Codificación de datos

Hemos visto que la memoria de las modernas computadoras digitales está diseñada para almacenar información en forma de bits, agrupados en conjuntos de a ocho, y que cada una de estas unidades de ocho bits se llama un byte. Usando la analogía natural entre valores de bits (inactivo/activo) y dígitos binarios (0 o 1), hemos visto cómo la memoria de una computadora digital moderna puede almacenar números no negativos entre 0 y 255.

Sin embargo, los problemas de la vida real involucran otras clases de datos. Muchos problemas requerirán manipular números mayores que 255, o números negativos, o con decimales; o aun, datos que no sean numéricos, como texto, imágenes, sonido, o video. Si la única manera posible de guardar contenidos en la memoria de la computadora es en forma de bits, ¿cómo podremos manipular estas otras clases de datos?

Codificación de texto

Cuando escribimos texto en nuestra computadora, estamos almacenando temporariamente en la memoria una cierta secuencia de caracteres, que son los símbolos que tipeamos en nuestro teclado. Estos caracteres tienen una **representación gráfica** en nuestro teclado, en la pantalla o en la impresora, pero mientras están en la memoria **no pueden ser otra cosa que bytes**, es decir, conjuntos de ocho dígitos binarios. Para lograr almacenar caracteres de texto necesitamos adoptar una **codificación**, es decir, una tabla que asigne a cada carácter un patrón de bits fijo.

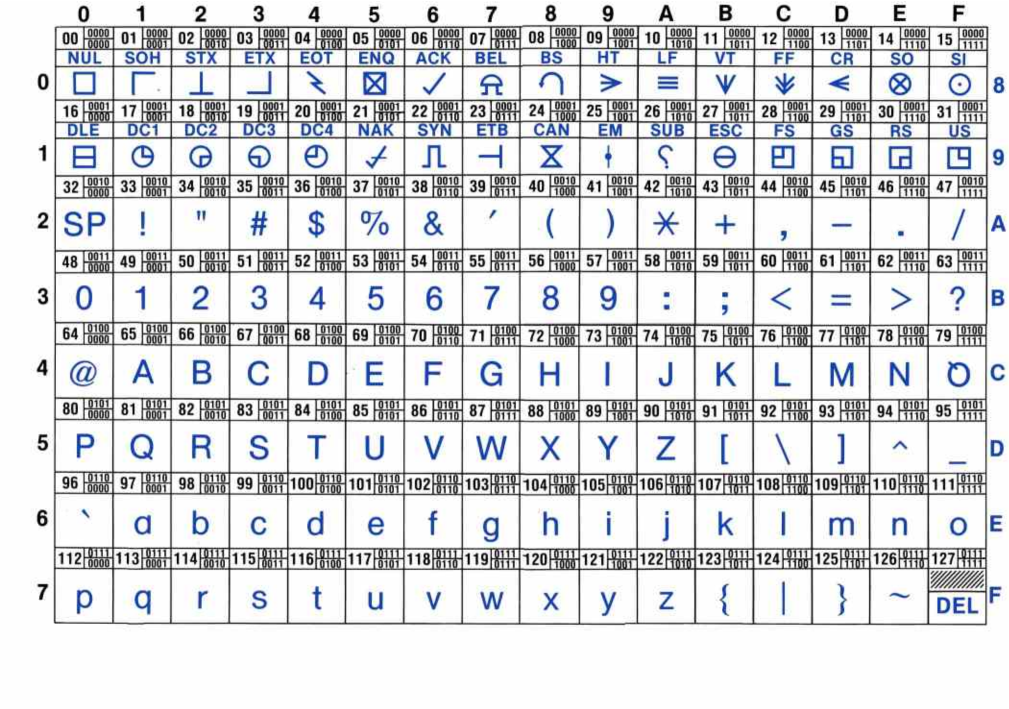
Además, esta codificación debe ser **universal**: para poder compartir información entre usuarios, o entre diferentes aplicaciones, se requiere algún estándar que sea respetado por todos los usuarios y las aplicaciones. Inicialmente se estableció con este fin el **código ASCII**[1], que durante algún tiempo fue una buena solución. El código ASCII asigna patrones de siete bits a un conjunto de caracteres que incluye:

- Las 25 letras del alfabeto inglés, mayúsculas y minúsculas;
- Los dígitos del 0 al 9,
- Varios símbolos matemáticos, de puntuación, etc.,
- El espacio en blanco,
- Y 32 caracteres no imprimibles. Estos caracteres no imprimibles son combinaciones de bits que no tienen una representación gráfica, sino que sirven para diversas funciones de comunicación de las computadoras con otros dispositivos.

