

# Representação da Informação



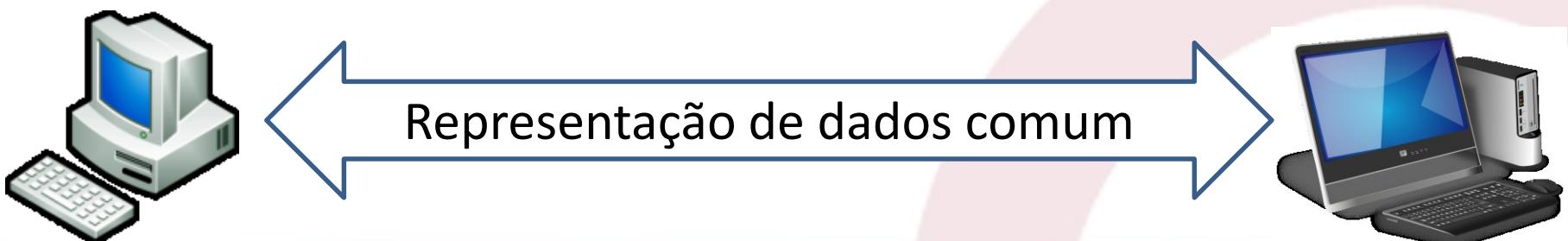
Autor: Patrício Domingues

Baseado em: "Representação da Informação",  
Mário Antunes & Vitor Carreira

Atualizado em 2019

# Enquadramento

- Níveis de heterogeneidade
  - A **arquitetura física**, os **sistemas operativos** e as **linguagens de programação** condicionam a representação de dados
  - São elementos indutores da heterogeneidade
    - Representação de dados diferentes
- A comunicação entre computadores requer uma representação uniforme

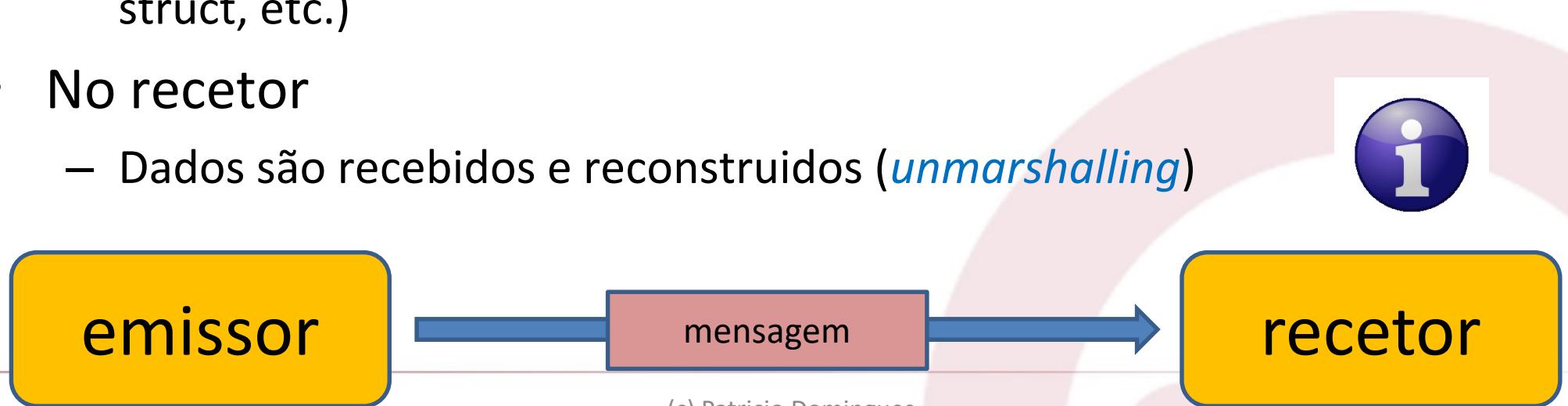


# Mensagens (1)

- Processos distribuídos
  - Processos que executam em máquinas remotas
- Comunicação
  - Por mensagens: bloco básico de comunicação entre processos distribuídos
  - Emissor e recetor
    - Muitas vezes a comunicação é bidirecional
- Alguns modelos de programação distribuída
  - Remote Procedure Calls (RPC)
  - Remote Method Invocation (RMI)
  - Event-based programming (serviços de notificação)

# Mensagens (2)

- Os dados usados por aplicações distribuídas podem ser estruturas complexas (e.g., listas, arrays, etc...)
- A mensagem enviada entre o cliente e o servidor tem de ser **serializada (linearizada)**, ou seja, tornada *flattened*
- No emissor
  - Dados são serializados e enviados para a rede (*marshalling*)
  - A informação enviada pode conter diferentes tipos de dados (e.g., struct, etc.)
- No recetor
  - Dados são recebidos e reconstruídos (*unmarshalling*)



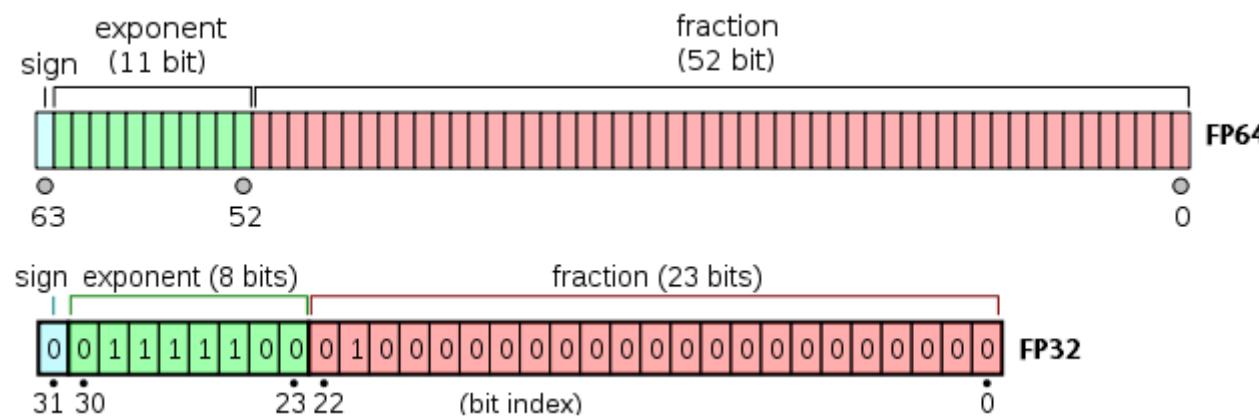
- A representação dos dados varia de arquitetura para arquitetura (x86, power PC, SPARC) , e de sistema operativo para sistema operativo (Windows, Linux, Mac OS,...)
- Diferenças encontradas:
  - Tamanho da palavra do processador
  - Ordem de representação dos dados em memória
  - Representação de números em vírgula flutuante
    - IEEE 754 (última atualização 2008)
  - Character set de representação
    - ASCII
    - EBCDIC
    - Unicode
    - ...



# Tamanho da palavra do processador

- Usualmente
  - tamanho de um inteiro (“int”) apresenta o tamanho da palavra (*word*) do processador
  - tamanho dos registos do processador
- O tamanho de um inteiro pode surgir com 16, 32, 64 ou 128 bits
- Este valor depende do tamanho definido para um inteiro pelo sistema operativo
- Indirectamente está associado ao tamanho da palavra do processador:
  - x86: 32 bits
  - AMD 64: 64 bits (num S.O. de 64 bits) e 32 bits (num S.O. de 32 bits)

- Norma IEEE 754 -2008
  - Representação de números em vírgula flutuante
    - Dois tipos principais - float: 32 bits e double 64 bits
    - Exemplo – representação em float de -248.75
      - binário: 1100.0011.0111.1000.1100.0000.0000.0000

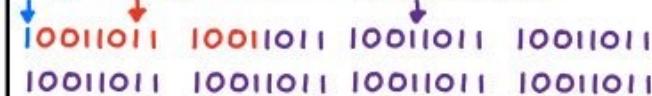


<https://bit.ly/32JcYHt>

# floating point

JULIA EVANS  
@bork

a double is 64 bits

sign exponent fraction  


$$\pm 2^{e-1023} \times 1.\text{frac}$$

That means there are  $2^{64}$  doubles

The biggest one is about

$$2^{1023}$$

doubles get farther apart as they get bigger

between  $2^n$  and  $2^{n+1}$  there are always  $2^{52}$  doubles, evenly spaced

that means the next double after  $2^{60}$  is  $2^{60} + 64 = \frac{2^{60}}{2^{52}}$

## weird double arithmetic

$$2^{52} + 0.2 = 2^{52}$$

← (the next number after  $2^{52}$  is  $2^{52} + 1$ )

$$1 + \frac{1}{2^{54}} = 1$$

← (the next number after 1 is  $1 + \frac{1}{2^{52}}$ )

$$2^{2000} = \text{infinity}$$

← infinity is a double

$$\text{infinity} - \text{infinity} = \text{nan}$$

← nan = "not a number"

doubles get farther apart as they get bigger

between  $2^n$  and  $2^{n+1}$  there are always  $2^{52}$  doubles, evenly spaced

that means the next double after  $2^{60}$  is  $2^{60} + 64 = \frac{2^{60}}{2^{52}}$

