Contenidos

- Datos vs Información
- Código binario
- Sistemas numéricos
  - Conversión de binario a decimal (BIN → DEC)
  - Conversión de decimal a binario (DEC → BIN)
  - Conversión de binario a hexadecimal (BIN → HEX)
  - HEX → BIN y BIN ↔ OCT
  - Cantidad de números con n dígitos y una base B
- Unidades de medida en Informática
  - Almacenamiento
  - Frecuencia
  - Tasa de transferencia
  - Tasa de transferencia de un bus
- Codificación de datos no numéricos
  - Texto
  - Imágenes
  - Audio
  - Vídeo

Información y unidades de medida

¿Qué es la Informática?

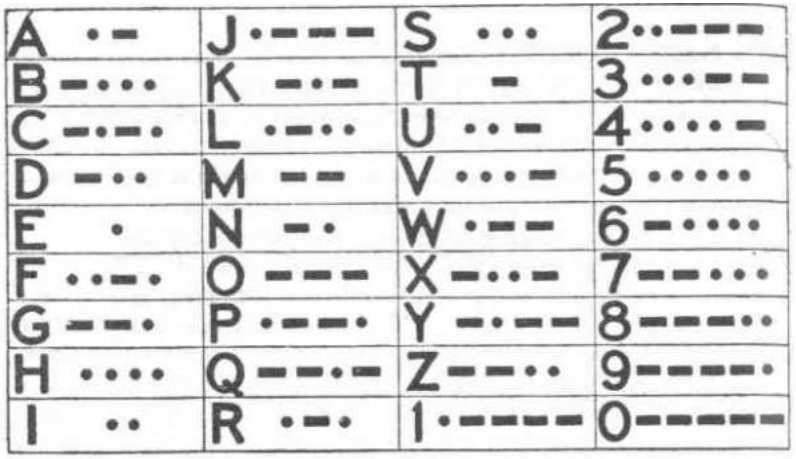
- La informática es la disciplina que se refiere al **tratamiento o procesamiento automático de la información mediante el uso de ordenadores**
- Para ayudarnos en el tratamiento automático de la información un ordenador nos ayuda a:
  - **Almacenarla** (Discos duros, DVDs, pendrives, ...)
  - **Organizarla** (Por ej. la organizamos en archivos distribuidos en carpetas)
  - **Procesarla**
  - **Transmitirla** (El ordenador nos la trasmite a través del monitor pero también nos permite difundirla por la red (email, web, mensajería, ...))
- El término viene de la unión de dos términos:
  - **INFOR**MACIÓN
  - AUTO**MÁTICA**

Datos vs Información

- **Datos**
  - Es una representación simbólica (numérica, alfabética, …) de un atributo, como por ejemplo el precio de un artículo, el peso, el nombre, ...
- **Información**
  - Es el resultado de **procesar los datos** para obtener un **mensaje que proporciona un conocimiento**
- Los datos en si no proporcionan información, deben pasar por un procesado previo
- Ejemplo. Tenemos los datos de ventas por mes de nuestra empresa de los dos últimos años:
  - El dato individual de un único mes no nos dice nada, nos falta perspectiva
  - Si examinamos conjuntamente los datos de todos los meses podremos ver:
    - Qué meses vendemos mucho y cuales poco
    - Si la tendencia de ventas es al alza o a la baja
    - Si un mes hemos vendido mas o menos que el mismo mes del año pasado, …
  - Para llegar a estas conclusiones hacemos un procesado de los datos para lo cual un ordenador nos ayuda de múltiples formas como por ejemplo facilitando la visualización de estos datos de una formá gráfica con la que resulta más fácil analizar los datos.

Codificación de los datos

- Los **datos son codificados**, tanto para ser almacenados como transmitidos
- El **acto de codificar** consiste en representar los datos con símbolos que se combinan de acuerdo a una serie de reglas predefinidas.
- Ejemplos de símbolos para representar datos son:
  - El **abecedario** → Lo utilizamos para codificar datos que sean palabras y texto. El lenguaje me determina las reglas de combinación de las letras
  - El **código morse** → Utiliza dos símbolos que con múltiples combinaciones codifica cada letra del abecedario y los números
- En ocasiones la codificación se tiene que adaptar a las limitaciones del medio por el que se transmite o se almacenan, como por ejemplo el código morse que es un código sencillo que se adapta a medios de transmisión de capacidad muy limitada (Pitidos en las antiguas líneas telefónicas y luces en barcos)



A ·−	J ·−−−	S ···	2 ··−−−
B −···	K −·−	T −	3 ···−−
C −·−·	L ·−··	U ··−	4 ····−
D −··	M −−	V ···−	5 ·····
E ·	N −·	W ·−−	6 −····
F ··−·	O −−−	X −··−	7 −−···
G −−·	P ·−−·	Y −·−−	8 −−−··
H ····	Q −−·−	Z −−··	9 −−−−·
I ··	R ·−·	1 ·−−−−	0 −−−−−

Representación de los datos en un ordenador

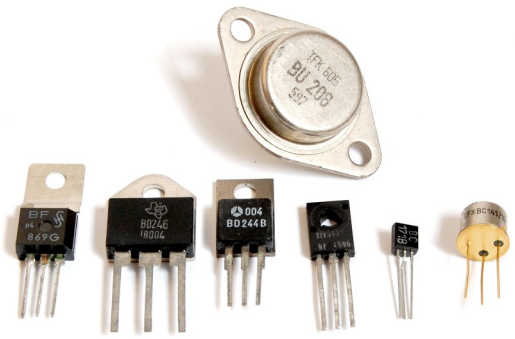
- Un ordenador utiliza el **código binario** para representar la información
- El código binario es un **sistema numérico** formado **por dos dígitos**, el 0 y 1
- Los 0's y 1's se combinan para formar letras, palabras, números, … de una forma análoga a como hace el código morse

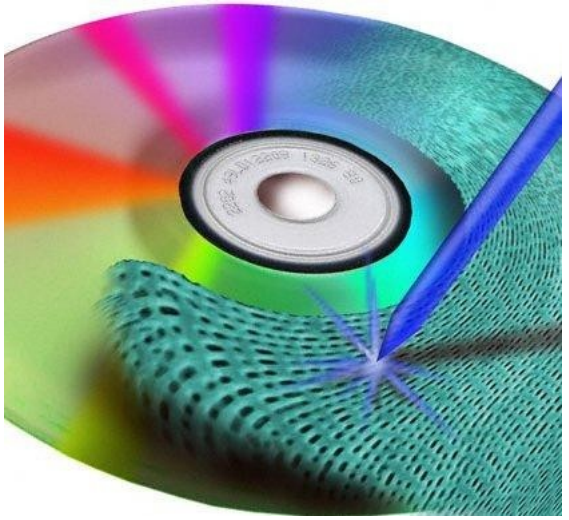
A	01000001	N	01001110	a	01100001	n	01101110
B	01000010	O	01001111	b	01100010	o	01101111
C	01000011	P	01010000	c	01100011	p	01110000
D	01000100	Q	01010001	d	01100100	q	01110001
E	01000101	R	01010010	e	01100101	r	01110010
F	01000110	S	01010011	f	01100110	s	01110011
G	01000111	T	01010100	g	01100111	t	01110100
H	01001000	U	01010101	h	01101000	u	01110101
I	01001001	V	01010110	i	01101001	v	01110110
J	01001010	W	01010111	j	01101010	w	01110111
K	01001011	X	01011000	k	01101011	x	01111000
L	01001100	Y	01011001	l	01101100	y	01111001
M	01001101	Z	01011010	m	01101101	z	01111010

El código binario en un ordenador

En Informática la forma de representar el 0 y 1 varía dependiendo del dispositivo. Vemos algunos ejemplos:

- En un circuito electrónico se representan mediante **impulsos eléctricos** con carga (1) o la ausencia de esta (0) gracias entre otros al uso de transistores
- En un CD o DVD se representan con dos tipos de **"marcas" sobre la superficie** del disco que hacen que haz láser del lector se refleje de una forma u otra
- En los antiguos dispositivos magnéticos como los disquetes el 0 y el 1 se representaban en la superficie del disco magnético aplicando **polaridades magnéticas** opuestas

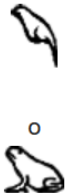




Sistema numérico de numeración

Dijimos que el código binario es un sistema numérico, ¿pero que es eso exactamente?

- Es un conjunto de **símbolos y reglas** que nos permite representar todos los números válidos
- El número de **símbolos es finito** y suelen llamarse dígitos
- Los sistemas de numeración pueden ser:
  - **Posicionales** → En estos el valor a representar depende de los símbolos utilizados y su posición
  - **No posicionales** → El valor a representar no depende de la posición de los símbolos. Es el utilizado por sistemas de numeración muy antiguos, como la numeración egipcia

Valor	1	10	100	1.000	10.000	100.000	1 millón, o infinito
Jeroglífico						 o	
Descripción	Bastón.	Asa o grillete.	Cuerda enrollada.	Flor de loto.	Dedo.	Renacuajo o rana.	Heh: hombre sentado con las manos alzadas.

Sistemas numéricos posicionales

- Nosotros utilizamos sistemas numéricos posicionales que están principalmente caracterizados por:
  - La **base**, que determina **el número de símbolos** que podemos utilizar en dicho sistema
  - Entre los dígitos se establece un orden de menor a mayor
  - **Vamos apilando dígitos en columnas y para pasar al siguiente orden de magnitud añadimos nuevos dígitos por la izquierda**. Cuando llegamos al dígito más alto en todas las columnas añadimos una columna adicionales por la parte izquierda con el símbolo más bajo
  - El dígito más a la izquierda es denominado más significativo y el que está más a la derecha menos significativo
- En informática nos interesa conocer 4 sistemas numéricos y como realizar la conversión entre ellos, principalmente **entre binario, decimal y hexadecimal**

Sistema	Base	Dígitos
Binario	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Conversión entre los cuatro sistemas

DEC	BIN	OCT	HEX
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Conversión más allá del 16

DEC	BIN	OCT	HEX	DEC	BIN	OCT	HEX	DEC	BIN	OCT	HEX
0	0	0	0	36	100100	44	24	72	1001000	110	48
1	1	1	1	37	100101	45	25	73	1001001	111	49
2	10	2	2	38	100110	46	26	74	1001010	112	4A
3	11	3	3	39	100111	47	27	75	1001011	113	4B
4	100	4	4	40	101000	50	28	76	1001100	114	4C
5	101	5	5	41	101001	51	29	77	1001101	115	4D
6	110	6	6	42	101010	52	2A	78	1001110	116	4E
7	111	7	7	43	101011	53	2B	79	1001111	117	4F
8	1000	10	8	44	101100	54	2C	80	1010000	120	50
9	1001	11	9	45	101101	55	2D	81	1010001	121	51
10	1010	12	A	46	101110	56	2E	82	1010010	122	52
11	1011	13	B	47	101111	57	2F	83	1010011	123	53
12	1100	14	C	48	110000	60	30	84	1010100	124	54
13	1101	15	D	49	110001	61	31	85	1010101	125	55
14	1110	16	E	50	110010	62	32	86	1010110	126	56
15	1111	17	F	51	110011	63	33	87	1010111	127	57
16	10000	20	10	52	110100	64	34	88	1011000	130	58
17	10001	21	11	53	110101	65	35	89	1011001	131	59
18	10010	22	12	54	110110	66	36	90	1011010	132	5A
19	10011	23	13	55	110111	67	37	91	1011011	133	5B
20	10100	24	14	56	111000	70	38	92	1011100	134	5C
21	10101	25	15	57	111001	71	39	93	1011101	135	5D
22	10110	26	16	58	111010	72	3A	94	1011110	136	5E
23	10111	27	17	59	111011	73	3B	95	1011111	137	5F
24	11000	30	18	60	111100	74	3C	96	1100000	140	60
25	11001	31	19	61	111101	75	3D	97	1100001	141	61
26	11010	32	1A	62	111110	76	3E	98	1100010	142	62
27	11011	33	1B	63	111111	77	3F	99	1100011	143	63
28	11100	34	1C	64	1000000	100	40	100	1100100	144	64
29	11101	35	1D	65	1000001	101	41	101	1100101	145	65
30	11110	36	1E	66	1000010	102	42	102	1100110	146	66
31	11111	37	1F	67	1000011	103	43	103	1100111	147	67
32	100000	40	20	68	1000100	104	44	104	1101000	150	68
33	100001	41	21	69	1000101	105	45	105	1101001	151	69
34	100010	42	22	70	1000110	106	46	106	1101010	152	6A
35	100011	43	23	71	1000111	107	47	107	1101011	153	6B

Conversión de binario a decimal (BIN → DEC)

- Se multiplica cada dígito del número binario por 2 elevado a la potencia consecutiva, comenzando en la derecha por la potencia 2^0
- Después de realizar cada una de las potencias y multiplicaciones, se suman todas y el número resultante será el equivalente al sistema decimal

Bit	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Valor	128 (2^7)	**64** (2^6)	32 (2^5)	16 (2^4)	8 (2^3)	**4** (2^2)	**2** (2^1)	**1** (2^0)
Número		1	0	0	0	1	1	1

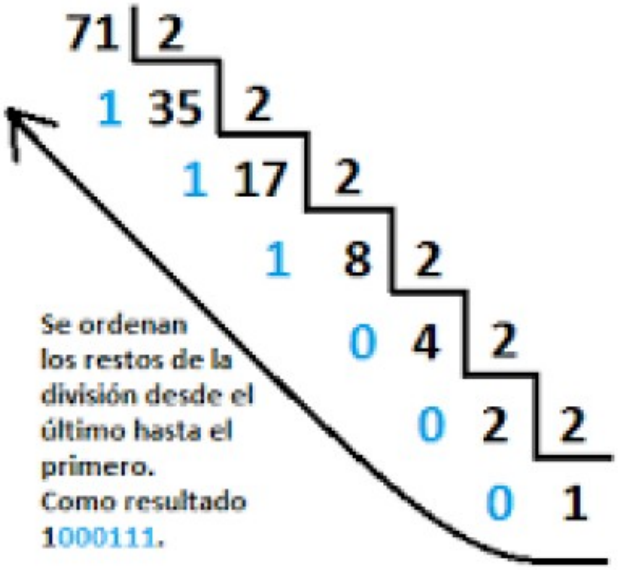
x2 ← x2 ← x2 ← x2 ← x2 ← x2 ← x2

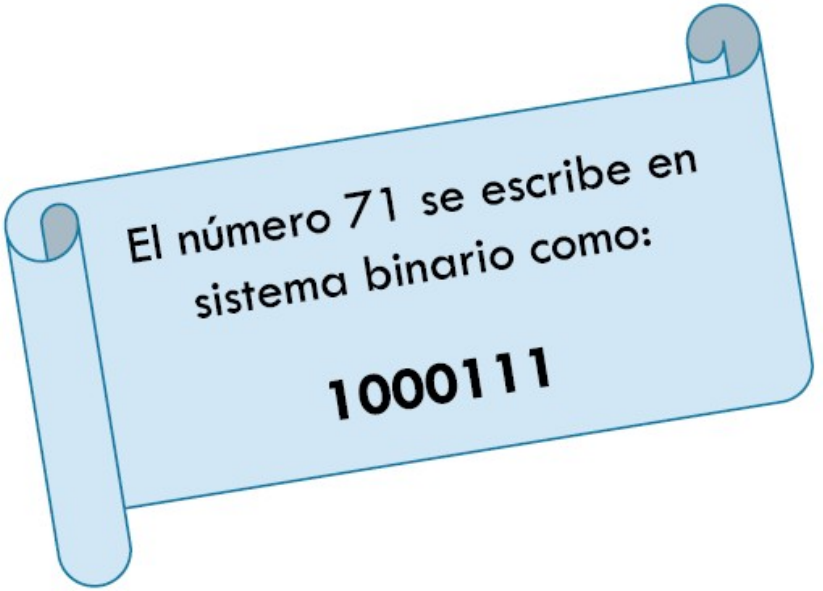
¿Qué número decimal es el **1000111**?