En general, prácticamente todos los símbolos que figuran en nuestro teclado tienen un código ASCII asignado. Como sólo se usan siete bits, el bit de mayor orden (el de más a la izquierda) de cada byte siempre es cero, y por lo tanto los códigos ASCII toman valores de 0 a 127.

[1] http://es.wikipedia.org/wiki/ASCII

Tabla ASCII, que codifica los caracteres más importantes utilizando siete bits (el bit más significativo de cada ocho es siempre cero).



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	00 0000 0000 NUL	01 0000 0001 SOH	02 0000 0010 STX	03 0000 0011 ETX	04 0000 0100 EOT	05 0000 0101 ENQ	06 0000 0110 ACK	07 0000 0111 BEL	08 0000 1000 BS	09 0000 1001 HT	10 0000 1010 LF	11 0000 1011 VT	12 0000 1100 FF	13 0000 1101 CR	14 0000 1110 SO	15 0000 1111 SI	8
1	16 0001 0000 DLE	17 0001 0001 DC1	18 0001 0010 DC2	19 0001 0011 DC3	20 0001 0100 DC4	21 0001 0101 NAK	22 0001 0110 SYN	23 0001 0111 ETB	24 0001 1000 CAN	25 0001 1001 EM	26 0001 1010 SUB	27 0001 1011 ESC	28 0001 1100 FS	29 0001 1101 GS	30 0001 1110 RS	31 0001 1111 US	9
2	32 0010 0000 SP	33 0010 0001 !	34 0010 0010 "	35 0010 0011 #	36 0010 0100 $	37 0010 0101 %	38 0010 0110 &	39 0010 0111 '	40 0010 1000 (	41 0010 1001)	42 0010 1010 *	43 0010 1011 +	44 0010 1100 ,	45 0010 1101 -	46 0010 1110 .	47 0010 1111 /	A
3	48 0011 0000 0	49 0011 0001 1	50 0011 0010 2	51 0011 0011 3	52 0011 0100 4	53 0011 0101 5	54 0011 0110 6	55 0011 0111 7	56 0011 1000 8	57 0011 1001 9	58 0011 1010 :	59 0011 1011 ;	60 0011 1100 <	61 0011 1101 =	62 0011 1110 >	63 0011 1111 ?	B
4	64 0100 0000 @	65 0100 0001 A	66 0100 0010 B	67 0100 0011 C	68 0100 0100 D	69 0100 0101 E	70 0100 0110 F	71 0100 0111 G	72 0100 1000 H	73 0100 1001 I	74 0100 1010 J	75 0100 1011 K	76 0100 1100 L	77 0100 1101 M	78 0100 1110 N	79 0100 1111 O	C
5	80 0101 0000 P	81 0101 0001 Q	82 0101 0010 R	83 0101 0011 S	84 0101 0100 T	85 0101 0101 U	86 0101 0110 V	87 0101 0111 W	88 0101 1000 X	89 0101 1001 Y	90 0101 1010 Z	91 0101 1011 [	92 0101 1100 \	93 0101 1101]	94 0101 1110 ^	95 0101 1111 _	D
6	96 0110 0000 `	97 0110 0001 a	98 0110 0010 b	99 0110 0011 c	100 0110 0100 d	101 0110 0101 e	102 0110 0110 f	103 0110 0111 g	104 0110 1000 h	105 0110 1001 i	106 0110 1010 j	107 0110 1011 k	108 0110 1100 l	109 0110 1101 m	110 0110 1110 n	111 0110 1111 o	E
7	112 0111 0000 p	113 0111 0001 q	114 0111 0010 r	115 0111 0011 s	116 0111 0100 t	117 0111 0101 u	118 0111 0110 v	119 0111 0111 w	120 0111 1000 x	121 0111 1001 y	122 0111 1010 z	123 0111 1011 {	124 0111 1100 |	125 0111 1101 }	126 0111 1110 ~	127 0111 1111 DEL	F

