se encargaban de ellos para centrarse en el desarrollo de Unicode, adoptado con el nombre oficial «ISO/IEC 10646».

3.4. Unicode

El Consorcio Unicode, en colaboración con la ISO, la IEC y otras organizaciones, desarrolla desde 1991 el código UCS (Universal Character Set), que actualmente contiene más de cien mil caracteres. Cada carácter está identificado por un nombre y un número entero llamado **punto de código**.

La primera versión (Unicode 1.1, 1991) definió el «BMP» (Basic Multilingual Plane). Es un código de 16 bits que permite representar $2^{16} = 65.536$ caracteres, aunque no todos los puntos de código se asignan a caracteres, para facilitar la conversión de otros códigos y para expansiones futuras. Posteriormente se fueron añadiendo «planos» para acomodar lenguajes exóticos, y la última versión (Unicode 6.2, 2012) tiene 17 planos, lo que da un total de $17 \times 2^{16} = 1.114.112$ puntos de código. De ellos, hay definidos 110.182 caracteres. Pero, salvo en países orientales, prácticamente sólo se utiliza el BMP.

Ahora bien, una cosa son los puntos de código y otra las codificaciones en binario de esos puntos. El estándar Unicode define siete formas de codificación. Algunas tienen longitud fija. Por ejemplo, «UCS-2» es la más sencilla: directamente codifica en dos bytes cada punto de código del BMP. Pero la más utilizada es «UTF-8», que es compatible con ASCII y tiene varias ventajas para el procesamiento de textos («UTF» es por «Unicode Transformation Format»; otras formas son UTF-16 y UTF-32).

UTF-8

La tabla 3.4 resume el esquema de codificación UTF-8 para el BMP. «U+» es el prefijo que se usa en Unicode para indicar «hexadecimal»; es equivalente a «0x».

Puntos	Punto en binario	Bytes UTF-8
U+0000 a U+007F	00000000 0xxxxxxx	0xxxxxxx
U+0080 a U+07FF	00000yyy yyxxxxxx	110yyyyy 10xxxxxx
U+0800 a U+FFFF	zzzzyyyy yyxxxxxx	1110zzzz 10yyyyyy 10xxxxxx

Tabla 3.4: Codificación en UTF-8.

Los 128 primeros puntos de código se codifican en un byte, coincidiendo con las codificaciones ASCII. Los siguientes 1.920, entre los que están la mayoría de los caracteres codificados en las normas ISO, necesitan dos bytes. El resto de puntos del BMP se codifican en tres bytes. Los demás planos (puntos U+10000 a U+10FFFF, no mostrados en la tabla) requieren 4, 5 o 6 bytes.

Algunos ejemplos se muestran en la tabla 3.5. Observe que:

- «E» está codificado en un solo byte, como en ASCII.

	Nombre Unicode	Punto de código	UTF-8
E	LATIN CAPITAL LETTER E	U+0045	0x45
ñ	LATIN SMALL LETTER N WITH TILDE	U+00F1	0xC3B1
€	EUROSIGN	U+20AC	0xE282AC
₧	PESETA SIGN	U+20A7	0xE282A7
₯	DRACHMA SIGN	U+20AF	0xE282AF
⇒	RIGHTWARDS DOUBLE ARROW	U+21D2	0xE28792
∞	INFINITY	U+2210	0xE2889E
∫	INTEGRAL	U+222B	0xE288AB

Tabla 3.5: Ejemplos de codificación en UTF-8.

- «ñ» no está en ASCII; el byte menos significativo de su punto de código coincide con la codificación en ISO Latin-9 (tabla 3.3), y, como puede usted comprobar, su codificación en UTF-8 se obtiene siguiendo la regla de la tabla 3.4.

- Sin embargo, «€», que también está en la tabla 3.3, tiene un punto de código superior a U+07FF y se codifica en tres bytes. El motivo es que los puntos U+00A0 a U+00FF son compatibles con ISO Latin-1, no con ISO Latin-9, y en el primero la codificación 0xA4 no está asignada al símbolo del euro, sino al símbolo monetario genérico, «¤».

3.5. Cadenas

Una «cadena» *(string)*, en informática, es una secuencia finita de símbolos tomados de un conjunto finito llamado alfabeto.

La representación de un texto plano, una vez elegido un código para la representación de los caracteres es, simplemente, la cadena de bytes que corresponden a la sucesión de caracteres del texto. A diferencia de otros tipos de datos, las cadenas tienen longitud variable. Un convenio que siguen algunos sistemas y lenguajes es indicar al principio de la cadena su longitud (un entero representado en uno o varios bytes). Pero el convenio más frecuente es utilizar una codificación de ASCII, 0x00 («NUL»), para indicar el fin de la cadena. Se les llama «cadenas C» *(C strings)* por ser el convenio del lenguaje C.

Como se trata de una sucesión de bytes, en principio no se plantea el dilema del «extremismo mayor o menor» (apartado 1.4): si el primer byte se almacena en la dirección d, el siguiente lo hará en $d + 1$ y así sucesivamente. Sin embargo:

- Esto es cierto para ASCII y para las normas ISO. Pero en UTF-16 el elemento básico de representación ocupa dos bytes (y en UTF-32, cuatro), y UTF-8 es de longitud variable, y el problema sí aparece. El consorcio Unicode ha propuesto un método que consiste en añadir al principio de la cadena una marca de dos bytes llamada «BOM» (Byte Order Mark) cuya lectura permite identificar si es extremista mayor o menor.