64+4+2+1=71

Conversión de decimal a binario (DEC → BIN)

- Se divide el número del sistema decimal entre 2 y su resultado entero se vuelve a dividir entre 2. Esto se hace sucesivamente hasta que el dividendo sea menor que el divisor, 2. Es decir, cuando el número a dividir sea 1 finaliza la división.
- A continuación se concatenan desde el último cociente hasta el primer resto. Este será el número binario que buscamos.





Conversión de decimal a binario (DEC → BIN)

- Vemos otro ejemplo de conversión para pasar el número 77 a binario:

División	Cociente (Parte entera)	Resto
77/2	38	**1**
38/2	19	**0**
19/2	9	**1**
9/2	4	**1**
4/2	2	**0**
2/2	1	**0**
Último cociente		**1**

Resultado conversión:

1001101

Conversión de binario a hexadecimal (BIN → HEX)

- Se agrupa el número binario en grupos de 4 empezando por el lado derecho. Si al terminar de agrupar no completa 4 dígitos, entonces se rellenan con 0's a la izquierda
- Posteriormente vea el valor que corresponde de acuerdo a la **tabla de conversión de dígitos hexadecimales a decimales**
- Ejemplo: Convertir a HEX el binario 11011110101
  - Agrupamos de 4 en 4:
    - 0101
    - 1111
    - 0110 → Le agregamos un cero
  - Buscamos la correspondencia en la tabla
    - 0101 → 5
    - 1111 → F
    - 0110 → 6
  - Los juntamos → **6F5**

Binario	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

HEX → BIN y BIN ↔ OCT

La **conversión de BIN a HEX** es más inmediata con la tabla

- Ejemplo: Convertir a BIN el hexadecimal **5B**
  - 5 → 0101
  - B → 1011
  - Los juntamos → 0101 1011 → **101 1011**
- La **conversión entre octal y binario** es análoga, salvo que en este caso agrupamos los bits en **de tres en tres**. Utilizaremos también una tabla de conversión, que en este caso tendrá tres dígitos para los números binarios
- Ejemplos
  - Octal **51** a binario → **101 001**
  - Binario **10110** a octal → **26**

Binario	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Binario	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Conversión de números entre sistemas a mano

Si hacemos las conversiones manualmente:

- No hace falta memorizar las tablas de conversión, basta con saber **contar en binario de 0 a 7 y de 0 a 15**
- Las conversiones **BIN ↔ HEX** y **BIN ↔ OCT** son inmediatas utilizando las tablas de conversión
- Para hacer conversiones **HEX ↔ OCT** es necesario:
  - Pasar primero el número a convertir a BIN
  - Utilizar las tablas para convertirlo al formato final que queremos
- Para la conversión **DEC ↔ HEX y DEC ↔ OCT** se puede aplicar métodos similares a los vistos en binario sólo que:
  - Para conversiones desde DEC Hay que dividir entre 16 para convertir a HEX y 8 para convertir a OCT. Es más rápido pasar antes por la conversión a BIN porque la división por 2 es más rápida
  - Para conversiones a DEC utilizaremos la potencias de 16 para HEX y de 8 para OCT. Otra vez es más rápido pasar antes por la conversión a BIN porque las potencias de 2 son más sencillas

Conversiones HEX → DEC y OCT → DEC sin pasar por binario

- Vemos un ejemplo de conversión DEC → HEX y DEC a OCT → DEC para el **número 5293** sin pasar por binario. Lo difícil de estas es que las divisiones entre 16 y 8 son más laboriosas que las de 2

División	Cociente (Parte entera)	Resto DEC	Resto HEX
5293/16	330	13	**D**
330/16	20	10	**A**
20/16	1	4	**4**
Último cociente			**1**

División	Cociente (Parte entera)	Resto OCT
5293/8	661	**5**
661/8	82	**5**
82/8	10	**2**
10/8	1	**2**
Último cociente		**1**

- $5293_{10} = 14AD_{16}$
- $5293_{10} = 12255_{8}$

Ejercicio. Conversión de números en distintos sistemas

DEC	BIN	OCT	HEX
12			
			1E
		64	
	1000001		
100			
			1FE
		177	
	110000		

Métodos de conversión no manuales

- Calculadora de Windows en modo programador
- Funciones en las hojas de cálculo

F2 =DEC.A.BIN(E2)

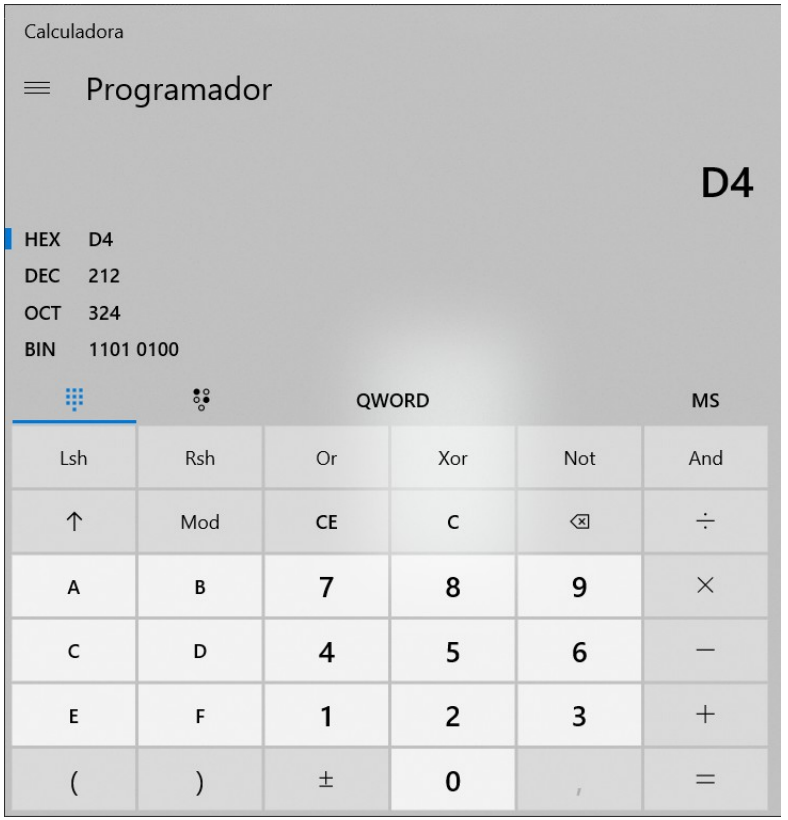
	A	B	C	D	E	F	G	H	I	J	K	L
1	DEC	BIN	OCT	HEX	DEC	BIN	OCT	HEX	DEC	BIN	OCT	HEX
2	0	0	0	0	36	100100	44	24	72	1001000	110	48
3	1	1	1	1	37	100101	45	25	73	1001001	111	49
4	2	10	2	2	38	100110	46	26	74	1001010	112	4A
5	3	11	3	3	39	100111	47	27	75	1001011	113	4B
6	4	100	4	4	40	101000	50	28	76	1001100	114	4C
7	5	101	5	5	41	101001	51	29	77	1001101	115	4D
8	6	110	6	6	42	101010	52	2A	78	1001110	116	4E
9	7	111	7	7	43	101011	53	2B	79	1001111	117	4F

G2 =DEC.A.OCT(E2)

	A	B	C	D	E	F	G	H	I	J	K	L
1	DEC	BIN	OCT	HEX	DEC	BIN	OCT	HEX	DEC	BIN	OCT	HEX
2	0	0	0	0	36	100100	44	24	72	1001000	110	48
3	1	1	1	1	37	100101	45	25	73	1001001	111	49
4	2	10	2	2	38	100110	46	26	74	1001010	112	4A
5	3	11	3	3	39	100111	47	27	75	1001011	113	4B
6	4	100	4	4	40	101000	50	28	76	1001100	114	4C
7	5	101	5	5	41	101001	51	29	77	1001101	115	4D
8	6	110	6	6	42	101010	52	2A	78	1001110	116	4E
9	7	111	7	7	43	101011	53	2B	79	1001111	117	4F

H2 =DEC.A.HEX(E2)

	A	B	C	D	E	F	G	H	I	J	K	L
1	DEC	BIN	OCT	HEX	DEC	BIN	OCT	HEX	DEC	BIN	OCT	HEX
2	0	0	0	0	36	100100	44	24	72	1001000	110	48
3	1	1	1	1	37	100101	45	25	73	1001001	111	49
4	2	10	2	2	38	100110	46	26	74	1001010	112	4A
5	3	11	3	3	39	100111	47	27	75	1001011	113	4B
6	4	100	4	4	40	101000	50	28	76	1001100	114	4C
7	5	101	5	5	41	101001	51	29	77	1001101	115	4D
8	6	110	6	6	42	101010	52	2A	78	1001110	116	4E
9	7	111	7	7	43	101011	53	2B	79	1001111	117	4F



Cantidad de números con n dígitos y una base B

Para un sistema con **base B** podemos determinar la cantidad de números distintos que podemos representar con **n dígitos** con el **cálculo de la potencia con base B y exponente n**

Número de dígitos	Binario (Base 2)		Octal (Base 8)		Decimal (Base 10)		Hexadecimal (Base 16)	
	Potencias de 2	Números	Potencias de 8	Números	Potencias de 10	Números	Potencias de 16	Números
1	2^1	2	8^1	8	10^1	10	16^1	16
2	2^2	4	8^2	64	10^2	100	16^2	256
3	2^3	8	8^3	512	10^3	1.000	16^3	4.096
4	2^4	16	8^4	4.096	10^4	10.000	16^4	65.536
5	2^5	32	8^5	32.768	10^5	100.000	16^5	1.048.576
6	2^6	64	8^6	262.144	10^6	1.000.000	16^6	16.777.216
7	2^7	128	8^7	2.097.152	10^7	10.000.000	16^7	268.435.456
8	2^8	256	8^8	16.777.216	10^8	100.000.000	16^8	4.294.967.296
9	2^9	512	8^9	134.217.728	10^9	1.000.000.000	16^9	68.719.476.736
10	2^10	1.024	8^10	1.073.741.824	10^10	10.000.000.000	16^10	1.099.511.627.776

Cantidad de direcciones de memoria

- Cuando hablemos de los microprocesadores veremos que los hay de 32 y de 64 bits.
- Entre otra cosas estos números indican **el número de dígitos binarios** que utiliza para **codificar las direcciones de la memoria RAM**, ya que cada dirección lleva asociada un valor numérico.
- A cuantos más bits mayor será el número de direcciones que podemos codificar y esto determina la cantidad de memoria teórica con la que puede trabajar un microprocesador
- ¿Cuantas direcciones distintas puede trabajar un procesador de 32 bits y de 64 bits?
  - 32 bits → 32 dígitos binarios → 2^{32} = 4.294.967.296 direcciones
  - 64 bits → 64 dígitos binarios → 2^{64} = 18.446.744.073.709.551.616 direcciones

Ejercicio: Cálculo del número de matrículas posibles

Vamos a probar el cálculo anterior para determinar la cantidad de matrículas de vehículos distintas.

- Las matrículas de los coches en España están formadas por 4 dígitos numéricos y 3 caracteres alfabéticos
- En los caracteres alfabéticos no se utilizan las letras vocales y las consonantes Ñ y Q, por lo que el número de letras que se pueden utilizar es 20
- Aplicando las regla vista antes con el que calculábamos la cantidad de números distintos que podemos representar con n dígitos de una base B ¿Cuantas matrículas de coche distintas se pueden matricular en España con este sistema?