Sin embargo, el código ASCII es insuficiente para muchas aplicaciones: no contempla las necesidades de diversos idiomas. Por ejemplo, nuestra letra **ñ** no figura en la tabla ASCII. Tampoco las vocales acentuadas, ni con diéresis, como tampoco decenas de otros caracteres de varios idiomas europeos. Peor aún, con solamente 256 posibles patrones de bits, es imposible representar algunos idiomas orientales como el chino, que utilizan miles de ideogramas.

Por este motivo se estableció más tarde una familia de nuevos estándares, llamada **Unicode**[2]. Uno de los estándares de codificación definidos por Unicode, el más utilizado actualmente, se llama **UTF-8**[3]. Este estándar mantiene la codificación que ya empleaba el código ASCII para ese conjunto de caracteres, pero agrega códigos de dos, tres y cuatro bytes para otros símbolos. El resultado es que hoy, con UTF-8, se pueden representar todos los caracteres de cualquier idioma conocido.

Codificación de multimedia

Otras clases de datos, diferentes del texto, también requieren codificación (porque siempre deben ser almacenados en la memoria en forma de bits y bytes), pero su tratamiento es diferente. Introducir en la computadora, por ejemplo, una **imagen analógica** (tal como un dibujo o una pintura hecha a mano), o un fragmento de **sonido** tomado del ambiente, requiere un proceso previo de digitalización. **Digitalizar** es **convertir en digital** la información que es **analógica**, es decir, convertir un rango continuo de valores (lo que está en la naturaleza) a un conjunto discreto de valores (que puede ser ingresado en la computadora).

Imagen En el caso de una imagen analógica, el proceso de digitalización involucra la división de la imagen en una fina cuadrícula, donde cada elemento de la cuadrícula abarca un pequeño sector cuadrangular de la imagen. A cada pequeño sector, o elemento cuadrangular, se le asignan valores discretos que codifican el color de la imagen en ese lugar. Por ejemplo, se pueden asignar tres valores, codificando la cantidad de rojo, de verde y de azul que contiene cada lugar de la imagen. Luego cada uno de estos valores se puede expresar como un entero. Si fuéramos a guardar cada uno de estos enteros en un byte, quedaríamos limitados a valores entre 0 y 255.

- Un elemento que fuera completamente rojo tendría valores (255, 0, 0). Un elemento de color verde puro tendría valores (0, 255, 0).

[2] http://es.wikipedia.org/wiki/Unicode
[3] http://es.wikipedia.org/wiki/Utf-8

- Un elemento blanco tendrá los máximos valores para todos los colores, que mezclados dan el color blanco: (255, 255, 255).
- Un elemento que es gris tendrá los tres valores aproximadamente iguales pero menores que 255.

En general, mientras más elementos podamos codificar, mejor será la aproximación a nuestra pieza de información original. Mientras más fina la cuadrícula (es decir, mientras mayor sea la **resolución**[4] de la imagen digitalizada), y mientras más valores discretos usemos para representar los colores, más se parecerá nuestra versión digital al original analógico.

Notemos que la digitalización de una imagen implica la discretización de dos **variables analógicas**, o sea, discretización en dos sentidos: por un lado, los infinitos puntos de la imagen analógica, bidimensional, deben reducirse a unos pocos rectángulos discretos. Por otro lado, los infinitos valores de color deben reducirse a sólo tres coordenadas, con finitos valores en el rango de nuestro esquema de codificación. Son dos ejemplos de conversión de variables analógicas a digitales. Este proceso de digitalización es el que hacen automáticamente una cámara de fotos digital o un celular, almacenando luego los bytes que representan la imagen tomada. Sin embargo, las modernas cámaras utilizan un esquema de codificación con mucha mayor **profundidad de color**[5] (es decir, más bytes por cada coordenada de color) que en el ejemplo anterior.