Javascript only has doubles (no integers!)

$$> 2**53$$

9007199254740992

$$> 2**53 + 1$$

9007199254740992

↑  
same number! uh oh!



doubles are scary and their arithmetic is weird

they're very logical!  
just understand how they work and don't use integers over  $2^{53}$  in Javascript ♥



- Uma representação binária de 32 bits...
- 1100.0011.0111.1000.1100.0000.0000.0000
  - Interpretado como **FP32**:
    - -248.75
  - Interpretado como **int32**:
    - -1015496704
  - Interpretado como **unsigned int 32**:
    - 3279470592

- Calcular a soma de 1E-9, 1E9 vezes
  - $1\text{E-9} = 1 \times 10^{-9}$  e  $1\text{E9} = 1 \times 10^9$
  - O resultado algébrico é 1 pois temos:  $1\text{E-9} \times 1\text{E9} = 1$
- Programa em C

```
#include <stdio.h>
int main(void){
    double sum = 0.0;
    double parcel = 1E-9;
    int i;
    for(i=0;i<1E9;i++){
        sum += parcel;
    }
    printf("Sum=%0.9f (expecting: 1.0)\n", sum);
    return 0;
}
```

$$\sum_{1}^{10^9} 10^{-9} = 1$$

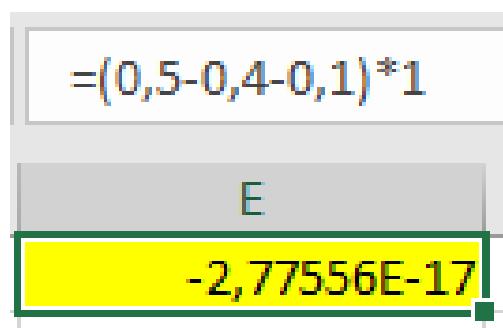
```
C:\Users\user>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019,
Type "help", "copyright", "credits" or "license" fo
>>> [0.1]*10
[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
>>> sum([0.1]*10)
0.9999999999999999
>>> sum([0.01]*100)
1.0000000000000007
>>>
```

python 3

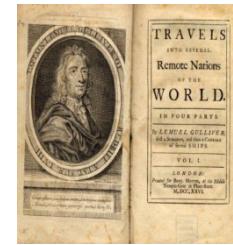
Resultado: Sum=0.999999993 (expecting: 1.0)

(c) Patrício Domingues

- Outras aplicações têm problemas similares
  - Exemplo: *Microsoft Excel*
    - $=(0,5-0,4-0,1)*1$
    - Resultado deveria ser 0
    - Contudo...



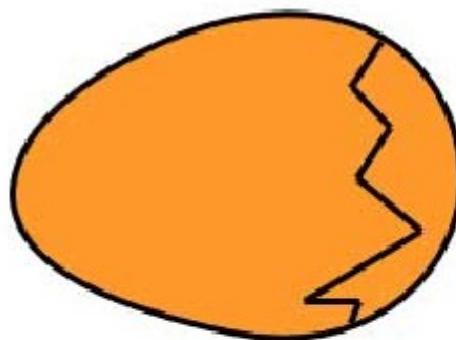
# Endianess (1)

- Representação de dados em memória - “Endianess”
  - Origem histórica do termo
    - Danny Cohen inspirou-se no livro “As Viagens de Gulliver”, Jonathan Swift (1726) para o nome *endianess*
    - 
    - 
    - “Danny Cohen; “On Holy Wars and a Plea for Peace” [<http://www.networksorcery.com/enp/ien/ien137.txt>, 1980]
    - Danny Cohen, "On Holy Wars and a Plea for Peace", IEEE Computer Magazine, October 1981.
  - Duas classes principais de arquiteturas
    - Little endian
    - Big endian

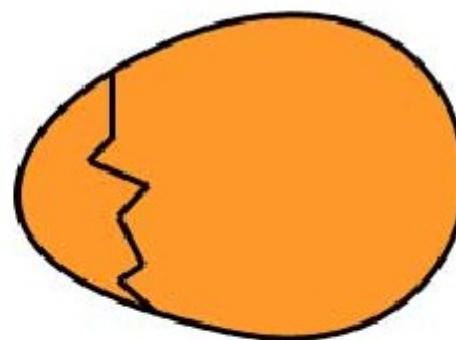
Big e little endian >>

# Endianess (2)

- O big endian e o little endian segundo “As viagens de Gulliver”...



**BIG ENDIAN** - The way people always broke their eggs in the Lilliput land



**LITTLE ENDIAN** - The way the king then ordered the people to break their eggs

# Gulliver's Travels

- Gulliver's Travels - Part 1: A voyage to Lilliput
  - Extracto do capítulo 4

moons make no mention of any other regions than the two great empires of Lilliput and Blefuscu. Which two mighty powers have, as I was going to tell you, been engaged in a most obstinate war for six-and-thirty moons past. It began upon the following occasion. It is allowed on all hands, that the primitive way of breaking eggs, before we eat them, was upon the larger end; but his present majesty's grandfather, while he was a boy, going to eat an egg, and breaking it according to the ancient practice, happened to cut one of his fingers. Whereupon the emperor his father published an edict, commanding all his subjects, upon great penalties, to break the smaller end of their eggs. The people so highly resented this law, that our histories tell us, there have been six rebellions raised on that account; wherein one emperor lost his life, and another his crown. These civil commotions were constantly foisted by the monarchs of Blefuscu; and when they were quelled, the exiles always fled for refuge to that empire. It is computed that eleven thousand persons have at several times suffered death, rather than submit to break their eggs at the smaller end. Many hundred large volumes have been published upon this controversy: but the books of the Big-endians have been long forbidden, and the whole party rendered incapable by law of holding employments.

# Endianess (3)

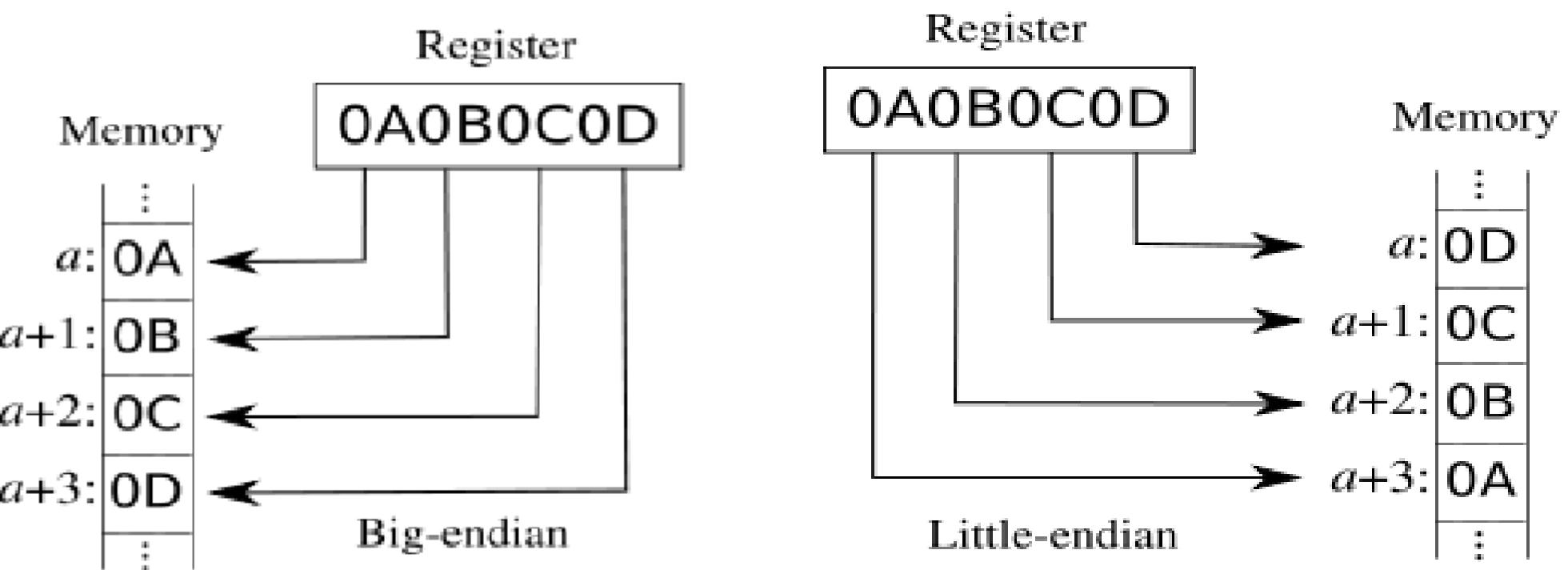
- Big Endian
  - Uma palavra do processador é armazenada com o byte mais significativo a ocorrer primeiro em memória
    - Exemplos: Sun Sparc (até versão 9), Motorola 68x0 (museu!)
- Little Endian
  - Uma palavra do processador é armazenada com o byte menos significativo a ocorrer primeiro em memória
    - Exemplos: x86, x86-64, ARM (até versão 3), Dec Alpha e Z80 (museu!)
- Bi-Endian
  - Sistemas que suportam big e little endian (configurados no arranque ou por segmentos de dados)
    - Exemplos: IA64 (itanium), PowerPC, MIPS, ARM (>=3), SPARC (>=9)