Unidades de medida en Informática

bit y byte

- Dijimos que el ordenador utiliza el código binario para almacenar la información, así que **la mínima cantidad de almacenamiento de información será un 1 o un 0**
- A esta unidad mínima se le llama **bit (b)**, que es el acrónimo de **binary digit**
- El bit es una unidad de medida muy baja por lo que se establecen medidas mayores (Al igual que no medimos las distancias por carretera en milímetros y las medimos en Kilómetros y metros)
- De hecho lo más habitual para **medir el almacenamiento** en Informática es **utilizar la unidad de medida byte (B) que son 8 bits**
- Para abreviarlas utilizamos la **b minúscula para el bit** y la **B mayúscula para el byte**
- El resto de unidades de medida toman el byte como referencia de unidad de medida más pequeña y entre estas distinguimos:
  - Las unidades del **sistema internacional** que utilizan **factores decimales**
  - Las unidades **ISO/IEC 80000-13** que utilizan **factores binarios**

Sistema internacional (Factores decimales)

- Cada unidad superior es 1000 veces una inferior

Nome	Abreviatura	Factor
KiloByte	KB	10^3 Bytes = 1.000 Bytes
MegaByte	MB	10^6 Bytes = 1.000.000 Bytes
GigaByte	GB	10^9 Bytes = 1.000.000.000 Bytes
TeraByte	TB	10^{12} Bytes = 1.000.000.000.000 Bytes
PetaByte	PB	10^{15} Bytes = 1.000.000.000.000.000 Bytes
ExaByte	EB	10^{18} Bytes = 1.000.000.000.000.000.000 Bytes
ZettaByte	ZB	10^{21} Bytes = 1.000.000.000.000.000.000.000 Bytes
YottaByte	YB	10^{24} Bytes = 1.000.000.000.000.000.000.000.000 Bytes

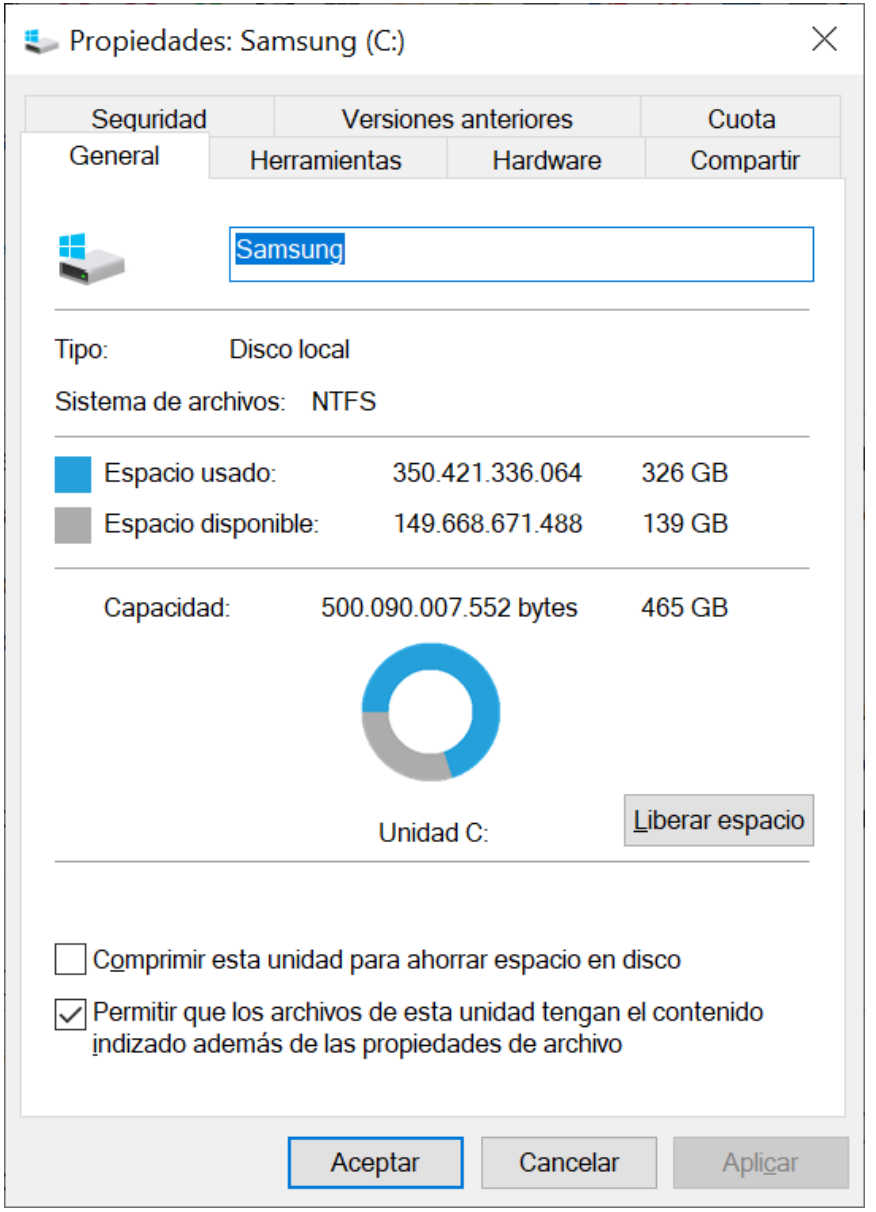
ISO/IEC 80000-13 (Factores binarios)

- Cada unidad superior es 1024 veces una inferior (2^{10})
- Sus abreviaturas son iguales a las de factores decimales con una i minúscula en el medio

Nome	Abreviatura	Factor
KibiByte	KiB	2^{10} Bytes = 1.024 Bytes
MebiByte	MiB	2^{20} Bytes = 1.048.576 Bytes
GibiByte	GiB	2^{30} Bytes = 1.073.741.824 Bytes
TebiByte	TiB	2^{40} Bytes = 1.099.511.627.776 Bytes
PebiByte	PiB	2^{50} Bytes = 1.125.899.906.842.624 Bytes
ExbiByte	EiB	2^{60} Bytes = 1.152.921.504.606.846.976 Bytes
ZebiByte	ZiB	2^{70} Bytes = 1.180.591.620.717.411.303.424 Bytes
YobiByte	YiB	2^{80} Bytes = 1.208.925.819.614.629.174.706.176 Bytes

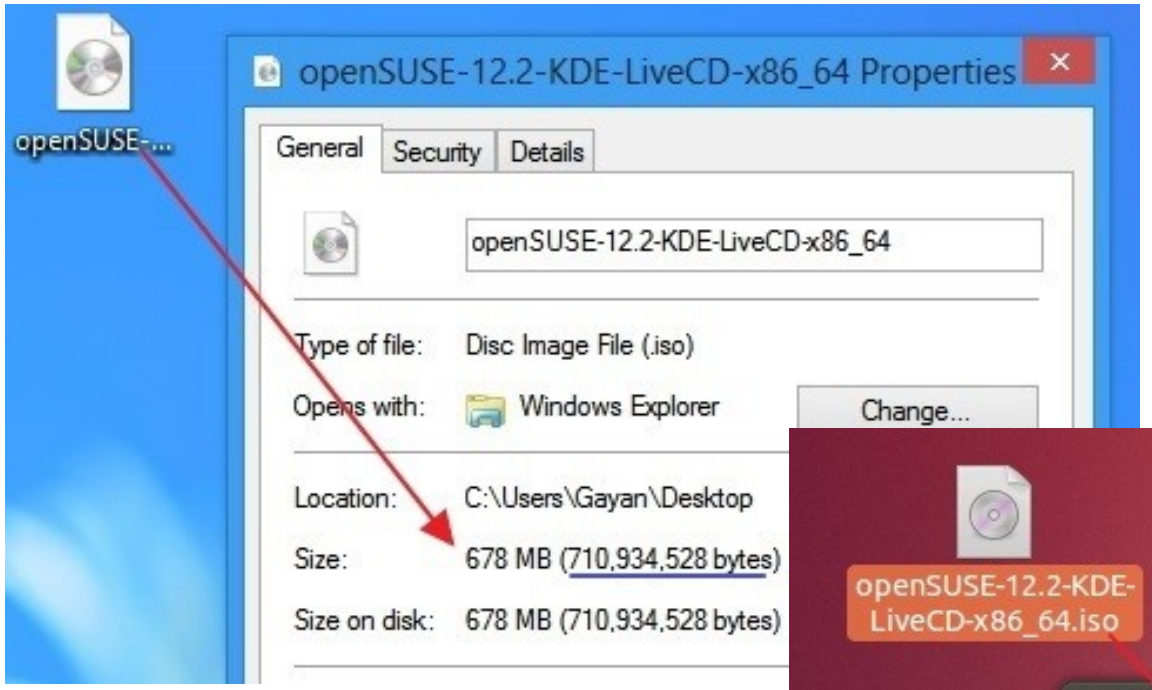
En Windows, ¿factores decimales o binarios?

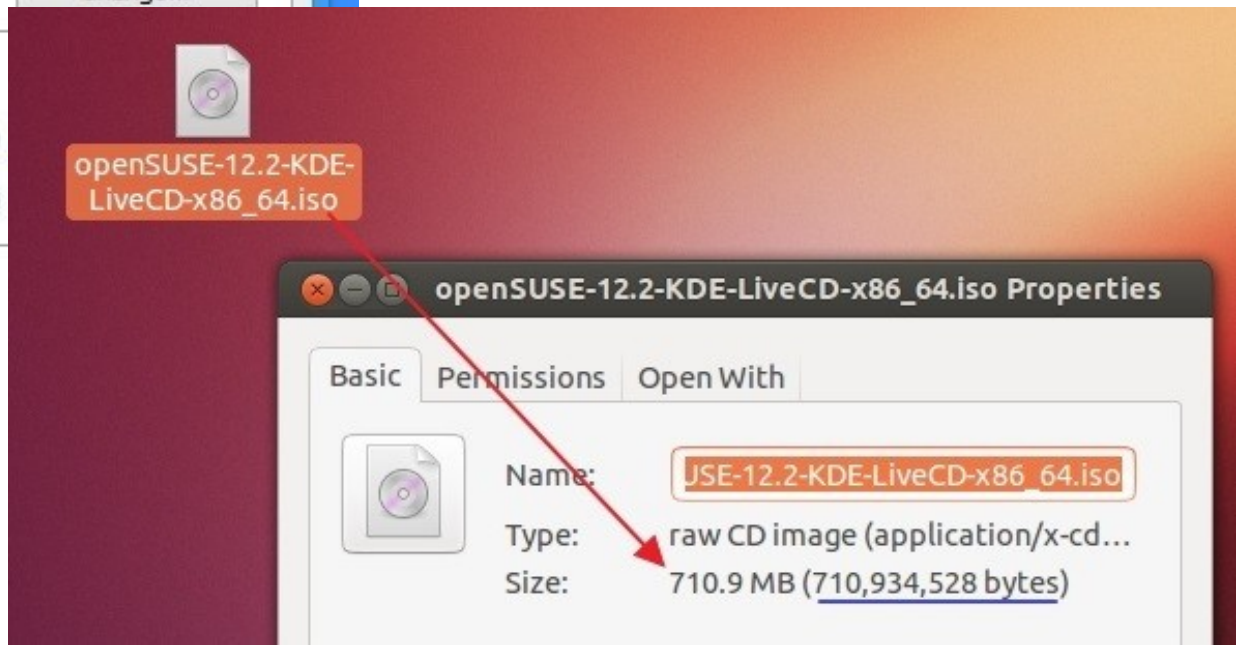
- En Windows si vemos la capacidad de una unidad de disco veremos que muestra la información en bytes y su conversión en GB, pero lo que Windows etiqueta como GB son en realidad GiB
- Por tanto **muestra la capacidad con la abreviatura incorrecta**, por lo que muestra como si las unidades tuvieran menos capacidad de la que tienen en realidad
- En el ejemplo de la captura el disco sería de 500 GB y 465 GiB



En Linux, ¿factores decimales o binarios?

- En cambio en los sistemas Linux, como Ubuntu, sí se muestra la conversión adecuadamente a la unidad de medida que representa:





Conversión entre unidades

- La conversión de unidades de almacenamiento es inmediata siempre que la unidad de origen y destino tengan el mismo factor (como pasar de MB a KB o de KiB a GiB)
- Si son de distinto factor, una decimal y otra binaria (como de MB a MiB o KiB a GB), lo mejor es hacer un paso intermedio de la unidad a byte

Ejercicio: Conversión de unidades

Dato origen	Unidad destino	Resultado conversión
2 MB	KB	
10.000 KiB	MiB	
50 MB	KiB	
200 GB	GiB	
1.600 Kb	MB	
2 GB	Mb	
1.024 Gb	KiB	

Unidades de frecuencia

La unidad de frecuencia son los **Hercios (Hz)** y es una unidad de frecuencia que indica el número de operaciones que se realizan por segundo que determina en gran medida lo rápido que funcionan distintos elementos del ordenador:

- En **un microprocesador** indica el **número de instrucciones** que ejecuta por segundo
- En un **bus de transferencia de datos**, como el de los dispositivos de almacenamiento, indica el **número de veces por segundo que se hace una transferencia**
- En un **monitor** indica **la cantidad de veces por segundo que se refresca la imagen en pantalla**

Por lo general cuantos más Hz tenga un componente más rápido será este y por tanto el funcionamiento del ordenador

Nome	Abreviatura	Factor
kilohercio	kHz	10^3 Hz = 1.000 hercios
Megahercio	MHz	10^6 Hz = 1.000.000 hercios
Gigahercio	GHz	10^9 Hz = 1.000.000.000 hercios
Terahercio	THz	10^{12} Hz = 1.000.000.000.000 hercios
Petahercio	PHz	10^{15} Hz = 1.000.000.000.000.000 hercios
Exahercio	EHz	10^{18} Hz = 1.000.000.000.000.000.000 hercios

Medidas de tasa o velocidad de transferencia

- La **tasa de transferencia** mide la cantidad de datos que se transmiten por un canal de comunicaciones por unidad de tiempo
- También se llama **bitrate** y se suele medir en **bits por segundo (bps) o bytes por segundo (Bps)** y sus múltiplos
- 1 Bps serán 8 bps, al igual que 1 byte son 8 bits

Nome	Abreviatura	Factor
kilobit/segundo	kbps	10^3 bits cada segundo
Megabit/segundo	Mbps	10^6 bits cada segundo
Gigabit/segundo	Gbps	10^9 bits cada segundo
Byte/segundo	B/s	8 bits cada segundo
kiloByte/segundo	kB/s	10^3 bytes cada segundo ou 8.000 bits cada segundo
MegaByte/segundo	MB/s	10^6 bytes cada segundo
GigaByte/segundo	GB/s	10^9 bytes cada segundo