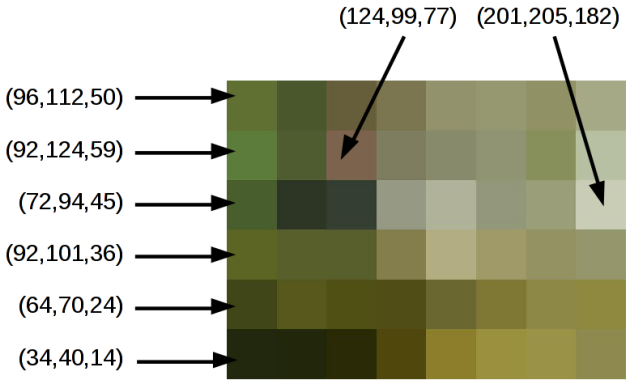


Imagen en baja resolución con 24 bits de profundidad de color.

- *La primera columna tiene predominancia de verde (la segunda componente es la mayor).*
- *Colores más oscuros tienen valores menores.*
- *El elemento que más tiende al rojo tiene la primera componente mayor.*
- *El elemento gris claro tiene las tres componentes aproximadamente iguales, y de valor alto.*

[4] http://es.wikipedia.org/wiki/Resolución_de_imágenes
[5] http://es.wikipedia.org/wiki/Profundidad_de_color

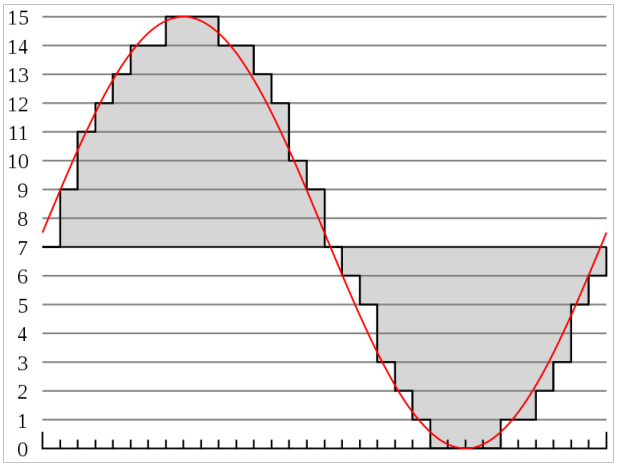
La misma imagen digital, en varias resoluciones

Sonido En el caso del **sonido** (o **audio**), también tenemos dos variables analógicas para digitalizar. Un micrófono actúa como un **transductor**: traduce los impulsos físicos del sonido, que llegan por el aire, a impulsos eléctricos. Necesitamos discretizar esos impulsos eléctricos, y como en el caso anterior, esto ocurrirá en dos sentidos. Por un lado, los impulsos eléctricos, provocados en el micrófono por el sonido, ocupan un intervalo continuo de tiempo, pero nos quedaremos sólo con unos cuantos valores por segundo. Por otro lado, en cada intervalo de tiempo, los impulsos eléctricos entregados por el micrófono ocupan un rango continuo de valores, entre ciertos valores eléctricos extremos; pero necesitamos quedarnos sólo con algunos valores enteros en ese rango.

Nuevamente, mientras más muestras por segundo y más valores diferentes reconozcamos en el rango de entrada, mejor se aproximará nuestra digitalización del sonido al original (ver figura). Aproximadamente así funcionan al captar nuestra voz los celulares, una cámara filmadora digital, o un programa de grabación de

sonido que usamos en nuestra computadora. El resultado es una secuencia de bytes que codifican el sonido que ingresó por el micrófono.

Digitalización de un ciclo de una onda sonora en 16 niveles.