- Se suele ilustrar el conflicto del extremismo con el «problema NUXI», que tiene un origen histórico: en 1982, al transportar uno de los primeros Unix de un ordenador (PDP-11) a otro (IBM Series/1), un programa que debía escribir «UNIX» escribía «NUXI». La explicación es que para generar los cuatro caracteres se utilizaban dos palabras de 16 bits, y el problema surge al almacenar en memoria los dos bytes de cada palabra: el programa seguía el convenio del PDP-11 (extremista menor), mientras que el ordenador que lo ejecutaba era extremista mayor.

3.6. Texto con formato

Para los textos «enriquecidos» con propiedades de los caracteres (tamaño, tipo de letra o *«font»*, color, etc.) con el fin de representar los distintos alógrafos de los grafemas, así como para otros documentos que pueden incluir también imágenes y sonidos (presentaciones, hojas de cálculo, etc.), se han sucedido muchos estándares «*de facto*»: convenios para programas comerciales que durante algún tiempo se han impuesto en el mercado de la ofimática. Los ejemplos más recientes son los de la «suite» Microsoft Office, que incluye el programa de procesamiento de textos «Word» y su formato «.doc», el de presentaciones «PowerPoint» y su formato «.ppt» el de hojas de cálculo «Excel» y su formato «.xls», etc. Estos formatos son privativos (o «propietarios»), ligados a los programas comerciales que los usan. Recientemente, debido en parte a la presión de las administraciones europeas por el uso de formatos abiertos, se han extendido dos estándares oficiales:

- **ODF** (Open Document Format) es un estándar ISO (ISO/IEC 26300:2006) que incluye convenios para textos (ficheros «`.odt`»), presentaciones («`.odp`»), hojas de cálculo («`.ods`»), etc.

- **OOXML** (Office Open XML) es otro estándar (ISO/IEC IS 29500:2008) similar, promocionado por Microsoft. El convenio para los nombres de los ficheros es añadir la letra «x» a las extensiones de los antiguos formatos: «`.docx`», «`.pptx`», «`.xlsx`», etc.

Ambos comparten el basarse en un metalenguaje que mencionaremos luego, XML. Todo documento se representa mediante un conjunto de ficheros XML archivados y comprimidos con ZIP. Puede usted comprobarlo fácilmente: descomprima un documento cualquiera que esté en uno de estos formatos (aunque no tenga la extensión «.zip») y verá aparecer varios directorios («carpetas») y en cada directorio varios ficheros XML, ficheros de texto plano que puede usted leer (con `cat`, o con `more`, o con `less`...) y editar con cualquier editor de textos.

PDF *(Portable Document Format)* es otro formato muy conocido para distribución de documentos. En su origen también privativo, es actualmente un estándar abierto (ISO 32000-1:2008), y la empresa propietaria de las patentes (Adobe) permite el uso gratuito de las mismas, por lo que hay muchas implementaciones tanto comerciales como libres. Es un formato complejo, muy rico en tipografías y elementos gráficos matriciales y vectoriales.

3.7. Hipertexto

Un «hipertexto» es un documento de texto acompañado de anotaciones, o marcas, que indican propiedades para la presentación (tamaño, color, etc.), o remiten a otras partes del documento o a otros documentos (hiperenlaces, o simplemente, «enlaces», para abreviar). En la red formada por los enlaces no solo hay documentos de texto: también se enlazan ficheros con imágenes, sonidos y vídeos, de modo que el concepto de «hipertexto» se generaliza a «hipermedia».

Se trata de un concepto que actualmente es bien conocido por ser el fundamento de la web. Para su implementación se utiliza el lenguaje HTML (HyperText Markup Language). A los efectos de almacenamiento y de transmisión, un documento HTML (que se presenta como una página web) es texto plano. El intérprete del lenguaje incluido en el navegador reconoce las marcas y genera la presentación adecuada.

3.8. Documentos XML

A diferencia de HTML, cuyo objetivo es la *presentación* de datos en un navegador, XML (eXtensible Markup Language) está diseñado para la *representación* de datos a efectos de almacenamiento y comunicación entre agentes. Pero XML es más que un lenguaje: es un **metalenguaje**. Quiere esto decir que sirve para definir lenguajes concretos, como ODF y OOXML, o como SVG, un lenguaje cuyo objetivo es la representación de gráficos (apartado 5.3).

XML también está basado en marcas para definir las reglas sintácticas del lenguaje concreto y para construir aplicaciones con ese lenguaje. Y, al igual que HTML, un documento XML es un fichero de texto plano. En principio, la codificación es UTF-8, pero también admite otras codificaciones, señalándolo al principio del documento para hacérselo saber a la aplicación que va a procesarlo. Así, la primera línea de un documento XML podría ser:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Si se omite la información de «encoding» se entiende que es UTF-8.

La descripción de XML como lenguaje y la forma de definir lenguajes específicos sería muy interesante, pero desborda los objetivos de este capítulo, que se limita a los convenios de representación de textos, sin entrar en su contenido.