Ejercicio: Conversión de unidades

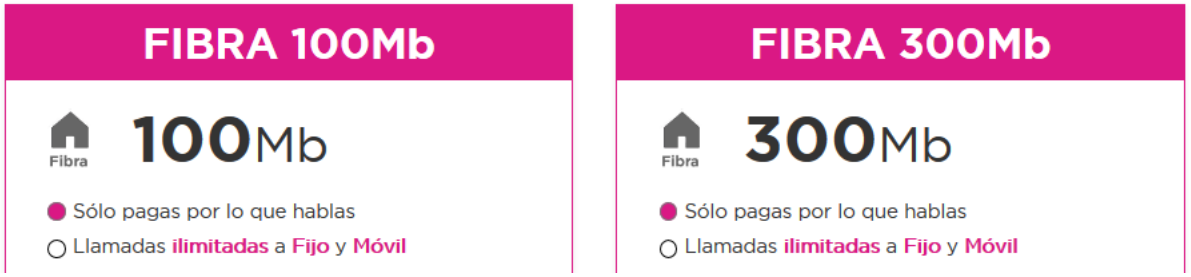
Dato origen	Unidad destino	Resultado conversión
15.000 Hz	Khz	
3200 Kbps	MB/s	
4 GB/s	Mbps	
1.024 MB/s	GB/s	

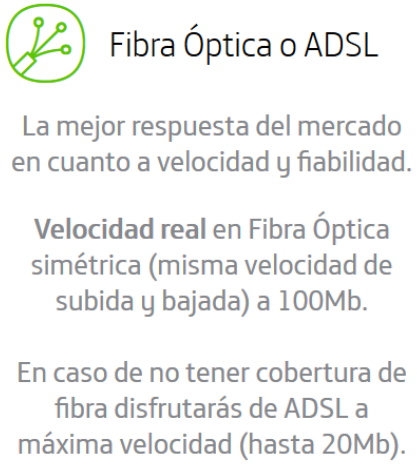
Tasa de transferencia en un bus de comunicaciones

- En un bus de comunicaciones de un ordenador su tasa de transferencia viene expresada en Mhz, que viene a indicar el número de bits por segundo que es capaz de transmitir
- Pero para calcula la tasa de transferencia del bus de comunicaciones de un ordenador también tenemos que tener en cuenta un elemento adicional que es el **ancho del bus que se mide en n.º de bits**
- Un bus en un ordenador no es más que el canal por el que se envían los datos y cuanto más ancho sea mayor será la cantidad de información que transmitirá por segundo (Es como añadir más carriles a una autopista)
- Ejemplos
  - Un canal con un ancho de 1 bit a 8Mhz → 8 Mbps → 1 MBps
  - Un canal con un ancho de 8 bits a 8Mhz → 64 Mbps → 8 MBps

¿Y tu proveedor de Internet y telefonía que te vende?

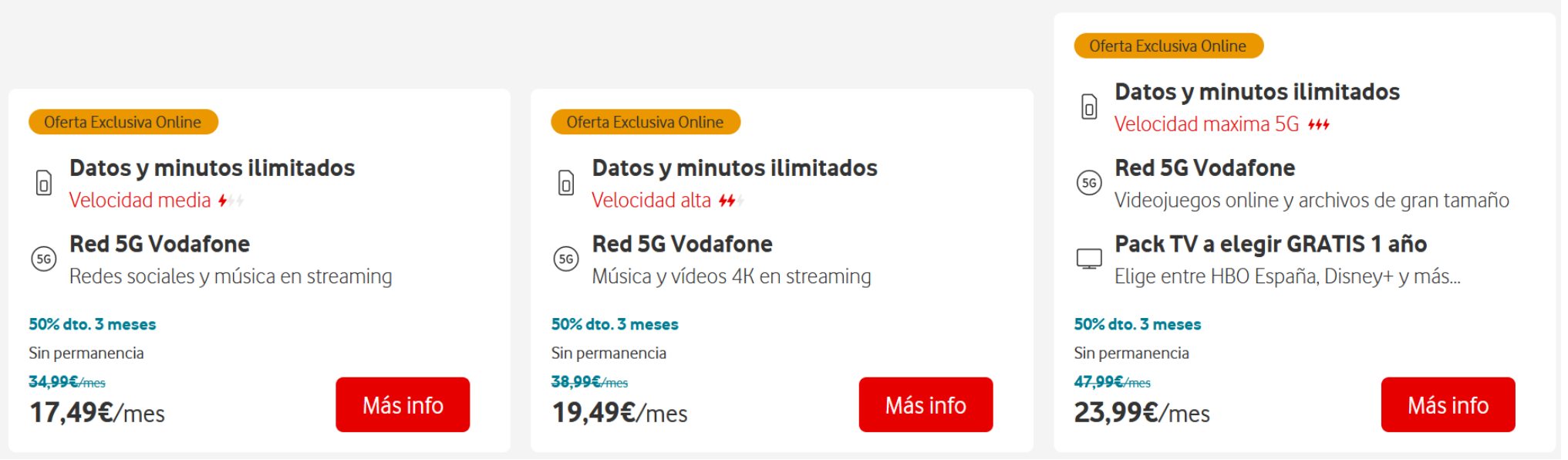
- El ancho de banda de fibra y ADSL al que se comprometen los proveedores de Internet lo miden en Megabits por segundo (Mbps). Fijaros en que los “Megas” que ofertan la b es minúscula de bit
- Sin embargo los “Gigas” de datos de la línea móvil son en Bytes (GB)





Líneas móviles sin límite de datos

- Los operadores que no limitan la cantidad de datos ahora cobran más o menos dependiendo del ancho de banda
- La máxima velocidad de la tarifa más cara dependerá de la red que tengamos:
  - 4G → 1Gbps (teórica)
  - 5G → 20 Gbps (teórica)



Ejercicio: Velocidad de transferencia de un bus

Características del BUS	Unidad destino	Velocidad de transferencia
1 bit y 100 Khz	Kbps	
32 bits y 1000 Khz	MB/s	
1 bit y 2,4 Ghz	MB/s	
32 bits y 66 Mhz	GB/s	

La evolución del ancho de banda I

En España en los 90 antes de llegar la banda ancha las primeras conexiones a Internet se hacían con módems que se conectaban a través de la línea telefónica. Estos daban velocidades que variaban dependiendo del módem del que dispusiéramos con velocidades máximas que fueron evolucionando de la siguiente forma:

- 14400 bps = 14,4 Kbps
- 28800 bps = 28,8 Kbps
- 36600 bps = 36,6 Kbps
- 56 Kbps

La evolución del ancho de banda II

- Algunas peculiaridades de los módems:
  - Los ruidos que hacían cuando establecían la conexión → https://www.youtube.com/watch?v=aV8DEJ8ydJQ
  - Ocupaban la línea telefónica para llamadas (el teléfono móvil no estaba muy extendido y generaba problemas domésticos)
  - La variabilidad de la velocidad sobre todo en las horas en las que más gente se conectaba
  - Lo sensibles que eran a los picos de tensión

- La **banda ancha no llegó hasta el año 2000** con la aparición de los operadores de **cable** que tenían concesión de explotación a nivel autonómico (R, Euskaltel, Telecable, ...) y la **ADSL** (Telefónica), que de aquella **ofrecían velocidades de 128 Kbps**, que además de ser más del doble de lo que se tenía con un módem daban velocidades más estables

Ejercicio: Descargar un DVD “90’s vs Actualidad”

Vamos a comparar el tiempo que nos llevaría hoy en día descargar un fichero de 4,7GB (Una imagen de un DVD repleta) de Internet con un ancho de banda actual de **200 Mbps** frente a un ancho de banda de los 90 con un módem de **56 Kbps**, suponiendo un caso ideal en el que la velocidad máxima se mantuviera durante todo el tiempo de descarga

Velocidad	Tamaño descarga	Tiempo de descarga
200 Mbps	4,7GB	
56 Kbps	4,7GB	

Streaming

- La transmisión en directo, o streaming como hemos adoptado del inglés, es la distribución digital de un contenido multimedia a través de una red de ordenadores que permite al usuario consumir un contenido a medida que este este se descarga
- Es una alternativa que **se contrapone al modelo de descarga de archivos** en el que tenemos que descargar por completo el contenido antes de poder consumirlo
- Actualmente consumimos servicios basados en streaming a diario tanto de vídeo como de música:
  - Vídeo → YouTube, Netflix, Amazon prime Video
  - Música y Podcasts → Spotify, Apple music, Amazon Music

Streaming sin paradas I

En streaming la información del contenido se va a almacenando en un búfer de datos del que vamos leyendo a medida que lo consumimos y en algunos reproductores como el de YouTube esto nos lo muestra en la barra de reproducción con tres colores:

- El rojo indica el punto en el que estamos de la reproducción
- El blanco indica cuanto se ha descargado del vídeo y está en el búfer
- El gris indica la parte pendiente de descargar del vídeo

Streaming sin paradas II

Las paradas en la reproducción se producen cuando lo descargado no es capaz de seguir el ritmo de la reproducción (Cuando la línea roja alcanza la blanca), y esto ocurre **cuando el ancho de banda del que disponemos no es lo suficientemente alto**

Para poder **consumir un contenido digital en streaming sin saltos** debemos disponer de una conexión a Internet con un ancho de banda igual o mayor que la tasa de transferencia mínima de del contenido a consumir. Esta tasa de transferencia la podemos calcular de la siguiente forma:

$$\frac{Tamaño\ total\ del\ contenido}{Duración\ del\ contenido\ en\ segundos}$$

Hecho este cálculo sólo nos quedaría convertir lo obtenido a la unidad que queramos bps, Kbps, Mbps, Gbps, Bps, KBps, ...

Ejercicio: Ancho de banda mínimo para streaming

- Queremos escuchar una canción que tiene:
  - Una duración de 4 minutos y 23 segundos
  - Un tamaño de 5,2 MB
- Supongamos que queremos ver una película que tiene:
  - Una duración de 1 hora 23 minutos y 40 segundos y que tiene
  - Un tamaño de 2,4GB
- Calcula el ancho de banda mínimo en Mbps que necesitamos para poder escuchar la canción y ver la película sin paradas:

Contenido	Ancho de banda mínimo
00:04:23 y 5,2 MB	
01:23:40 y 2,4GB	

Codificación del texto

Codificación de datos no numéricos

- Tal como hablamos antes los ordenadores codifican toda la información en código binario y con las conversiones entre distintos sistemas de numeración nos es fácil imaginar cómo almacenan los números y cuanto espacio se necesita
- Lo que veremos ahora es como en un ordenador se codifican otros tipos de información en binario y cuanto espacio de almacenamiento necesitan.
- Lo veremos para los siguientes:
  - Texto
  - Imágenes
  - Sonido
  - Vídeo

Texto

- La codificación del texto se hace asignando un valor numérico a cada carácter imprimible. Dependiendo de cómo se haga esta codificación podemos hablar de los siguientes códigos:
- **ASCII** → Es un código que utiliza códigos numéricos de **7 bits** por lo que es capaz de representar **128 caracteres (2^7)**. La primera publicación del código ASCII es del **año 1963**
- **ASCII extendido** → El ASCII extendido añade un bit adicional para codificar los caracteres, así con **8 bits** dispone de **256 caracteres (2^8)**
- **Unicode** → Este código utiliza hasta **4 bytes** para codificar los caracteres, **32 bits → 2^{32} → 4.294.967.296 caracteres**. Su primera publicación es del **año 1991**.

ASCII – 7 bits – 128 caracteres

- El código ASCII reserva los primeros 32 caracteres (del 0 al 31) para caracteres de control, no pensados para representar información imprimible (Escape, salto de línea, tabuladores, ...)

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

asciicharstable.com

ASCII extendido – 8 bits – 256 caracteres

- El problema del ASCII es que los 128 caracteres eran suficientes para la lengua inglesa, pero no para representar los caracteres de otros idiomas, como por ejemplo la ñ del idioma español, los acentos, las interrogaciones y exclamaciones de apertura, ...
- Se solucionó parcialmente en el ASCII extendido en el que se añadió un bit adicional para codificar los caracteres, así con **8 bits** dispone de **256 caracteres (2^8)**

128	Ç	144	É	160	á
129	ü	145	æ	161	í
130	é	146	Æ	162	ó
131	â	147	ô	163	ú
132	ä	148	ö	164	ñ
133	à	149	ò	165	Ñ
134	å	150	û	166	ª
135	ç	151	ù	167	º
136	ê	152	ÿ	168	¿
137	ë	153	Ö	169	⌐
138	è	154	Ü	170	¬
139	ï	155	¢	171	½
140	î	156	£	172	¼
141	ì	157	¥	173	¡
142	Ä	158	₧	174	«
143	Å	159	ƒ	175	»