Archivos

Una vez que se ha obtenido la secuencia de bytes que representa la información, ya se trate de texto, de multimedia o de otros tipos de datos, si no queremos perder esta información necesitamos guardarla en algún medio de almacenamiento permanente, como un disco rígido, o un pendrive, creando un **archivo.** Este archivo **es precisamente esa secuencia de bytes**, almacenados en algún medio de almacenamiento.

El archivo, que contiene y transporta nuestra información, ocupará una cierta cantidad de bytes en ese medio de almacenamiento, y, dependiendo de la información que guarda, puede tratarse de grandes cantidades de bytes. Si, por ejemplo, se trata de una imagen de muy alta **resolución** (donde la cuadrícula de digitalización es sumamente fina) y con gran **profundidad de color** (donde los colores son representados con muchos valores discretos, necesitando muchos bytes), entonces la imagen digitalizada será más grande. Es típico que las cámaras digitales modernas entreguen imágenes digitales con una resolución de decenas de millones de puntos o **pixels**[6], y estas imágenes suelen ser de muy gran tamaño[7].

Múltiplos del bit y del byte

Para manejar con comodidad esas grandes cantidades de bits y de bytes recurrimos a múltiplos, tal como se hace habitualmente con los metros para grandes distancias (en cuyo caso hablamos de kilómetros, u otras unidades), o con los gramos para grandes pesos (en cuyo caso hablamos de kilogramos, u otras unidades).

Hay dos sistemas de múltiplos de bits y bytes en uso, y a veces esto causa confusión: el Sistema Internacional (SI) y el sistema de Prefijos Binarios.

Sistema Internacional (SI) El **Sistema Internacional (SI)**[8] define múltiplos para todas las unidades de medida. Estos múltiplos se denominan mediante prefijos que acompañan a la unidad fundamental de cada magnitud (como, por ejemplo, **kilo** acompaña a **gramo** para crear el múltiplo **kilogramo**). En el SI, estos prefijos corresponden a múltiplos elegidos como potencias de 10. Los múltiplos que suelen ser más interesantes para las ciencias y las ingenierías corresponden a aquellas potencias de 10 cuyo exponente es múltiplo de 3 (como 10^3, 10^6, 10^9...).

[6] http://es.wikipedia.org/wiki/Pixel
[7] http://www.360cities.net/london-photo-es.html: ¡Una fotografía panorámica de miles de millones de pixels!
[8] http://es.wikipedia.org/wiki/Sistema_Internacional_de_Unidades#Tabla_de_m.C3.BAltiplos_y_subm.C3.BAltiplos

Por ejemplo, 10^3 (que es igual a 1000) se asocia con el prefijo **kilo** y se simboliza **k**, de modo que kilogramo significa mil gramos y se escribe 1 kg. Del mismo modo, 10^6 recibe el prefijo **mega** (simbolizado por **M**), 10^9 recibe el prefijo **giga** (simbolizado por **G**), y 10^{12} recibe el prefijo **tera** (simbolizado por **T**). De esta manera, **un millón de bytes** se indica como **1 megabyte** y se escribe **1MB**.

Prefijos Binarios Sin embargo, en computación es habitual medir la información con otros múltiplos que no son potencias de 10 sino potencias de 2. Por ejemplo, para ciertos usos es conveniente considerar los bytes en conjuntos de 1024 (que es 2^{10}) y no en conjuntos de 1000 (que es 10^3). Por este motivo una organización de estándares, la **IEC**, creó un conjunto de múltiplos basados en **Prefijos Binarios**[9]. Estos múltiplos son cantidades que tienen tamaños similares a los múltiplos del SI y son simbolizados con las mismas letras, pero son potencias de 2 cuyo exponente es múltiplo de 10 (como 2^{10}, 2^{20}, 2^{30}...). Para construir los prefijos del sistema de Prefijos Binarios, se agrega la sílaba **bi** luego de la primera sílaba del prefijo SI que los representa.