# *Big e Little endian (1)*

- Considerar A = 0x11223344
- Representação de A na memória pode ser
  - Byte 0x11, seguido de byte 0x22, seguido de byte 0x33, seguido de byte 0x44
    - Byte mais significativo (0x11) ocorre primeiro na memória
      - BigEndian
    - Byte 0x44, seguido de byte 0x33, seguido de byte 0x22, seguido de byte 0x11
      - Byte menos significativo (0x44) ocorre primeiro na memória
        - LittleEndian

Big	44	33	22	11
Little	11	22	33	44

# *Big e Little endian (2)*



# Endianess: programa

```
int main(void) {
    long test = 0x11223344;
    char *ptr; /* Ptr. p/ char => acesso byte a byte */
    assert (sizeof(long)==4);
    ptr = (char *)&test;
    if ((0x11 == ptr[0]) && (0x22 == ptr[1])
        && (0x33 == ptr[2]) && (0x44 == ptr[3]))
    {
        printf ("Big endian.\n");
    } else if ((0x44 == ptr[0]) && (0x33 == ptr[1])
        && (0x22 == ptr[2]) && (0x11 == ptr[3]))
    {
        printf("Little endian.\n");
    }
} else
    printf("Arquitetura desconhecida.\n");
}
```

# Endianess – troca de informação (#1)

- Como trocar informação entre dois sistemas?
  - Os dois sistemas podem ou não seguir o mesmo *endianess*
  - Duas possíveis soluções
    - Usar representação comum
      - Formato de rede (big endian)
    - Usar marcador no início dos dados
      - BOM – Byte Order Mark



Formato de rede >>

- Usar representação comum
  - No caso de dados trocados pela rede, o formato de ordenação é o *Big Endian*
    - Formato de rede
  - Sempre que um sistema *Little Endian* transmite para a rede, os seus dados sujeitos ao “endian da arquitetura” devem ser convertidos em Big Endian
  - Na receção, caso se trate de um *Little Endian*, ocorre conversão do formato de rede (i.e. *Big Endian*) para *Little Endian*
  - Os sistemas *big endian* não precisam de converter

# Endianess – troca de informação (#3)

- Marcador BOM
  - Byte Order Mark
    - Uso de número mágico como marcador
      - Exemplo: 0xFEFF
    - O marcador é escrito antes de cada bloco de dados
      - 0xFEFF[DADOS]
    - O processo que efetua a leitura verifica o BOM
      - Se BOM for lido como 0xFEFF, então os dois sistemas usam a mesma representação
      - Se BOM for lido como 0xFFFF (trocado), então os dois sistemas usam representação diferenciada, sendo necessário conversão
    - O BOM é empregue nas codificações UTF (UTF-16 e UTF-32)

# Codificação ASCII

- ASCII
  - American Standard Code for Information Interchange
  - Criado em 1968
    - 128 códigos (requerem 7 bits para serem representados)
  - Sendo uma norma definida pelos Estados Unidos,  
apenas lida com caracteres não acentuados
- Extensão do ASCII a 256 elementos
  - 8 bits
  - Do 128 ao 255 é empregue uma versão localizada
    - Varia consoante os países (PT: ç, ã, é, etc.)

# Tabela ASCII

- Versão base
  - Código de 0 a 127

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0 000		<b>NUL</b> (null)	32	20 040		&#32;	<b>Space</b>	64	40 100		&#64;	<b>Ø</b>	96	60 140		&#96;	<b>`</b>
1	1 001		<b>SOH</b> (start of heading)	33	21 041		&#33;	<b>!</b>	65	41 101		&#65;	<b>A</b>	97	61 141		&#97;	<b>a</b>
2	2 002		<b>STX</b> (start of text)	34	22 042		&#34;	<b>"</b>	66	42 102		&#66;	<b>B</b>	98	62 142		&#98;	<b>b</b>
3	3 003		<b>ETX</b> (end of text)	35	23 043		&#35;	<b>#</b>	67	43 103		&#67;	<b>C</b>	99	63 143		&#99;	<b>c</b>
4	4 004		<b>EOT</b> (end of transmission)	36	24 044		&#36;	<b>\$</b>	68	44 104		&#68;	<b>D</b>	100	64 144		&#100;	<b>d</b>
5	5 005		<b>ENQ</b> (enquiry)	37	25 045		&#37;	<b>%</b>	69	45 105		&#69;	<b>E</b>	101	65 145		&#101;	<b>e</b>
6	6 006		<b>ACK</b> (acknowledge)	38	26 046		&#38;	<b>&amp;</b>	70	46 106		&#70;	<b>F</b>	102	66 146		&#102;	<b>f</b>
7	7 007		<b>BEL</b> (bell)	39	27 047		&#39;	<b>'</b>	71	47 107		&#71;	<b>G</b>	103	67 147		&#103;	<b>g</b>
8	8 010		<b>BS</b> (backspace)	40	28 050		&#40;	<b>(</b>	72	48 110		&#72;	<b>H</b>	104	68 150		&#104;	<b>h</b>
9	9 011		<b>TAB</b> (horizontal tab)	41	29 051		&#41;	<b>)</b>	73	49 111		&#73;	<b>I</b>	105	69 151		&#105;	<b>i</b>
10	A 012		<b>LF</b> (NL line feed, new line)	42	2A 052		&#42;	<b>*</b>	74	4A 112		&#74;	<b>J</b>	106	6A 152		&#106;	<b>j</b>
11	B 013		<b>VT</b> (vertical tab)	43	2B 053		&#43;	<b>+</b>	75	4B 113		&#75;	<b>K</b>	107	6B 153		&#107;	<b>k</b>
12	C 014		<b>FF</b> (NP form feed, new page)	44	2C 054		&#44;	<b>,</b>	76	4C 114		&#76;	<b>L</b>	108	6C 154		&#108;	<b>l</b>
13	D 015		<b>CR</b> (carriage return)	45	2D 055		&#45;	<b>-</b>	77	4D 115		&#77;	<b>M</b>	109	6D 155		&#109;	<b>m</b>
14	E 016		<b>SO</b> (shift out)	46	2E 056		&#46;	<b>.</b>	78	4E 116		&#78;	<b>N</b>	110	6E 156		&#110;	<b>n</b>
15	F 017		<b>SI</b> (shift in)	47	2F 057		&#47;	<b>/</b>	79	4F 117		&#79;	<b>O</b>	111	6F 157		&#111;	<b>o</b>
16	10 020		<b>DLE</b> (data link escape)	48	30 060		&#48;	<b>0</b>	80	50 120		&#80;	<b>P</b>	112	70 160		&#112;	<b>p</b>
17	11 021		<b>DC1</b> (device control 1)	49	31 061		&#49;	<b>1</b>	81	51 121		&#81;	<b>Q</b>	113	71 161		&#113;	<b>q</b>
18	12 022		<b>DC2</b> (device control 2)	50	32 062		&#50;	<b>2</b>	82	52 122		&#82;	<b>R</b>	114	72 162		&#114;	<b>r</b>
19	13 023		<b>DC3</b> (device control 3)	51	33 063		&#51;	<b>3</b>	83	53 123		&#83;	<b>S</b>	115	73 163		&#115;	<b>s</b>
20	14 024		<b>DC4</b> (device control 4)	52	34 064		&#52;	<b>4</b>	84	54 124		&#84;	<b>T</b>	116	74 164		&#116;	<b>t</b>
21	15 025		<b>NAK</b> (negative acknowledge)	53	35 065		&#53;	<b>5</b>	85	55 125		&#85;	<b>U</b>	117	75 165		&#117;	<b>u</b>
22	16 026		<b>SYN</b> (synchronous idle)	54	36 066		&#54;	<b>6</b>	86	56 126		&#86;	<b>V</b>	118	76 166		&#118;	<b>v</b>
23	17 027		<b>ETB</b> (end of trans. block)	55	37 067		&#55;	<b>7</b>	87	57 127		&#87;	<b>W</b>	119	77 167		&#119;	<b>w</b>
24	18 030		<b>CAN</b> (cancel)	56	38 070		&#56;	<b>8</b>	88	58 130		&#88;	<b>X</b>	120	78 170		&#120;	<b>x</b>
25	19 031		<b>EM</b> (end of medium)	57	39 071		&#57;	<b>9</b>	89	59 131		&#89;	<b>Y</b>	121	79 171		&#121;	<b>y</b>
26	1A 032		<b>SUB</b> (substitute)	58	3A 072		&#58;	<b>:</b>	90	5A 132		&#90;	<b>Z</b>	122	7A 172		&#122;	<b>z</b>
27	1B 033		<b>ESC</b> (escape)	59	3B 073		&#59;	<b>;</b>	91	5B 133		&#91;	<b>[</b>	123	7B 173		&#123;	<b>{</b>
28	1C 034		<b>FS</b> (file separator)	60	3C 074		&#60;	<b>&lt;</b>	92	5C 134		&#92;	<b>\</b>	124	7C 174		&#124;	<b> </b>
29	1D 035		<b>GS</b> (group separator)	61	3D 075		&#61;	<b>=</b>	93	5D 135		&#93;	<b>]</b>	125	7D 175		&#125;	<b>}</b>
30	1E 036		<b>RS</b> (record separator)	62	3E 076		&#62;	<b>&gt;</b>	94	5E 136		&#94;	<b>^</b>	126	7E 176		&#126;	<b>~</b>
31	1F 037		<b>US</b> (unit separator)	63	3F 077		&#63;	<b>?</b>	95	5F 137		&#95;	<b>_</b>	127	7F 177		&#127;	<b>DEL</b>