ASCII extendido – 8 bits – 256 caracteres

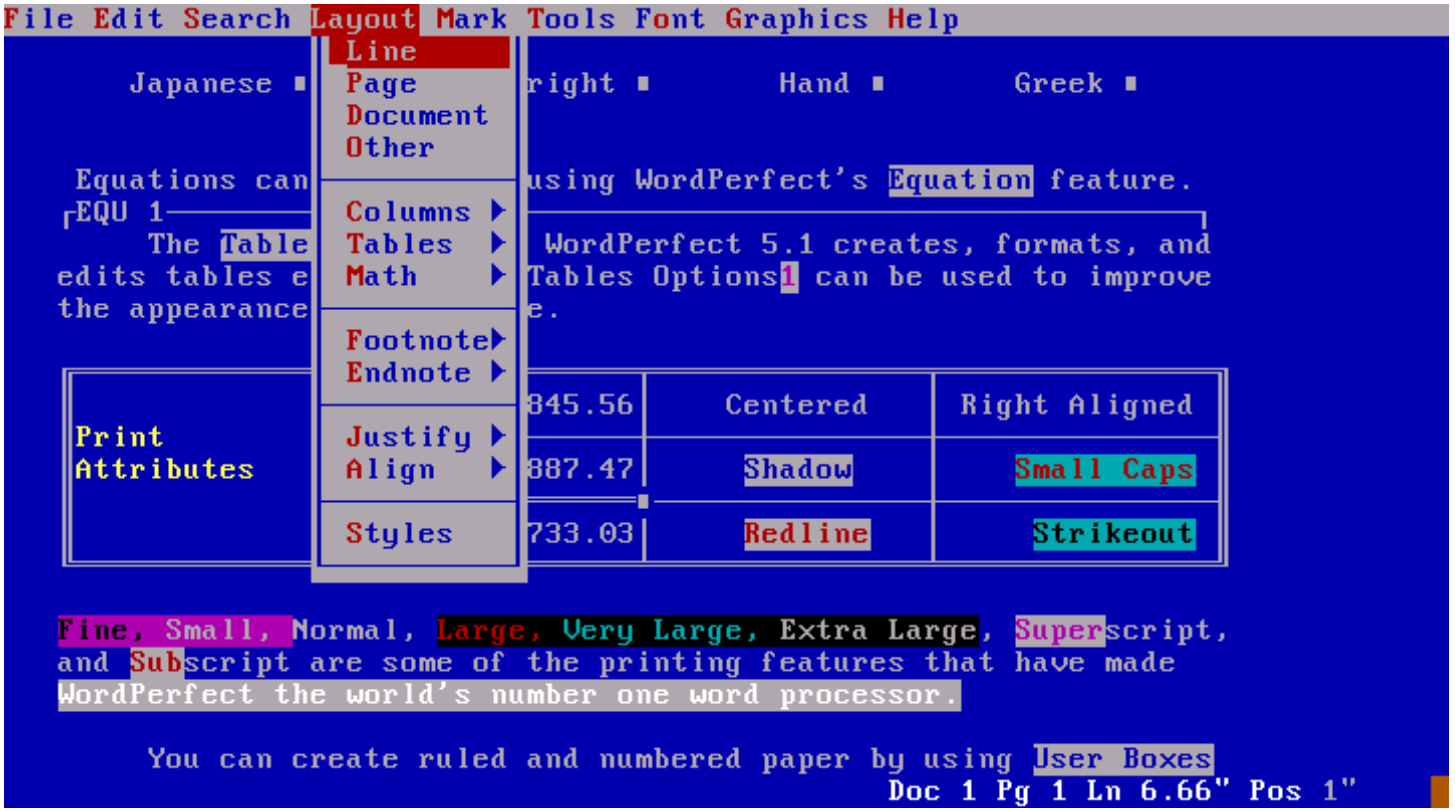
Vemos una tabla con todo el juego de caracteres que incorporó el ASCII extendido al ASCII original (del 128 al 255)

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	n
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

ASCII extendido – 8 bits – 256 caracteres

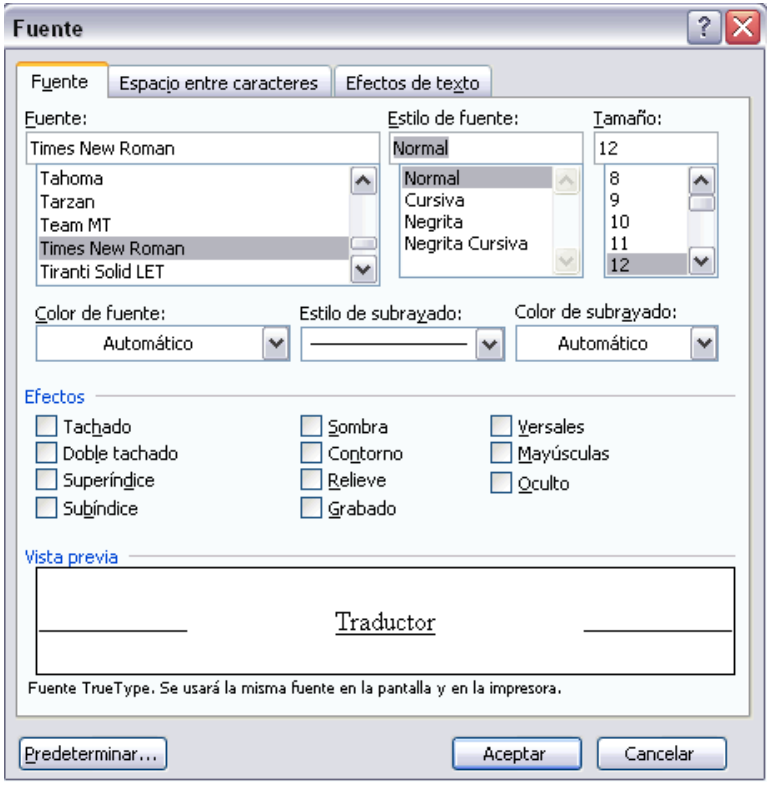
- Algunos sistemas operativos sin interfaz gráfica, como el MS-DOS, utilizaba los caracteres del ASCII extendido para poder hacer elementos como los menús, tablas e incluso ventanas
- En estos el cursor del ratón se representaba con un rectángulo

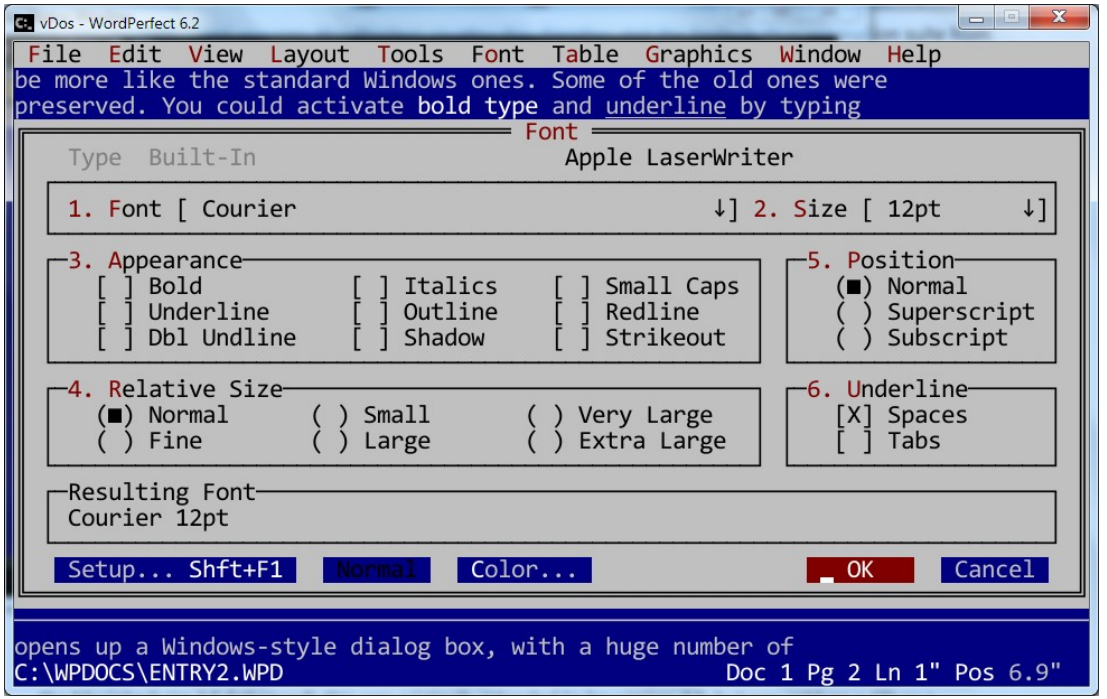
176	▒	192	└	208	╨
177	▒	193	┴	209	╤
178	▓	194	┬	210	╥
179	│	195	├	211	╙
180	┤	196	─	212	╘
181	╡	197	┼	213	╒
182	╢	198	╞	214	╓
183	╖	199	╟	215	╫
184	╕	200	╚	216	╪
185	╣	201	╔	217	┘
186	║	202	╩	218	┌
187	╗	203	╦	219	█
188	╝	204	╠	220	▄
189	╜	205	═	221	▌
190	╛	206	╬	222	▐
191	┐	207	╧	223	▀



ASCII extendido – 8 bits – 256 caracteres

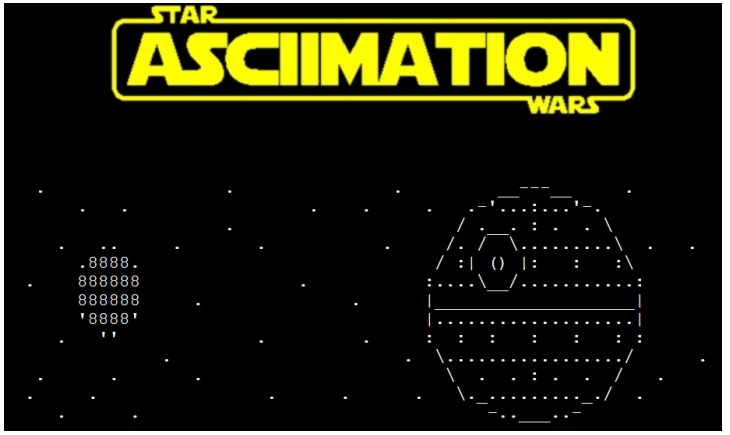
- Vemos también como se puede hacer un díalogo típico de formato de carácter con sus checkboxes, radiobuttons y botones con caracteres ASCII





Usos más artísticos del ASCII

- Hoy en día podemos ver como se utiliza el ASCII para hacer cosas curiosas:
  - Arte ASCII
  - Vídeos en ASCII → Star wars en ASCII http://www.asciimation.co.nz
  - Cámara ASCII (aplicaciones smartphone)



Unicode – 32 bits (4 bytes) – 4.294.967.296 caracteres

- El Unicode al utilizar 4 bytes puede codificar más de 4000 millones de caracteres, lo que le permite codificar casi cualquier carácter de cualquier idioma, incluidas lenguas muertas
- Poco a poco el Unicode se va extendiendo y consolidando pero el ASCII tiene un dominio histórico que es difícil de vencer

Tarea: Código ASCII. Descifrar mensaje en hexadecimal

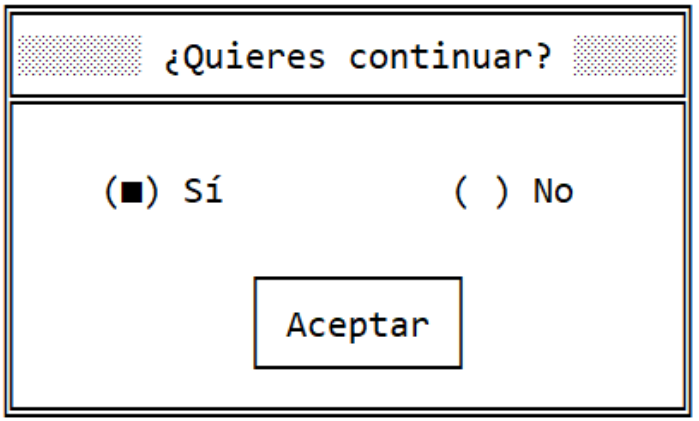
Utilizando la tabla de ASCII obtén el texto del siguiente mensaje codificado en hexadecimal:

5069656E73612C206F7267616E697A6120792065A65637574612E

Ten en cuenta que cada carácter en el ASCII extendido se codifica con 8 bits, por lo que ¿Cuantos caracteres hexadecimales se utilizan para codificar cada carácter?

Tarea: Código ASCII. Cuadro de diálogo con caracteres ASCIIm

- Utilizando la tabla de caracteres ASCII extendido tienes que replicar el cuadro de diálogo de la siguiente captura:

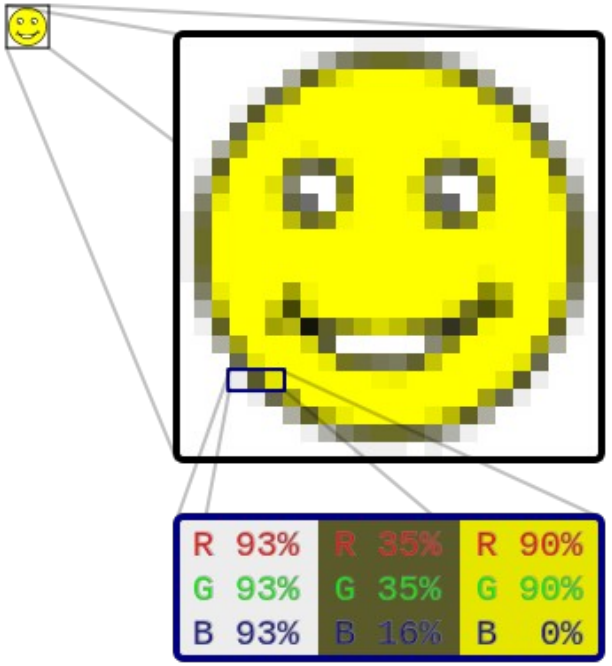


- Lo harás utilizando el bloc de notas, pero como en el teclado no disponen de los caracteres que necesitas utilizarás el siguiente método para introducir un carácter de la tabla ASCII a partir de su código:
  - Activa el teclado numérico con la tecla Bloel teq Num si no está ya activado
  - Pulsando la tecla Alt y si soltarla introduce en el teclado numérico el número de código ASCII que quieres escribir
  - Suelta la tecla Alt, tras lo cual se escribirá el carácter que has codificado
- Cuando acabes guarda el fichero como DialogoAsciiSiNo.txt

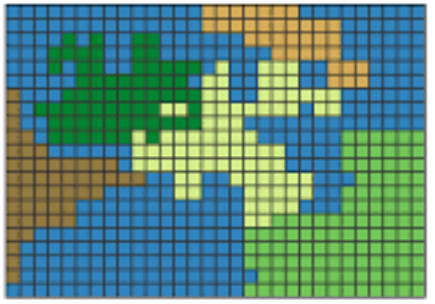
Codificación de imágenes

Imágenes ráster o mapas de bits

- La forma más sencilla de codificar en binario imágenes son las llamadas **imágenes ráster o mapas de bit**
- La imagen se representa como una **matriz de dos dimensiones de puntos** en el cada punto es lo que llamamos un píxel
- A cada pixel se le puede asignar un color, con lo que formamos la imagen de una forma similar a un mosaico