Así, la unidad binaria más cercana al kilobyte (kB) es el **Kibibyte** (**KiB**), que vale 2^{10} bytes (o sea, 1024 bytes, y no 1000). La unidad binaria más cercana al megabyte (MB) es el **Mebibyte** (**MiB**) que vale 2^{20} bytes (o sea, 1048576 bytes, y no exactamente un millón). La unidad binaria más cercana al gigabyte (GB) es el **Gibibyte** (**GiB**) que vale 2^{30} bytes (1073741824 bytes, y no exactamente mil millones). Existen otros múltiplos mayores, que por el momento no tienen uso diario, pero que de acuerdo con las tendencias tecnológicas, iremos encontrando con mayor frecuencia en el futuro cercano (**Tera, Peta, Exa, Zetta, Yotta...**).

En computación se utilizan, en diferentes situaciones, ambos sistemas de unidades, porque es costumbre usar el SI para hablar de **velocidades de transmisión de datos**, pero usar Prefijos Binarios al hablar de **almacenamiento**. Así, cuando un proveedor de servicios de Internet ofrece un enlace de **1Mbps**, nos está diciendo que por ese enlace podremos transferir **exactamente 1 millón de bits por segundo**. Por el contrario, los fabricantes de **medios de almacenamiento** (como memorias, discos rígidos o pendrives) acostumbran hablar en términos de múltiplos binarios, y por lo tanto deberían (aunque normalmente no lo hacen) utilizar **Prefijos Binarios** para expresar las capacidades de almacenamiento de esos medios. Así, un "pendrive de cuatro gigabytes", que normalmente tiene una capacidad de 4 * 2^{30} bytes, debería publicitarse en realidad como "pendrive de cuatro Gibibytes".

Compresión

Muchas veces es interesante reducir el tamaño de un archivo, para que ocupe menos espacio de almacenamiento o para que su transferencia a través de una red sea más rápida. Al ser todo archivo una secuencia de bytes, y por lo tanto de números, disponemos de métodos y herramientas matemáticas que permiten, en ciertas condiciones, reducir ese tamaño. La manipulación de los bytes de un archivo con este fin se conoce como **compresión.**

La compresión de un archivo se ejecuta mediante un programa que utiliza un algoritmo especial de compresión. Este algoritmo puede ser de **compresión sin pérdida, o con pérdida**[10].

Compresión sin pérdida Decimos que la compresión ha sido sin pérdida cuando, del archivo comprimido, puede extraerse exactamente la misma información que antes de la compresión, utilizando otro algoritmo que ejecuta el trabajo inverso al de compresión. En otras palabras, la compresión sin pérdida es **reversible**: siempre puede volverse a la información de partida. Esto es un requisito indispensable cuando necesitamos recuperar exactamente la secuencia de bytes original, como en el caso de un archivo de texto. Como usuarios de computadoras, es muy probable que hayamos utilizado más de una vez la compresión sin pérdida, al tener que comprimir un documento de texto que nosotros hicimos utilizando un programa utilitario como ZIP[11], RAR u otros. Si la compresión no fuera reversible, no podríamos recuperar el archivo de texto tal cual lo escribimos.

Compresión con pérdida En algunos casos, el resultado de la compresión de un archivo es otro archivo del cual **ya no puede recuperarse** la misma información original, pero que de alguna manera sigue sirviendo a los fines del usuario. Es el caso de la compresión de imágenes, donde se reduce la calidad de la imagen, ya sea utilizando menos colores, o disminuyendo la resolución. También es el caso de la compresión de audio, al descartar componentes del sonido con frecuencias muy bajas o muy altas, inaudibles para los humanos (como en la tecnología de grabación de CDs), con lo cual la diferencia entre el original digital y el comprimido no

[9] http://es.wikipedia.org/wiki/Prefijo_binario
[10] http://es.wikipedia.org/wiki/Digitalización#Compresión
[11] http://es.wikipedia.org/wiki/Formato_de_compresión_ZIP

es perceptible al oído. También es útil, para algunos fines, reducir la calidad del audio quitando componentes audibles (lo que hacen, por ejemplo, algunos grabadores "de periodista" para lograr archivos más pequeños, con audio de menor fidelidad, pero donde el diálogo sigue siendo comprensible).