Source: [www.LookupTables.com](http://www.LookupTables.com)

# Ainda sobre a tabela ASCII

- Caracteres de controlo ASCII (código de 0 a 31)
  - Código: 10 LF - line feed ('\n')
  - Código: 13 CR – Carriage Return ('\r')
- Caracteres visíveis base (código de 32 a 127)
  - Código: 65 A ("A" maiúscula)
  - Código: 97 a ("a" minúscula)
- Código ASCII estendido (código de 128 a 255)
  - Exemplo ISO 8859-1 (ou ISO Latin-1)
    - Código 199 Ç (cedilha maiúscula)
    - Código 231 ç (cedilha minúscula)

# Mostrar a tabela ASCII

- Simples programa em linguagem C para mostrar os elementos da tabela ASCII (códigos de 32 a 255)

```
#include <stdio.h>
void ShowAscii (void) {
    int i;
    for (i=32;i<=255;i++) {
        printf ("%d=%'c'\n", i,i);
    }
}
int main (void) {
    ShowAscii ();
    return 0;
}
```

# ASCII estendido (#1)

- Norma ISO 8859
  - Série de ASCII 8 bits para representar alfabetos ocidentais
- ISO-8859-1 - Latin 1
  - Western Europe and Americas: Afrikaans, Basque, Catalan, Danish, Dutch, English, Faeroese, Finnish, French, Galician, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish and Swedish.
- ISO-8859-2 Latin 2
  - Latin-written Slavic and Central European languages: Czech, German, Hungarian, Polish, Romanian, Croatian, Slovak, Slovene.
- ISO-8859-3 - Latin 3
  - Esperanto, Galician, Maltese, and Turkish.
- ISO-8859-4 - Latin 4
  - Scandinavia/Baltic, Estonian, Latvian, and Lithuanian.
- ISO-8859-5 - Cyrillic
  - Bulgarian, Byelorussian, Macedonian, Russian, Serbian and Ukrainian.

**ISO-8859-5**

Ё	Ђ	Ѓ	€	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	-	Ӯ	ӽ
Ӑ	Ӗ	Ҫ	Ӗ	Ҫ	Ҫ	Ӗ	Ҫ	Ҫ	Ҫ	Ҫ	Ҫ	-	Ӯ	ӽ
Ӑ	Ӗ	Ҫ	Ӗ	Ҫ	Ҫ	Ӗ	Ҫ	Ҫ	Ҫ	Ҫ	Ҫ	-	Ӯ	ӽ
Ӑ	Ӗ	Ҫ	Ӗ	Ҫ	Ҫ	Ӗ	Ҫ	Ҫ	Ҫ	Ҫ	Ҫ	-	Ӯ	ӽ
Ӑ	Ӗ	Ҫ	Ӗ	Ҫ	Ҫ	Ӗ	Ҫ	Ҫ	Ҫ	Ҫ	Ҫ	-	Ӯ	ӽ

**(Continua >)**

# ASCII estendido (#2)

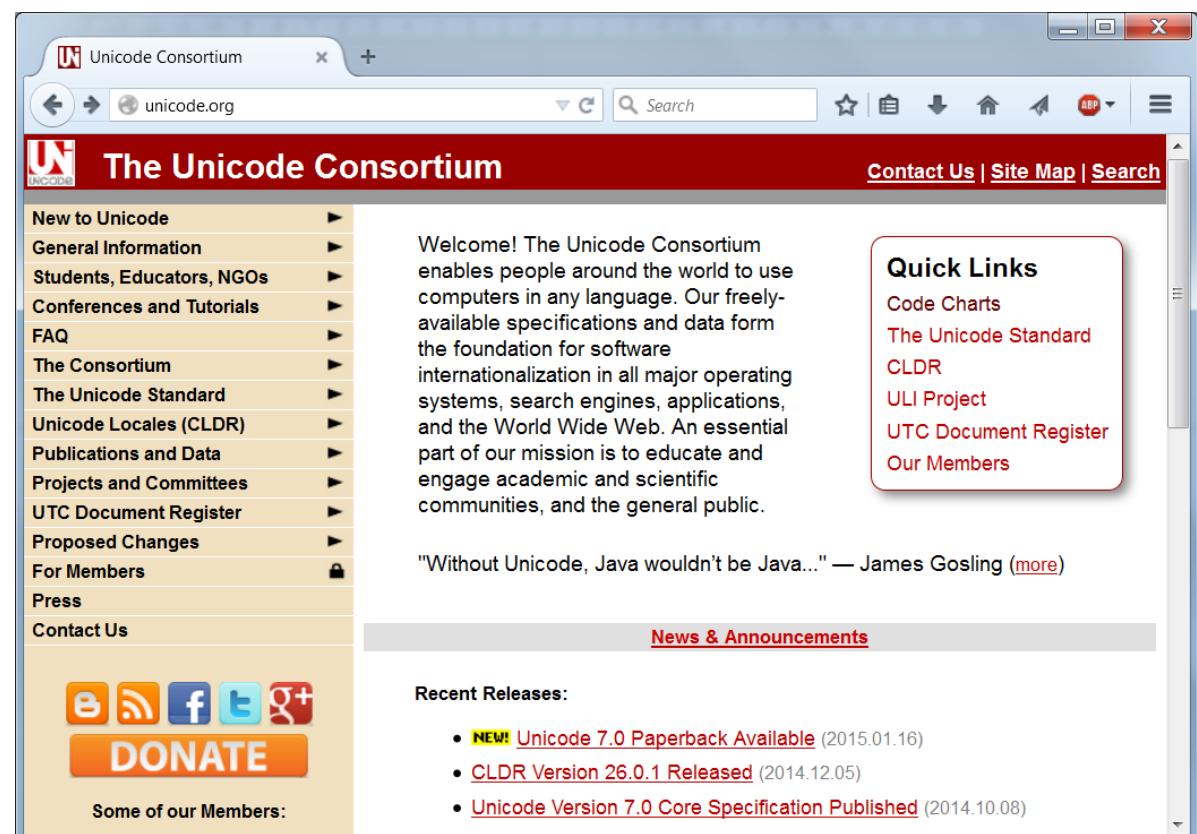
- Norma ISO 8859 (continuação)
  - Série de ASCII 8 bits para representar alfabetos ocidentais
- ISO-8859-6 - Arabic
  - Non-accented Arabic.
- ISO-8859-7- Modern Greek
  - Greek.
- ISO-8859-8 - Hebrew
  - Non-accented Hebrew.
- ISO-8859-9 - Latin 5
  - Same as 8859-1 except for Turkish instead of Icelandic
- ISO-8859-10 - Latin 6
  - Latin6, for Lappish/Nordic/Eskimo languages: Adds the last Inuit (Greenlandic) and Sami (Lappish) letters that were missing in Latin 4 to cover the entire Nordic area.
- **Fonte:** <https://www.terena.org/activities/multiling/ml-docs/iso-8859.html>

- 256 caracteres não chegam para algumas línguas
  - CJK
    - China (> 6000 ideogramas)
    - Japão (> 7400 ideogramas)
    - Coreia (> 8000 ideogramas)
  - Solução
    - Consórcio Unicode



As of June 1, 2019, the annual membership fee will change as described in the following table:

Membership Levels	Current Fee	New Fee
Full	\$18,000	\$21,000
Institutional, governmental	\$12,000	\$14,000
Supporting, for-profit organization	\$7,500	\$8,750
Supporting, non-profit organization	N/A	\$5,000
Associate, for-profit organization	\$2,500	\$2,900



The screenshot shows the homepage of the Unicode Consortium. The header features the "UNICODE" logo and the text "The Unicode Consortium". The main menu on the left includes links such as "New to Unicode", "General Information", "Students, Educators, NGOs", "Conferences and Tutorials", "FAQ", "The Consortium", "The Unicode Standard", "Unicode Locales (CLDR)", "Publications and Data", "Projects and Committees", "UTC Document Register", "Proposed Changes", "For Members", "Press", and "Contact Us". Below the menu is a "DONATE" button and a "Some of our Members:" section. The right side of the page contains a welcome message, a "Quick Links" sidebar with links to "Code Charts", "The Unicode Standard", "CLDR", "ULI Project", "UTC Document Register", and "Our Members", and a "Recent Releases" section with three items.