Polygon features



Raster polygon features

El principio del mapa de bits no es nuevo

Mosaico

Punto de cruz

Hojas cuadriculadas

Características de los mapas de bits

- **Resolución**
  - Es el número de píxeles de la imagen
  - Se expresa **ancho x alto** (640x480, 300x240, ...)
- **Profundidad de color**
  - Determina la cantidad de colores posibles que puede tener un pixel
  - Se expresa en bits por pixel, o simplemente bits (8bits, 16 bits, 24 bits, ...)
  - El número de colores que puede representar es $2^{\text{Nº de bits}}$

N.º de bits	Cantidad de colores
8	$2^8 = 256$
16	$2^{16} = 65.536$
24	$2^{24} = 16.777.216$
32	$2^{32} = 4.294.967.296$

Alta y baja resolución en gráficos raster

A mayor resolución de una imagen, más grande será esta y mayor nivel de detalle

Al hacer zoom sobre una imagen de poca resolución percibiremos pérdida de nitidez y bordes aserrados al percibir mejor cada píxel

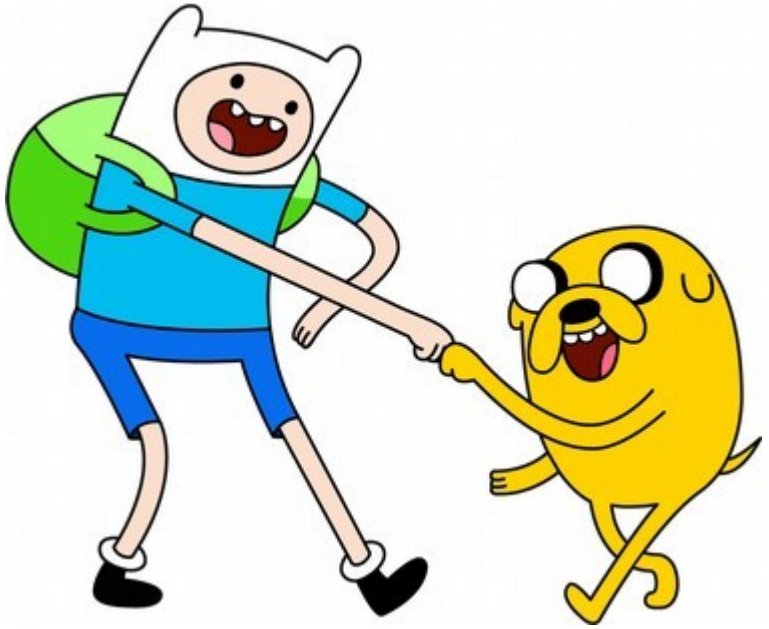


Imagen a 378x311

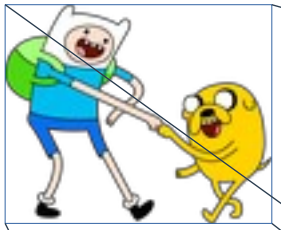


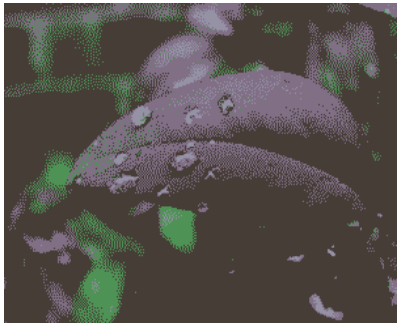
Imagen a 100x82

Imagen a 100x82 con zoom

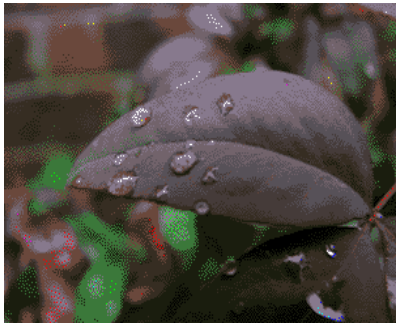
Los bits y las imágenes

- Cuanto mayor es la profundidad mayor es el **número de bits** que determina la cantidad de colores que puede representar, ya que a cada color se le asigna un número
- Así que cuantos más bits mas colores, por lo que más se parecerá a la imagen tal como la percibiriamos los objetos en la realidad.

1 bit (2 colores)
5Kb



2 bits (4 colores)
6Kb



4 bits (16 colores)
13Kb

8 bits (256 colores)
38Kb

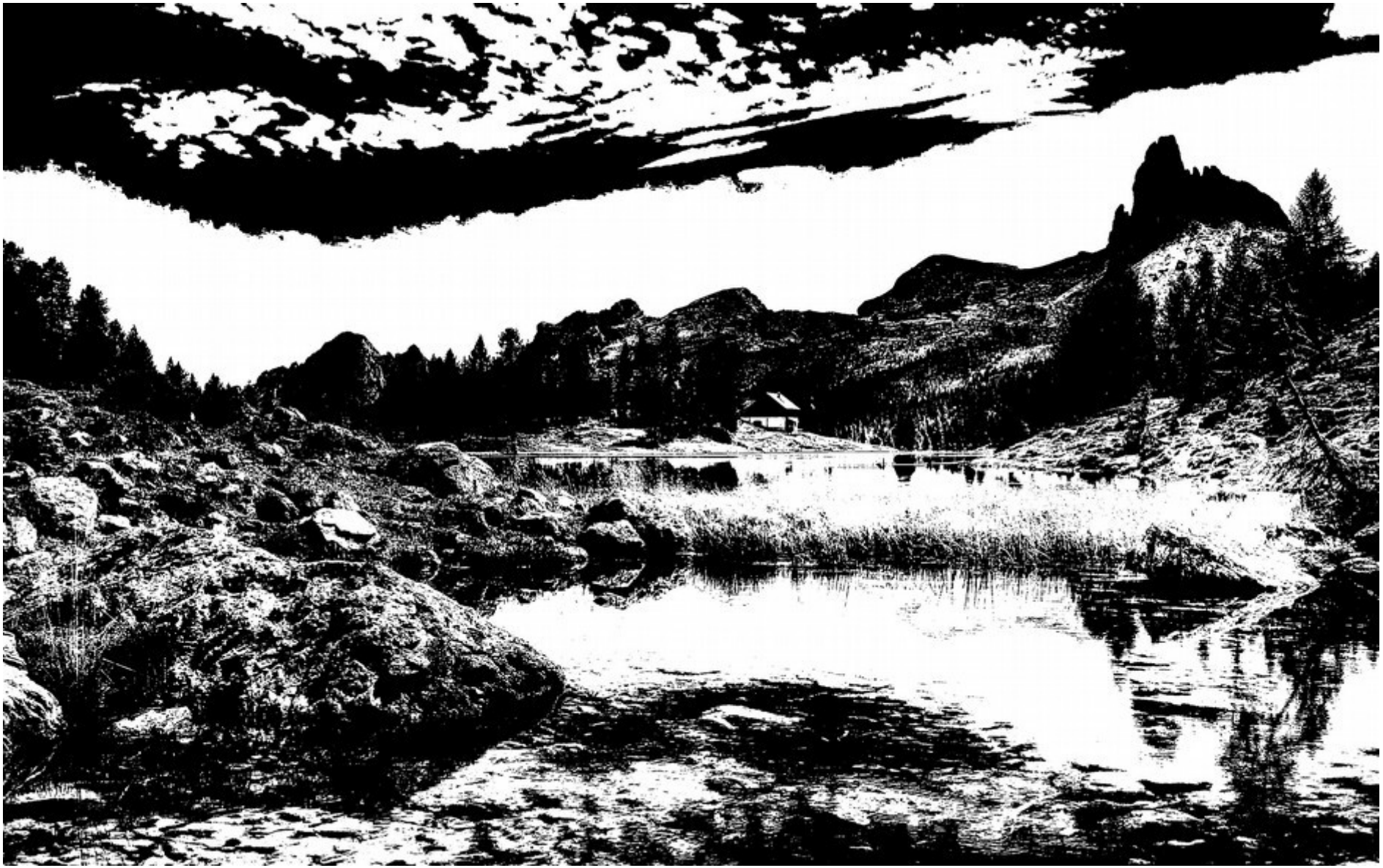
24 bits (16.777.216 colores) 98Kb

Paisaje a 24 bits (16.777.216 colores)

Paisaje a 8 bits (256 colores)

Paisaje a 4 bits (16 colores)

Paisaje a 1 bit (2 colores)

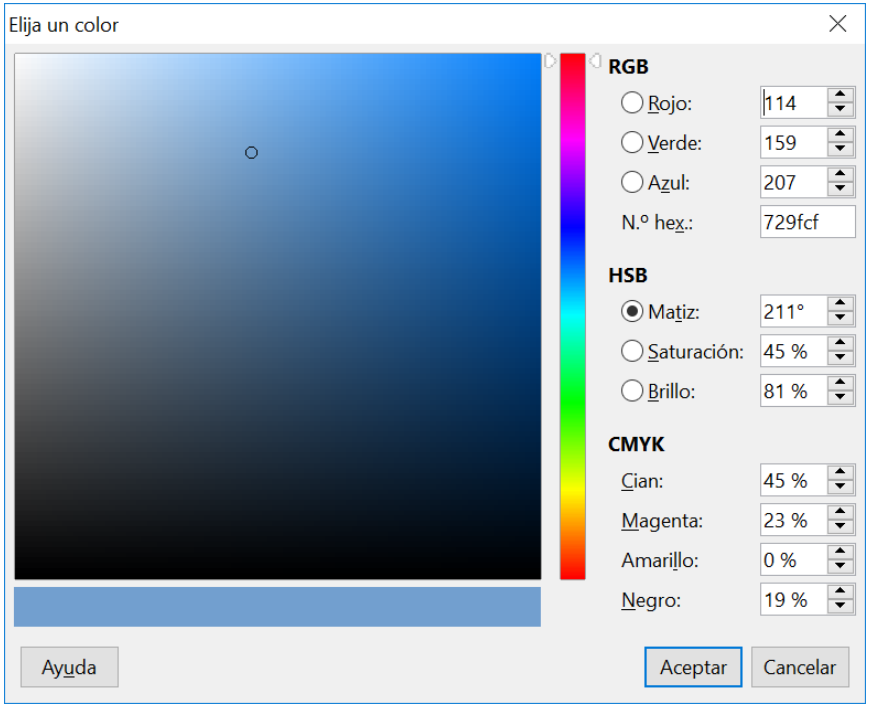


Paisaje a 8 bits (256 colores) en escala de grises

Paisaje a 1 bit (2 colores) con tramado

Codificación del color

- En muchos programas se utiliza el modelo RGB para expresar el color. En este. En este modelo se utilizan tres colores que al combinarlos con distintas intensidades nos da una gran variedad cromática
- Si tenemos 24 bits de profundidad la intensidad de cada color se expresa con 8 bits (un valor entre 0 y 255)
- Para simplificar la notación se utiliza su equivalente en hexadecimal



¿Cuánto ocupa una imagen?

- El tamaño del fichero de una imagen **crecerá proporcionalmente** a
  - La **resolución** = Número de píxeles que almacena
  - La **profundidad de color** = Número de bits por pixel
- El **tamaño** de una imagen será el **número de bits totales** necesarios para codificar el color de todos sus píxeles
- Así podemos calcular el espacio que necesitamos para una imagen a partir de su resolución y profundidad de color
- Ejemplo:

Imagen de 640x480 a 24 bit

640 * 480 = 307.200 píxeles

307.200 pixeles * 24 bits por pixel = 7.372.800 bits en total

7.372.800 bits = 921.600 bytes = 921,6 KB = **0,9216 MB**

Algoritmos de compresión

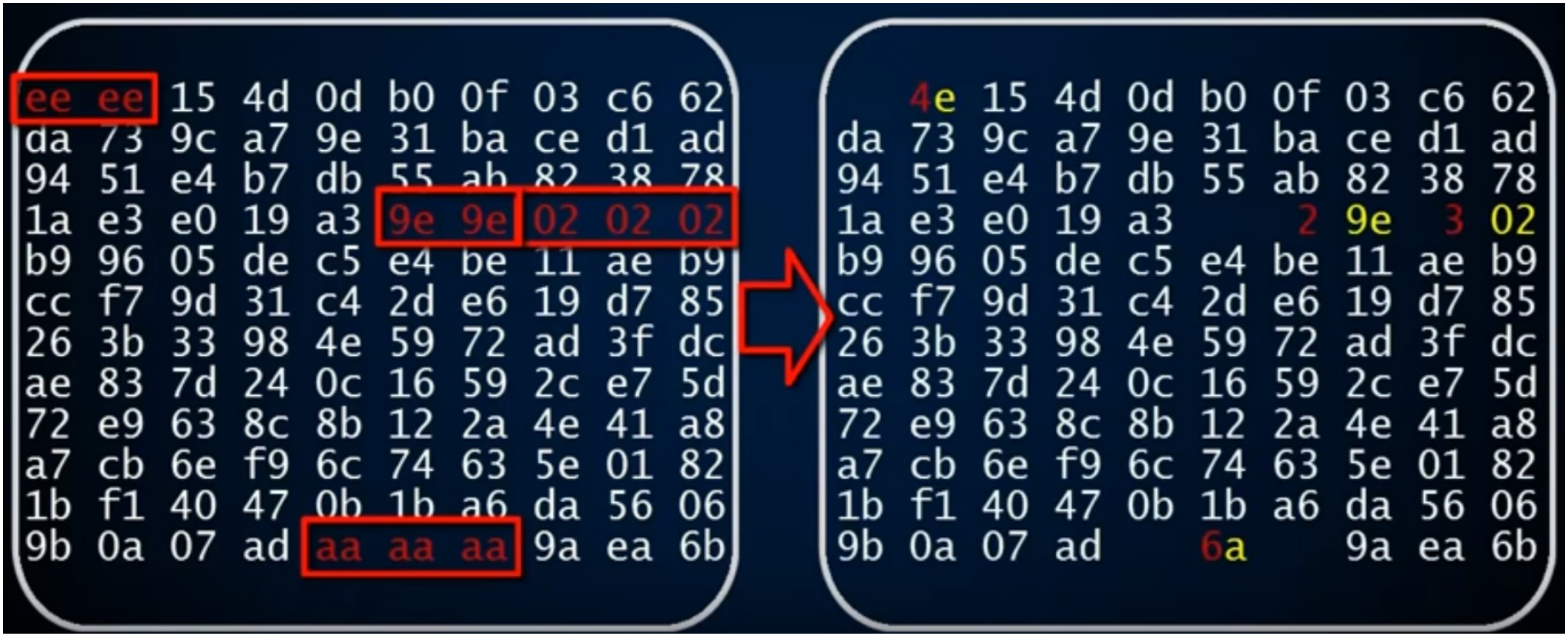
- Si una imagen de 640x480 ocupa casi 1MB con imágenes grandes nuestros ficheros tendrán tamaños difíciles de manejar
- Es el caso del formato de imágenes con formato **BMP (Bitmap)**, que no comprime nada, por lo que no es un formato adecuado para ficheros grandes
- Para reducir este tamaño a las imágenes se aplican **algoritmos de compresión** de los que hay dos tipos:
  - **Sin pérdida de información**. Eliminan información redundante que se puede replicar. Ejemplos de formatos → **GIF y PNG**
  - **Con pérdida de información**. Eliminan información irrelevante o que puede pasar desapercibida al ojo no entrenado. Ejemplo de formato → **JPG o JPEG**
- Nosotros para el cálculo del tamaño de una imagen no tendremos en cuenta la compresión

JPG con compresión decreciente de izquierda a derecha

- Si utilizamos un algoritmo de compresión con pérdida podremos ajustar el nivel de compresión y cuando más comprimamos más detalle perderemos
- Lo vemos en la siguiente imagen en la que la compresión es variable, de más a menos de izquierda a derecha

¿Cómo funciona la compresión?