## Unicode >>

# Unicode (1)

- Norma para a codificação, representação e manipulação consistente dos símbolos empregues na escrita
- Norma Unicode 
  - Empregue na internacionalização (i18n) e localização de software
  - Empregue em alguns sistemas operativos, em linguagens de programação (JAVA, .NET) e em linguagens de estruturação de dados (XML)
- Porquê a designação i18n?
  - I+n<sub>1</sub>t<sub>2</sub>e<sub>3</sub>r<sub>4</sub>n<sub>5</sub>a<sub>6</sub>t<sub>7</sub>i<sub>8</sub>o<sub>9</sub>n<sub>10</sub>a<sub>11</sub>I<sub>12</sub>i<sub>13</sub>z<sub>14</sub>a<sub>15</sub>t<sub>16</sub>i<sub>17</sub>o<sub>18</sub>+n
- Fonte: <http://www.i18nguy.com/origini18n.html>

# Unicode (2)

- Norma para codificação de **símbolos** (engloba caracteres)
  - Usa *codepoints*
  - Codepoint: valor inteiro, usualmente representado em base 16 (i.e., formato hexadecimal)
  - Exemplo: U+12ca é *Unicode com código 0x12ca (4810 decimal)*
  - O unicode define *code points* entre 0 e 0x10FFFF
    - total de 1114112 *code points* - 2048 que são considerados inválidos
      - 1112064
- O unicode está dividido em 17 planos (0 a 16)
  - $17 * 65536$  (número de code points por plano) = 1,114,112
  - Plano 0 (Basic MultiPlane=BMP): 0x000000 a 0x00FFFF
  - Identificador de plano (0 a 16): 0x00----... a 0x10----
  - Presentemente, estão definidos cerca de 137000 símbolos

# Unicode (3)

- Não representa apenas carateres – exemplo
    - Fonte: <http://www.unicode.org/charts/PDF/U2600.pdf>

Miscellaneous Symbols															
260	261	262	263	264	265	266	267	268	269	26A	26B	26C	26D	26E	26F
0	☀️	❑	☠️	☰	♀	♐	♠	☦	❑	⚠️	❑	⌚	⌚	⌚	⌚
1	☁️	☑️	☭	☰	♂	♑	♥️	☦	❑	🚩	⚡	🏺	เหรียญ	🏧	☔
2	☂️	☒️	☢️	☰	♂	〰	◊	♻️	❑	⚒️	ヰ	⌚	⌚	⌚	⌚
3	☃️	✗	☣️	☰	♓	♓	♣️	♻️	❑	⚓	⚥	⌚	⌚	⌚	⌚
4	✂️	☂️	⚕️	☰	ܗ	ܰ	♠️	♻️	❑	✖️	♀	Ѡ	🚫	⭐️	🚢
5	★	☕️	☥	☰	ܱ	ܰ	♥️	♻️	❑	💲	♂	✳️	☀️	👉	⭐️
6	☆	🏡	₩	☰	ܲ	ܰ	♦️	♻️	❑	💲	♂	⤻	⤻	₩	⭐️
7	⚡	🏡	₭	☰	ܲ	ܰ	♣️	♻️	❑	ܱ	⚥	ܲ	ܲ	ܲ	⛷️

# Unicode (4)

- Outro exemplo de símbolos - EMOJI

Nº	Code	Brow.	Chart	Apple	Goog <sup>d</sup>	Twtr.	One	FBM	Wind.	Sams.	GMail	SB	DCM	KDDI	Name
1	<a href="#">U+1F600</a>											—	—	—	grinning face
2	<a href="#">U+1F601</a>														grinning face with smiling eyes
3	<a href="#">U+1F602</a>											—	—		face with tears of joy
4	<a href="#">U+1F923</a>			—							—	—	—	—	rolling on the floor laughing
5	<a href="#">U+1F603</a>														smiling face with open mouth
6	<a href="#">U+1F604</a>											—	—	—	smiling face with open mouth & smiling eyes
7	<a href="#">U+1F605</a>													—	smiling face with open mouth & cold sweat
8	<a href="#">U+1F606</a>										—		—	—	smiling face with open mouth & closed eyes
9	<a href="#">U+1F609</a>														winking face

<http://www.unicode.org/emoji/charts/full-emoji-list.html>

- Leitura recomendada: Erard M., “How the Appetite for Emojis Complicates the Effort to Standardize the World’s Alphabets”, NYTimes, (<http://nyti.ms/2yJfiCg>)

# Unicode “Scream”

- Outro exemplo de símbolo – Scream (E. Munch)

Hex Code Point(s)	e107
Formal Unicode Notation	U+E107
Decimal Code Point(s)	57607
UTF-8 Hex (C Syntax)	0xEE 0x84 0x87
UTF-8 Hex Bytes	EE 84 87
UTF-8 Octal Bytes	356 204 207
UTF-16 Hex (C Syntax)	0xE107
UTF-16 Hex	e107
UTF-16 Dec	57607
UTF-32 Hex (C Syntax)	0x0000E107
UTF-32 Hex	E107
UTF-32 Dec	57607
Python Src	u"\uE107"
PHP Src	"\xee\x84\x87"
C/C++/Java Src	"\uE107"

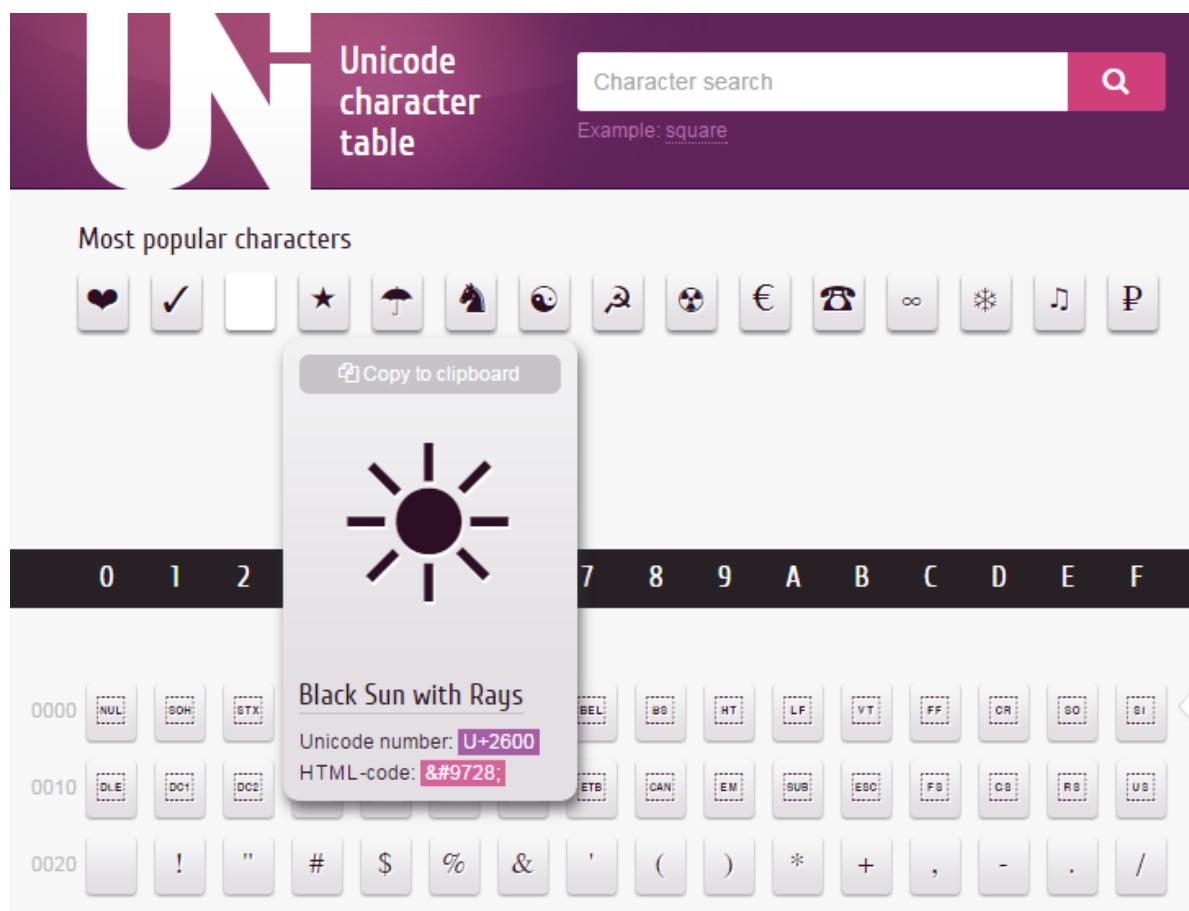


## Unicode Category Information

Unicode Category	Emoticons
Unicode Range	1F600–1F64F
Unicode Subcategory	Faces

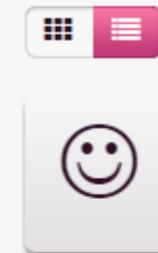
# Tabela *unicode* (online)

- <http://unicode-table.com/en/>



The screenshot shows the Unicode character table interface. At the top, there's a search bar with 'Character search' and a magnifying glass icon. Below it is an example input field with 'square' and a placeholder 'Example: square'. On the left, there's a sidebar with 'Most popular characters' and a grid of icons. In the center, there's a large preview area for the 'Black Sun with Rays' emoji, which is a black sun with eight rays. Below the preview, its details are listed: 'Black Sun with Rays', 'Unicode number: U+2600', and 'HTML-code: &#9728;'. The bottom part of the interface shows a grid of various other characters and symbols.

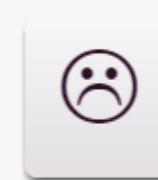
## Emoji



[White Smiling Face](#)

Miscellaneous Symbols

U+263A &#9786;



[White Frowning Face](#)

Miscellaneous Symbols

U+2639 &#9785;



[Black Smiling Face](#)

Miscellaneous Symbols

U+263B &#9787;



[Grinning Face](#)

Emoticons (Emoji)

U+1F600 &#128512;



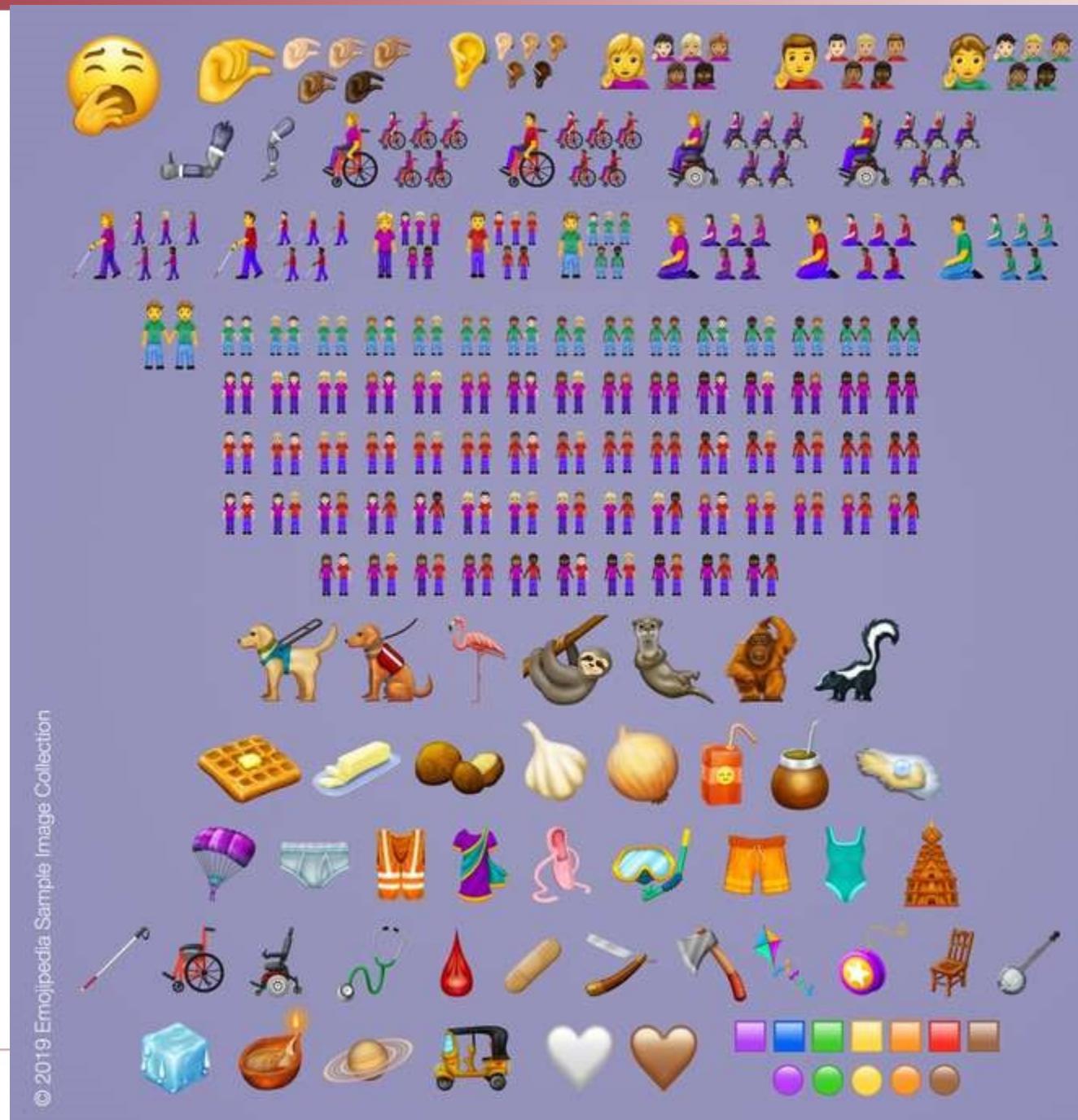
[Grinning Face with Smiling Eyes](#)

Emoticons (Emoji)

U+1F601 &#128513;

# Unicode – emojis - 2019

- Emojis  
acrescentados em  
2019
  - 59 novos emojis
  - 171 variantes de  
género e cor da  
pele
  - Total
    - $59+171 = 230$



# Unicode – candidatos Emoji

- <https://unicode.org/emoji/future/emoji-provisional.html>

When looking at the linked proposals for candidates or accepted characters it is important to keep two points in mind:

1. New proposals must follow the form in [Submitting Emoji Proposals](#). This form may have changed since earlier proposals were submitted.
2. The UTC may accept a proposal for reasons other than those stated in the proposal, and does not necessarily endorse or consider relevant all of the proposed reasons.

<a href="#">Smileys &amp; Emotion</a>							
<a href="#">face-hat</a>							
<a href="#">№</a>	<a href="#">Code</a>	<a href="#">Sample</a>	<a href="#">Samp2</a>	<a href="#">CLDR Short Name</a>	<a href="#">Other Keywords</a>	<a href="#">Attributes</a>	<a href="#">Proposal</a>
1	<a href="#">X101302</a>		—	ninja	fighter   hidden   stealth ↳ smiley (feedback welcome on whether to make human-form, which would 'cost' 12 emoji)	> 	<a href="#">L2/18-197</a>
2	<a href="#">X101311</a>		—	disguised face	disguise   face   glasses   incognito   nose	>  (1F973)	<a href="#">L2/18-311</a>
<a href="#">Animals &amp; Nature</a>							
<a href="#">animal-mammal</a>							
<a href="#">№</a>	<a href="#">Code</a>	<a href="#">Sample</a>	<a href="#">Samp2</a>	<a href="#">CLDR Short Name</a>	<a href="#">Other Keywords</a>	<a href="#">Attributes</a>	<a href="#">Proposal</a>
3	<a href="#">X101306</a>		—	mammoth	extinction   large   tusk   woolly	> 	<a href="#">L2/17-420</a>
<a href="#">animal-bird</a>							
4	<a href="#">X101304</a>		—	feather	flight   light   plumage	> 	<a href="#">L2/18-200</a>
5	<a href="#">X101307</a>		—	dodo	extinction   large   Mauritius	> 	<a href="#">L2/17-441</a>

- Codificação
  - As regras para representar um carácter unicode na memória (na forma de *bytes*) designam-se por codificação
- Codificações disponíveis com unicode
  - UTF-8
  - UTF-16
  - UTF-32
- UTF
  - UTF → Unicode Transformation Format

Vamos começar pela codificação...UTF-32 >>

- UTF-32
  - Usa 4 bytes (32 bits) para cada carácter unicode
  - Representação direta do *code point*
- Vantagem
  - Os *code points* são diretamente indexáveis
    - Exemplo: o 3º carácter de uma string unicode inicia-se no 8º byte
      - Cada carácter tem 4 bytes
- Desvantagens
  - 4 bytes por carácter é excessivo na maior dos casos
  - Não é portável devido ao *endianess*
  - Não é compatível com algumas funções do C (e.g., `strlen`)
- Em consequência disso, o UTF-32 é pouco usado...