- El funcionamiento más básico de la compresión de archivos es el de buscar patrones de datos que se repiten con cierta periodicidad, por lo que en vez de guardar todas las repeticiones almacena una sola repetición acompañada del número de veces que se repita
- Por tanto al comprimir un archivo el tamaño comprimido variará con respecto al original dependiendo de sus datos, de forma que se podrá comprimir más cuantos más patrones de repetición se localicen



https://www.youtube.com/watch?v=-LTiVOY8aGY

Ejercicio: Cálculo del tamaño de una imagen y ratios de compresión

- Calcula el tamaño de una imagen tal que:
  - Tenga una resolución de 1024x768
  - Una profundidad de color de 24 bits
- Utiliza el Gimp para crear una imagen con esa resolución que sea toda de color blanco
- Expórtala a BMP con el nombre **ImagenBlanca.bmp** y compárala con el cálculo que has hecho
- Expórtala a JPG con el nombre **ImagenBlanca.jpg** y compara los tamaños de este formato con compresión con el BMP sin compresión

Ejercicio: Cálculo del tamaño de una imagen y ratios de compresión

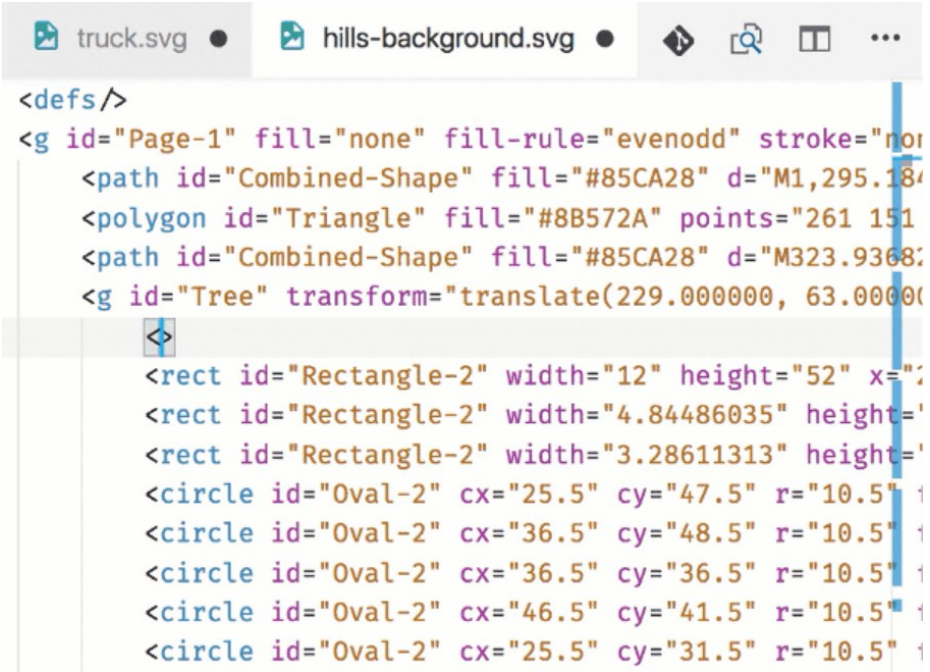
- La imagen que creaste antes tiene muchos patrones repetidos y en proporción se comprime mucho con respecto a la original. Para contrastar la variación de compresión con una imagen con más variedad:
  - Busca ahora en Internet una imagen de un paisaje cualquiera
  - Ábrela con el Gimp y expórtala a BMP y JPG con el nombre Paisaje.bmp y Paisaje.jpg
  - Compara la diferencia de tamaños de esta imagen en BMP con la que descargaste
- Deberás observar que la foto del paisaje proporcionalmente se comprime mucho menos que la que creaste, por lo que para saber cuanto calcula el % de las imágenes comprimidas en formato JPG con respecto a las imágenes en formato BMP
- Comprime ahora las imágenes .BMP con el compresor 7zip en formato 7z. Este es un tipo de compresión sin pérdidas en la que verás como proporcionalmente es mucho mayor que una compresión con pérdidas como es el JPG
- Entregarás con la tarea en un fichero comprimido:
  - Todas las imágenes .BMP y .JPG que has generado
  - Una captura del explorador de archivos donde se vean todos los tamaños
  - Un documento donde indiques los %'s del tamaño de los ficheros comprimidos con respecto al .BMP sin comprimir

Imágenes vectoriales

- Otra forma de almacenar imágenes en informática es utilizado imágenes vectoriales
- En estos la imagen se representa como un conjunto de objetos geométricos cada uno definido por atributos como: tipo de forma, posición, tamaño, color, ...
- Ejemplo: En el caso de un círculo:
  - En un gráfico vectorial este quedaría determinado por:
    - La **posición** en coordenadas de su centro
    - La longitud de su **radio**
    - El grosor y color de su **borde**
    - El color de **relleno**
  - En un mapa de bits este quedaría determinado por todos los píxeles que caen en el área de la matriz que ocupa el círculo: borde, relleno y el color de cada uno de estos píxeles

Almacenamiento de imágenes vectoriales

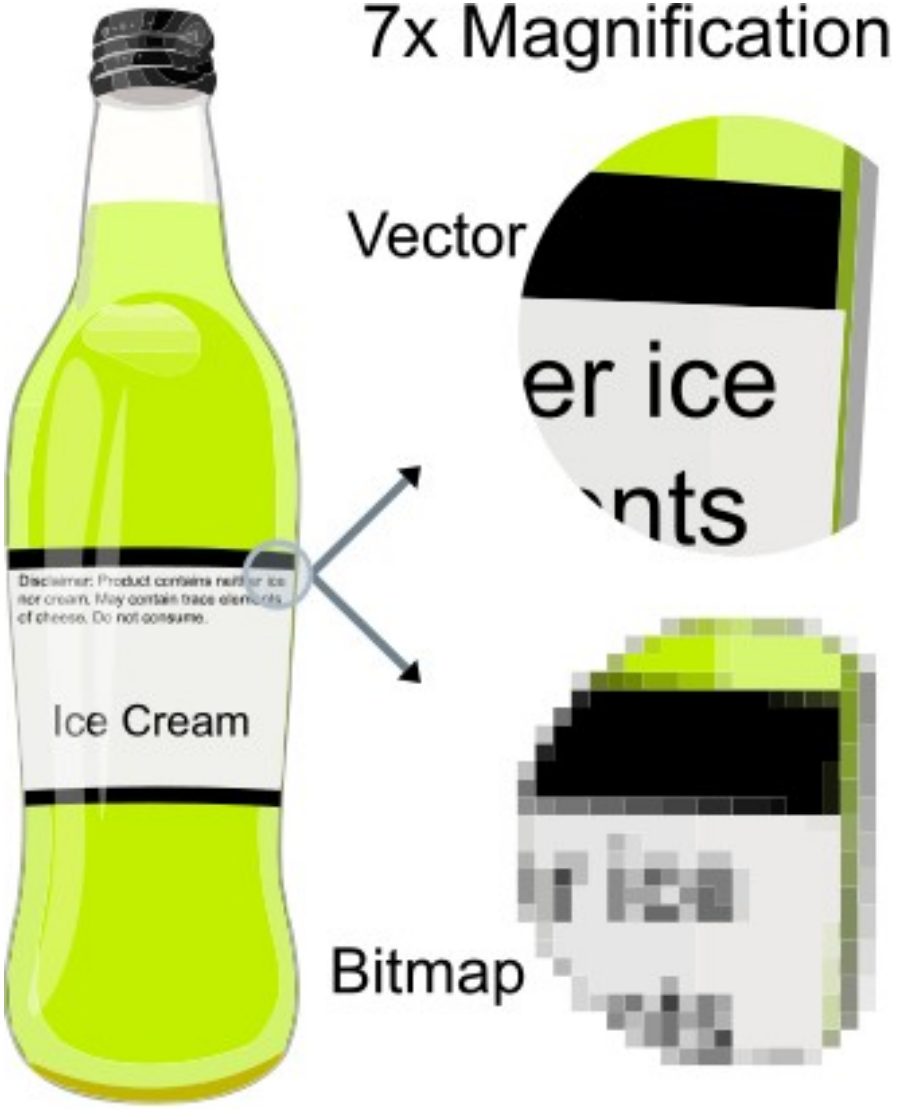
- Hay múltiples formatos de imágenes vectoriales, pero todos se basan en el principio de tener que almacenar formas geométricas de una u otra forma
- El **formato SVG** es uno de los más populares para imágenes vectoriales
- En este **la imagen se modela en forma de texto en el que se indican todos los atributos de las formas geométricas de la imagen**. Este texto se almacena de forma estructurada en un fichero de texto que sigue un estándar denominado **XML**:



Ventajas de los gráficos vectoriales

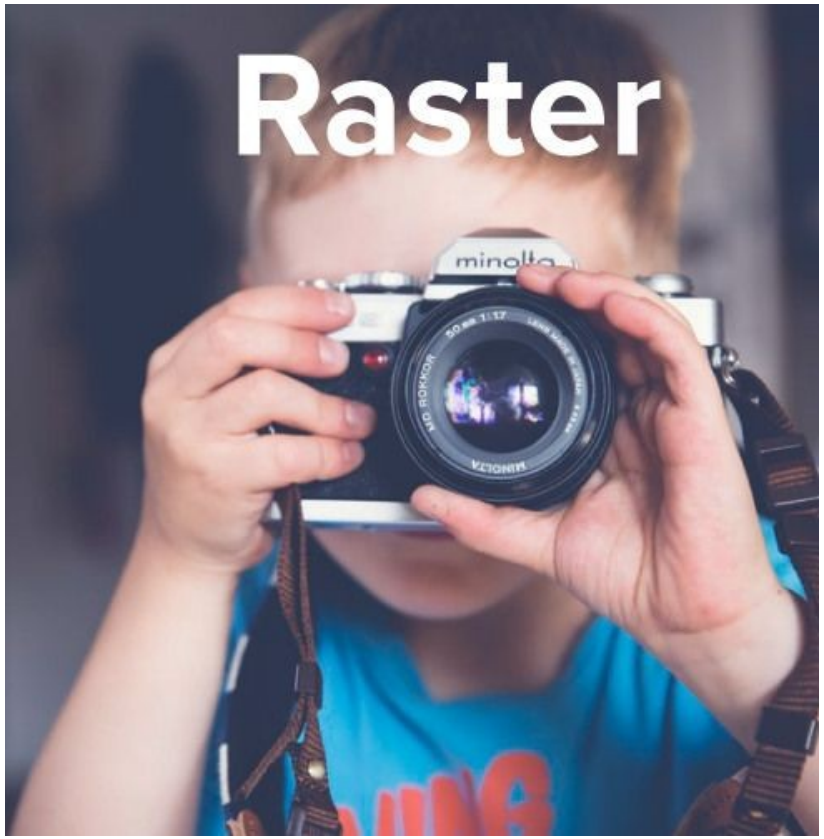
Los gráficos vectoriales ofrecen las siguientes ventajas en comparación con los mapas de bits:

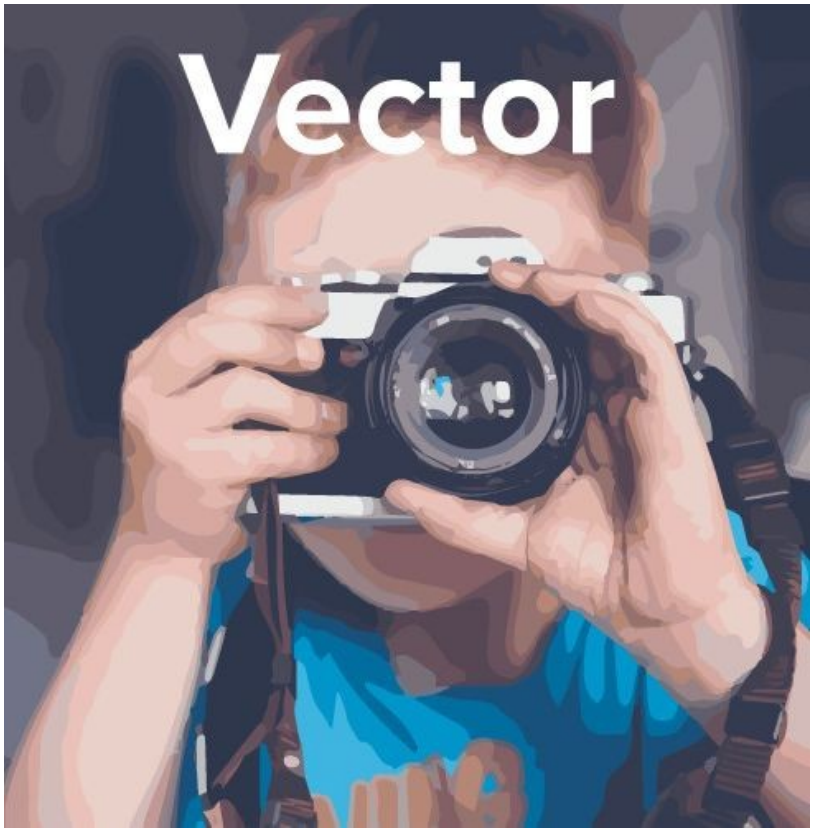
- Podemos **ampliar la imagen sin sufrir la pérdida de calidad** con la que nos encontrábamos con los mapas de bits
- **Flexibilidad de edición**. Podemos seleccionar las formas de la imagen y modificarlas de una forma relativamente sencilla
- En **imágenes muy grandes el fichero** resultante de un gráfico vectorial **suele ser mucho más pequeño** que el equivalente en una imagen de mapa de bits