- UTF-16
  - Uma das codificações do unicode
  - Tamanho variável
    - Usa 16 ou 32 bits (dependendo do carácter)
  - Para o plano 0 (BMP), o UTF-16 usa 16 bits para cada carácter
    - O valor do UTF-16 é igual ao *code point* do caratere
  - Para os restantes planos, o UTF-16 usa 32 bits (4 bytes)
  - Usa os valores de 0 a 0x10FFFF para codificar símbolos
  - O plano é representado pelo dois primeiros símbolos
    - 0x00----, 0x01----, 0x02----,...,0x10----
    - Cada plano varia de 0000 a FFFF (65536 símbolos)

# UTF-16 (2)

- UTF-16
  - Para determinação do *endianess*, o UTF-16 permite que os dois primeiros bytes sejam 0xFEFF
    - Byte Order Mark (BOM)
    - Se não existir BOM, deve assumir-se codificação *big endian*
- Uso
  - O UTF-16 é empregue internamente e nas APIs de texto do windows 2000, XP, 2003, Vista, 7, 8, 10.
    - As strings são codificadas com UTF-16

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00003E40	6F	00	6D	00	70	00	61	00	6E	00	79	00	4E	00	61	00	o.m.p.a.n.y.N.a.
00003E50	6D	00	65	00	00	00	00	00	4D	00	69	00	63	00	72	00	m.e.....M.i.c.r.
00003E60	6F	00	73	00	6F	00	66	00	74	00	20	00	43	00	6F	00	o.s.o.f.t. .C.o.
00003E70	72	00	70	00	6F	00	72	00	61	00	74	00	69	00	6F	00	r.p.o.r.a.t.i.o.
00003E80	6E	00	00	00	4C	00	12	00	01	00	46	00	69	00	6C	00	n...L.....F.i.l.
00003E90	65	00	44	00	65	00	73	00	63	00	72	00	69	00	70	00	e.D.e.s.c.r.i.p.
00003EA0	74	00	69	00	6F	00	6E	00	00	00	00	00	57	00	69	00	t.i.o.n.....W.i.
00003EB0	6E	00	64	00	6F	00	77	00	73	00	20	00	49	00	6E	00	n.d.o.w.s. .I.n.
00003EC0	73	00	74	00	61	00	6C	00	65	00	72	00	00	00	00	00	s.t.a.l.l.e.r...

- UTF-8
  - Elaborado para ser compatível com o ASCII (7 bits) e evitar os problemas de endianess do UTF-16 e UTF-32
    - Usa a mesma codificação do ASCII quando representa ASCII (código 0 a 127)
  - Se o código  $\geq 128$ 
    - Usa 2, 3 ou 4 bytes consoante o *code point*
    - No modo  $> 1$  byte, cada byte do UTF-8 está compreendido entre 128 e 255

Vantagens & desvantagens >>

# UTF-8 (2)

- UTF-8
  - 00000000 -- 0000007F: 0xxxxxxx
    - Codificado com 1 byte
  - 00000080 -- 000007FF: 110xxxxx **10**xxxxxx
    - Codificado com 2 bytes
  - 00000800 -- 0000FFFF: 1110xxxx **10**xxxxxx **10**xxxxxx
    - Codificado com 3 bytes
  - 00010000 -- 001FFFFF: 11110xxx **10**xxxxxx **10**xxxxxx **10**xxxxxx
    - Codificado com 4 bytes

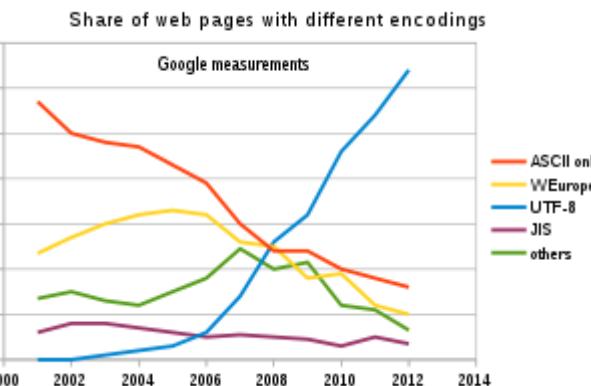
10 = octeto de continuação



Fonte: <http://www.cprogramming.com/tutorial/unicode.html>

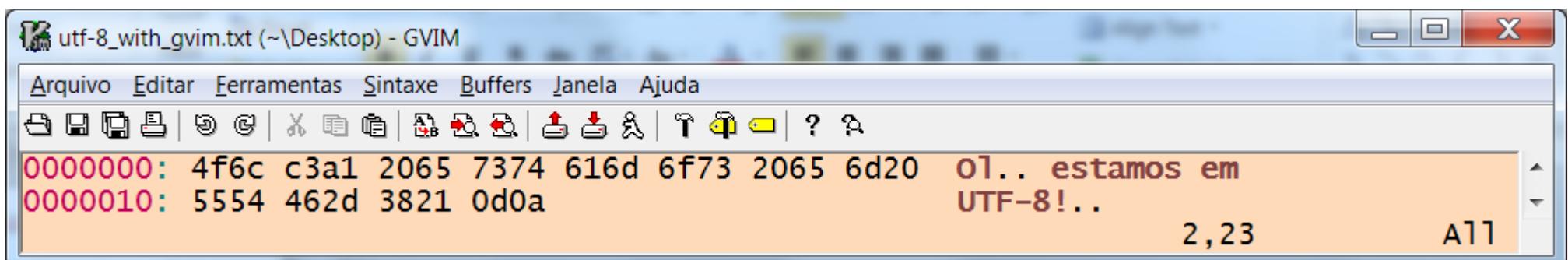
# UTF-8 (3)

- Vantagens
  - Uma string em ASCII (7 bits) é UTF-8 válido
  - Empregue em páginas HTML
    - <meta charset="UTF-8" />
  - Insensível à ordenação de bytes (*endianess*)
    - UTF-8: sequência de bytes (e não um valor *multibyte*)
    - Exemplo: EE.84.87 (emoji “o grito”) 
- Desvantagens
  - ASCII estendido (e.g., Ç, ã, ...) requer 2 bytes
  - Para certas línguas asiáticas (CJK), o UTF-8 pode requerer 3 bytes em vez de 2 bytes empregues para outras representações
- UTF-8 predomina na Web
  - ~90% do conteúdo da web



# UTF-8 (4)

- Exemplo
  - String: **Olá estamos em UTF-8!**
  - Visualização em hexadecimal
    - 4f = O
    - 6c = l
    - c3a1 = á
    - 20=espaço (32 decimal)
    - 65 = e
    - ...
- Uso do UTF-8 no vim
  - Modo de última linha
  - :set encoding=utf-8
  - :set fileencoding=utf-8
- Para visualizar em hexadecimal
  - :%!xxd



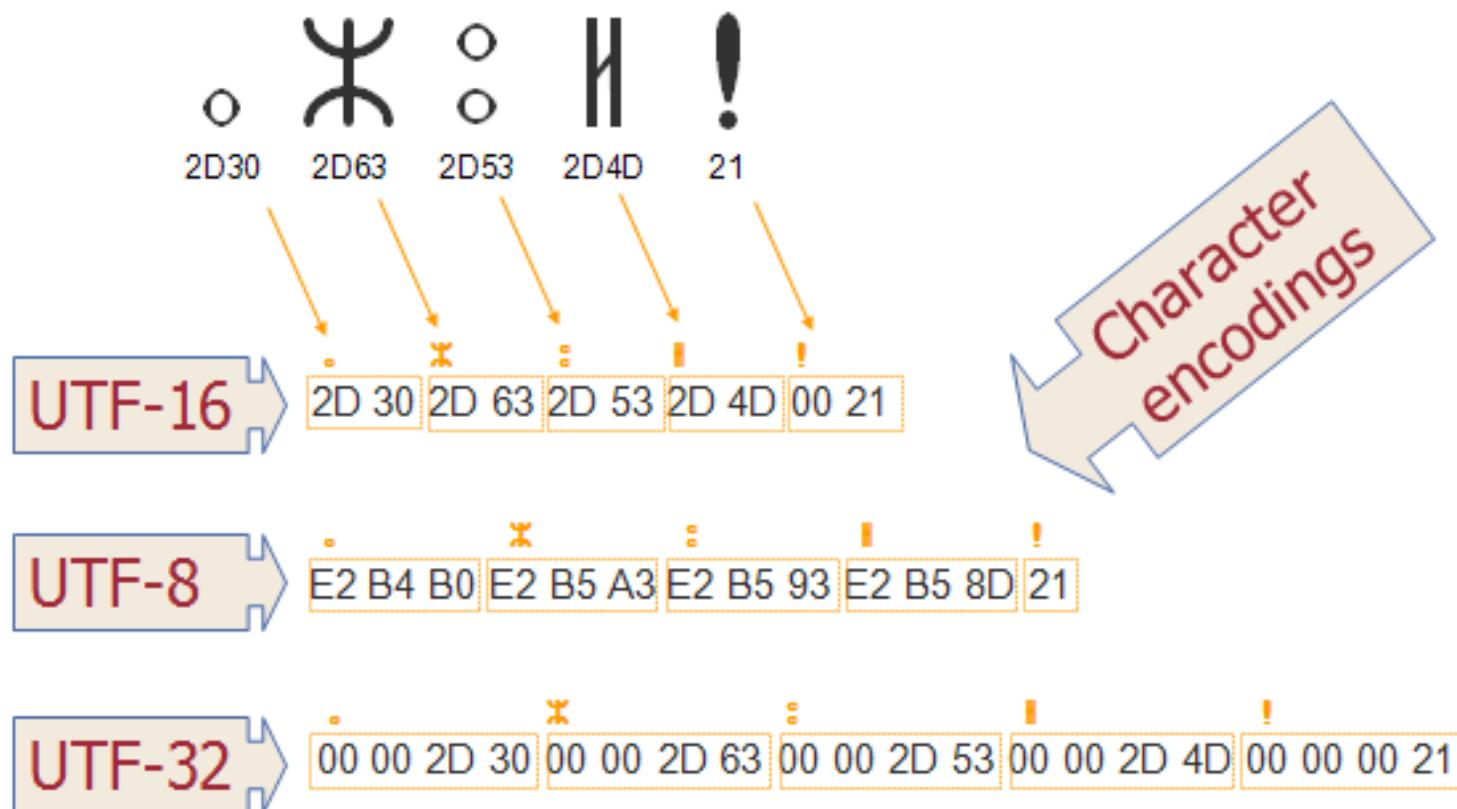
- NOTA - conversor UTF-8 online
  - <https://mothereff.in/utf-8>
- Conversor vários formatos
  - <https://r12a.github.io/apps/conversion/>