Ventajas de los mapas de bits

- Los mapas de bits representan mucho mejor las variaciones y gradientes de color, lo cual las hace más adecuadas en **imágenes que requieren realismo fotográfico**
- Por eso también disponen de una **gran variedad de efectos de imagen**, en comparación con los gráficos vectoriales

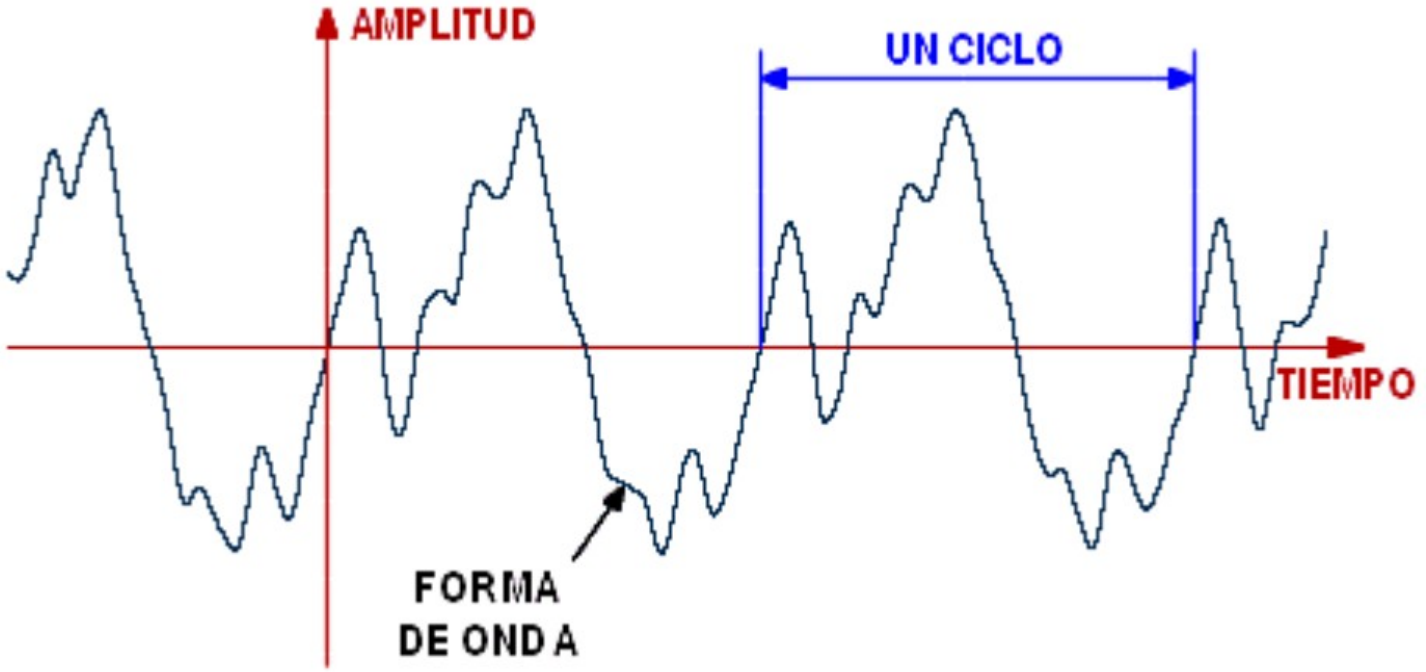




Codificación del audio

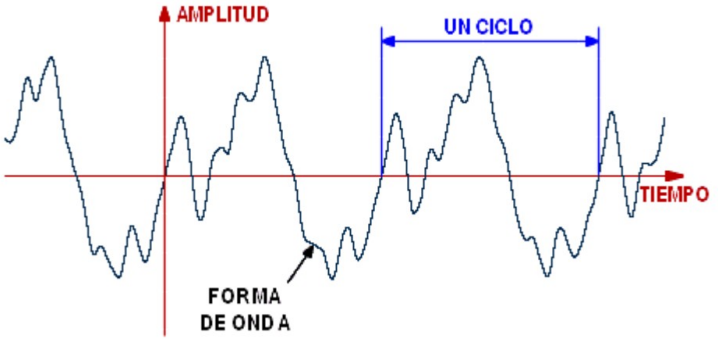
La representación del sonido

- Podemos representar el sonido en un **eje cartesiano** en el que:
  - El eje horizontal es el **tiempo**
  - El eje vertical es la **amplitud de onda**. Esta nos determina la potencia del sonido
  - Cuantas más veces atraviese la onda el eje horizontal en un mismo intervalo de tiempo, mayor será su **frecuencia**. La frecuencia entre otras cosas determina si un sonido es más agudo (frecuencias más altas) o más grave (frecuencias más bajas)
- Varias ondas sonoras pueden combinarse de forma que obtendremos una nueva onda resultado de sumar las amplitudes en cada instanteus características de amplitud y frecuencia.



https://eltamiz.com/elcedazo/2011/03/30/eso-que-llamamos-musica-a-proposito-del-sonido/

Representación del sonido en el ordenador I



- Para codificar el sonido en un ordenador:
  - **Se define la calidad de la muestra** que suele ser de **16 o 32 bits**. Estos bits son los que utilizaremos para guardar el valor numérico de la amplitud. Cuantos más bits mejor calidad
  - Se define la **frecuencia de toma de la muestra**, que no tiene nada que ver con la frecuencia de la onda de la que hablamos antes. El número de veces que vamos a tomar el valor de amplitud de la onda. Frecuencias habituales son:
    - Telefonía → 8 kHz (8000 muestras por segundo)
    - Radio → 22 kHz (22000 muestras por segundo)
    - CD → 44,1 kHz (44100 muestras por segundo)
- De esta forma guardamos una serie de puntos que representan parcialmente la onda de sonido original
- Cuanto mayor sea la frecuencia, más muestras tendremos para representar la onda original más fielmente

Representación del sonido en el ordenador II

- Además podemos tener varios canales de sonido y cada uno de ellos tendrá su audio independiente:
  - 1 canal → Sonido mono
  - 2 canales → Estéreo
  - 6 canales → Dolby 5.1
- Así podemos calcular lo que ocupa el sonido en base al siguiente cálculo:

Tamaño = N.º de canales * Calidad de la muestra * Frecuencia * duración

- **Ejemplo**

¿Cuánto ocupa un audio **estéreo** de **30 segundos** con una calidad de **32 bits** y una frecuencia de **22KHz**?

Tamaño = 2 canales * 32 bits * 22000 muestras * 30 segundos =

42.240.000 bits = 5.280.000 bytes = 5,28 MB

Compresión de audio

- Al igual que nos pasa con las imágenes el audio en un ordenador es habitual que pase por procesos de compresión que reducen el tamaño de los ficheros
- El cálculo que hicimos antes es **sin aplicar ningún tipo de compresión**
- En el caso del audio los algoritmos de compresión se llaman **códecs de audio** y al igual que nos pasaba con las imágenes también los tenemos de dos tipos:
  - **Algoritmos de compresión sin pérdida** → Eliminan información redundante que se puede replicar.
  - **Algoritmos de compresión con pérdida** → Estos se aprovechan de las limitaciones del sistema auditivo humano que no percibe todas las frecuencias
- Al igual que pasa con las imágenes en los algoritmos con pérdida podemos regular el nivel de compresión. Cuando mayor nivel de compresión apliquemos menos fiel será el audio comprimido al original

Ejemplos de formatos de audio

- **WAV y AIFF**: Son dos formatos con compresión sin pérdidas. El primero fue creado por IBM y Microsoft y el segundo por Apple. Mantienen lo más posible la calidad del audio original a costa de generar archivos de gran tamaño.
- **MP3**: Es un algoritmo de compresión **con pérdidas** con formato propietario. El MP3 se hizo muy popular en los años 90, pero después aparecieron otros formatos de compresión que mejoraron considerablemente la calidad de audio que daba el MP3, por lo que hoy en día no es la mejor opción. Era un formato propietario hasta el año 2017.
- **AAC:** Formato **con pérdida** que ofrece una calidad similar al MP3 pero necesitando menos espacio.
- **OGG**, es un formato **con pérdida** libre y abierto que entre otros es el que utiliza Spotify. Mejora la calidad de audio del MP3, sobre todo en frecuencias altas
- **FLAC** (Free Lossless Audio Codec)**.** Es un formato de compresión **sin pérdida**, que comprime el tamaño hasta un 50% o 60% sin perder apenas calidad. Es el formato que utiliza la plataforma Tidal.
- **ALAC** (Apple Lossless Audio Codec). El formato de Apple como alternativa al FLAC. Es el que se utiliza en Itunes y otros servicios de Apple

El MP3 y la industria discográfica

- El formato MP3 se le atribuyen los inicios del cambio de modelo en el que escuchamos música ya que cuando apareció empezó el cambio de utilizar un **soporte físico** para escuchar música (Discos de vinilo, CD de audio o cinta de casete) a un **formato digital fácilmente distribuible** a través de Internet gracias a su reducido tamaño. Poco después de popularizarse el formato fueron múltiples los reproductores MP3 que llegaron al mercado de consumo
- Durante muchos años fueron múltiples las plataformas de compartir archivos tipo P2P (Peer to Peer) que se utilizaron para compartir música, entre otras cosas: Emule, Kazaa, torrent, … pero la que quizá se hizo más popular en sus inicios fue Napster (Creada en 1999) por sus demandas con la industria discográfica
- Esto cambio considerablemente el modelo de la industria discográfica que basaba su volumen de ventas en la venta del soporte físico y que paso por un largo proceso de asimilación hasta que llegamos al modelo actual en el que escuchar música por servicios de streaming es lo más habitual

napster

Codificación del vídeo

Representación del vídeo

- Para representar el vídeo ya tenemos todo el trabajo hecho ya que:
  - Es una secuencia de imágenes en un tiempo determinado
  - Estas imágenes van acompañadas de un audio
- Cómo ya sabemos como calcular el tamaño de una imagen sólo necesitamos saber cuantas imágenes componen el vídeo, y esto viene determinado por un parámetro denominado **frames por segundo (fps)**.
- La interfaz entre el cerebro y la visión del ser humano puede procesar de 10 a 12 imágenes separadas por segundo, percibiéndolas individualmente (si se excede este número la percibirá como movimiento)
- Aún así lo mínimo deberían ser 30 frames por segundo

Resoluciones de vídeo populares hoy en día

Calculo del tamaño de un fragmento de vídeo

Ejemplo:

¿Cuánto ocupará un vídeo de 30 segundos grabado con una resolución de 640x480 a 30 fps, 32 bits de profundidad de color y un sonido estéreo de 32 bits y una frecuencia de 22 kHz?

- **Imágenes**:
  - 640 * 480 * 32 = 9.830.400 bits por cada imagen = 1.228.800 bytes = 1,2288 MB tamaño por imagen
  - 1,2288 MB * 30 fps * 30 segundos = **1105,92 MB** el tamaño de todas las imágenes
- **Audio**
  - 2 canales * 32 bits * 22000 muestras por segundo * 30 segundos = 42.240.000 bits = 5.280.000 bytes = **5,28 MB**
- **Sumamos imágenes y audio**
  - 1105,92 MB + 5,28 MB = 1111,2 MB = **1,1112 GB**

Formatos de vídeo

- En el caso del vídeo también se aplican algoritmos de compresión con los que reducimos el espacio que ocupa un fichero de vídeo en disco, pero en este caso los formatos son más complejos, porque:
  - A diferencia de otros formatos la extensión de un fichero de vídeo no es indicativo del formato del fichero, sino que sólo indican el contenedor usado
  - Los contenedores pueden contener varios archivos de audio, vídeo, imágenes, pistas de subtítulos, información adicional para sincronizar audio y vídeo
  - Según el formato en los vídeos se utilizan múltiples códecs de audio y vídeo
- Es el software de reproducción de vídeo el que debe ser capaz de examinar el contenido del contenedor y además de reproducirlo correctamente darnos opciones de configuración adicionales (Cambiar la pista de audio, mostrar o ocultar subtítulos, ...)
- El contenido de audio y el vídeo del contenedor se codifica utilizando **códecs, que son los que opcionalmente realizan la compresión sobre el audio y el vídeo**
- También tendremos códecs que realizan la compresión con o sin pérdida, pero lo habitual es utilizar con algún tipo de pérdida salvo casos en los que sepamos que en el futuro tendremos que editar de nuevo este vídeo

Ejercicio: Cálculos de tamaños de ficheros

- Los audios de un videojuego tienen una duración total de 10 minutos y están grabados para dolby 5.1 de 6 canales, una frecuencia de calidad cd de 44.1 KHz y una calidad de 32bits. ¿Cuanto ocupan en disco todos los audios del disco si no se aplica ningún tipo de compresión?
- Tenemos un móvil que graba vídeos con una resolución de 320x200 con 16 bits de colo y a 20 fps. El sonido es mono, con calidad de 16 bits e 20 kHz.
  - ¿Cuanto ocupa un vídeo de 1 minuto de duración si no se aplica ningún tipo de compresión?
  - Si tenemos una tarjeta de memoria de 1GB, ¿Cuanto tiempo de grabación podemos almacenar en la tarjeta de memoria?