# UTF32 vs. UTF16 vs. UTF8

- Exemplo

- Fonte:

<http://www.w3.org/International/articles/definitions-characters/images/encodings.png>



# Unicode /utf-x

- Exemplo

## White Smiling Face U+263A

Unicode number **U+263A**HTML-code **&#9786;**Block [Miscellaneous Symbols](#)Set [Special symbols, Emoji](#)

### Encoding

Encoding	hex	dec (bytes)	dec	binary
UTF-8	E2 98 BA	226 152 186	14850234	11100010 10011000 10111010
UTF-16BE	26 3A	38 58	9786	00100110 00111010
UTF-16LE	3A 26	58 38	14886	00111010 00100110
UTF-32BE	00 00 26 3A	0 0 38 58	9786	00000000 00000000 00100110 00111010
UTF-32LE	3A 26 00 00	58 38 0 0	975568896	00111010 00100110 00000000 00000000

# Exemplo de codificação

- Exemplo em python 3

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
# use python 3
for codec in ['latin_1', 'utf_8', 'utf_16', 'utf_32']:
    print(codec, 'Avançada'.encode(codec))
```



\x: significa que os dois caracteres seguintes são um byte em formato hexadecimal

# Conversor iconv

- Utilitário linha de comando iconv

- Disponível em vários SO

- Uso

```
iconv -f origem -t destino in -o out
```

- Converter *latin1* para UTF8

```
iconv -f latin1 -t UTF8 in -o out
```

- Converter *latin1* para UTF16 (com *BOM*)

```
iconv -f latin1 -t UTF16 in -o out
```

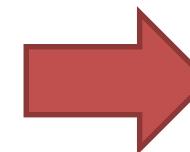
- Converter UTF8 para UTF16-LittleEndian (sem *BOM*)

```
iconv -f UTF8 -t UTF16-LE in -o out
```

iconv -l → Lista tabela de formatos suportados

# Unicode e Punycode

- O DNS suporta nomes com simbologia UNICODE
- Exemplo, Top Level Domains
- Associado a International Domain Name (IDN)
  - Caracteres UNICODE não representáveis por ASCII (exemplos: Ѽ ئ ئ ئ ئ ئ ئ )
  - Prefixo XN--
    - ASCII Compatible Encoding (ACE prefix)
    - Indica que o que segue está em codificação Punycode
      - Representar UNICODE com ASCII LDH (Letters+Digits+Hyphen)
  - RFC3490 (IDNA), RFC 3492 (Punycode)



XN--11B4C3D  
XN--1CK2E1B  
XN--1QQW23A  
XN--2SCRJ9C  
XN--30RR7Y  
XN--3BST00M  
XN--3DS443G  
XN--3E0B707E  
XN--3HCRJ9C  
XN--3OQ18VL8PN36A  
XN--3PXU8K  
XN--42C2D9A  
XN--45BR5CYL  
XN--45BRJ9C  
XN--45Q11C  
XN--4GBRIM  
XN--54B7FTA0CC  
XN--55QW42G  
XN--55QX5D  
XN--5SU34J936BGSG  
XN--5TZM5G  
XN--6FRZ82G  
XN--6QQ986B3XL  
XN--80ADXHKS  
XN--80AO21A  
XN--80AQECDR1A  
XN--80ASEHDB  
XN--80ASWG  
XN--8Y0A063A  
XN--90A3AC  
XN--90AE  
XN--90AIS  
XN--9DBQ2A  
XN--9ET52U  
XN--9KRT00A  
XN--B4W605FERD  
XN--BCK1B9A5DRE4C  
XN--C1AVG  
XN--C2BR7G  
XN--CCK2B3B

# Abuso de URL (Unicode)

## WhatsApp: How the supermarket voucher scam works

© 7 November 2017 | UK



Scammers have used WhatsApp to trick people into handing over personal information by tempting them with bogus supermarket vouchers.

The messenger app was used to send fake vouchers to people, purporting to be from trusted chains such as Asda, Tesco and Aldi.

The messages claimed to offer hundreds of pounds in savings so long as the user followed a link to an online survey asking for personal details.

The scam is a form of phishing, where fraudsters pose as reputable organisations to gain personal details.

For example, in the screenshot above, the d in Aldi is actually a ð - a Latin character with a small dot underneath the recognisable letter.

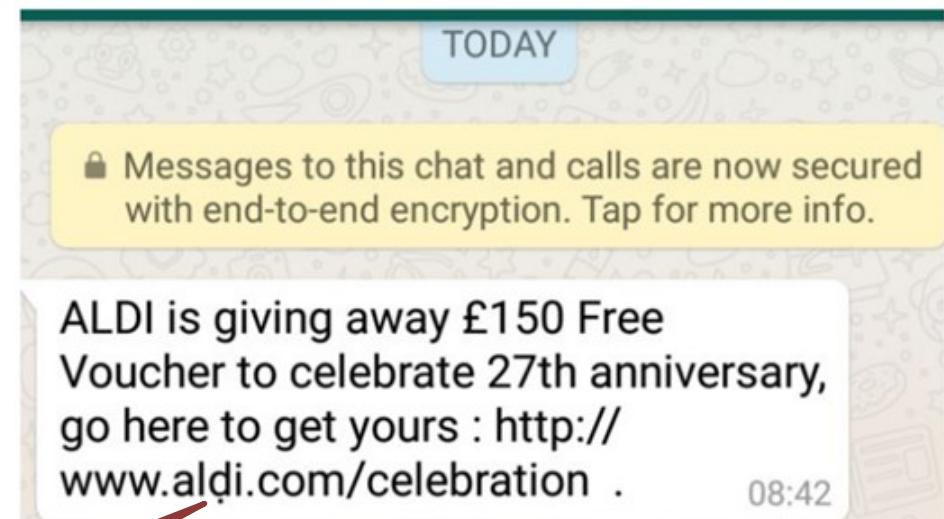
<http://www.bbc.com/news/uk-41900814>

*"d with dot below"* ð

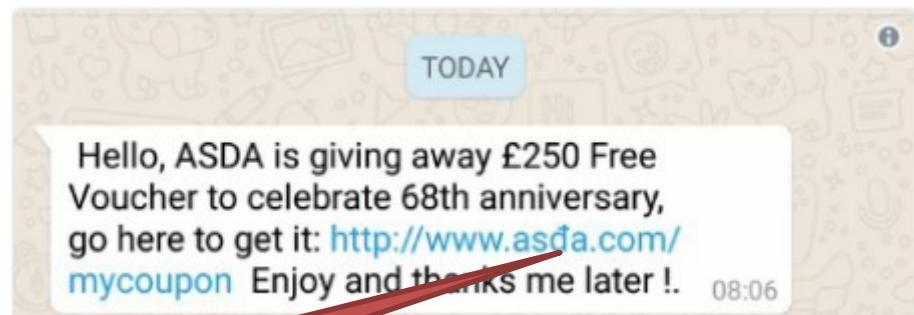
Unicode: U+1E0D

Não é d, mas *"d with stroke"* ð

Unicode: U+0111



*Phising com URL unicode*

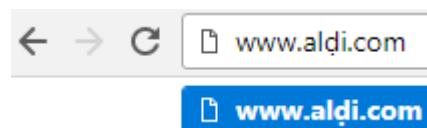


## ■ Navegador Vivaldi



- Substituição do ¸
- [www.aldi.com](http://www.aldi.com)
- Mostrado como:  
<http://www.xn--ali-wyy.com>

## ■ Navegador Chrome



- Não faz substituição...

## ■ Navegador firefox



- Não faz substituição
- Configurar: i) about:config; ii) procurar punycode; iii) alterar variável “network.IDN\_show\_punycode” para True

(c) Patrício Domingues

Preference Name	Status	Type	Value
network.IDN_show_punycode	modified	boolean	true

# Unicode slide show

## ■ Itera pelos símbolos do Unicode

### Unicode 10.0 Slide Show

[136,690 characters in 274 blocks covering 139 scripts]

[Start](#) [Pause](#) [Skip Block](#) [Random Character](#)

đ

Plane 0 : Basic Multilingual Plane (BMP)

Block 2 : Latin Extended-A

**U+0111 : LATIN SMALL LETTER D WITH STROKE**

First Block

0 : Basic Latin

Last Block

274 : Variation Selectors Supplement

Speed

10 characters a second

Font Size

72 points

Font Colour

Black

Background Colour

White

Random Characters

Use Custom Fonts

[Go To U+](#)

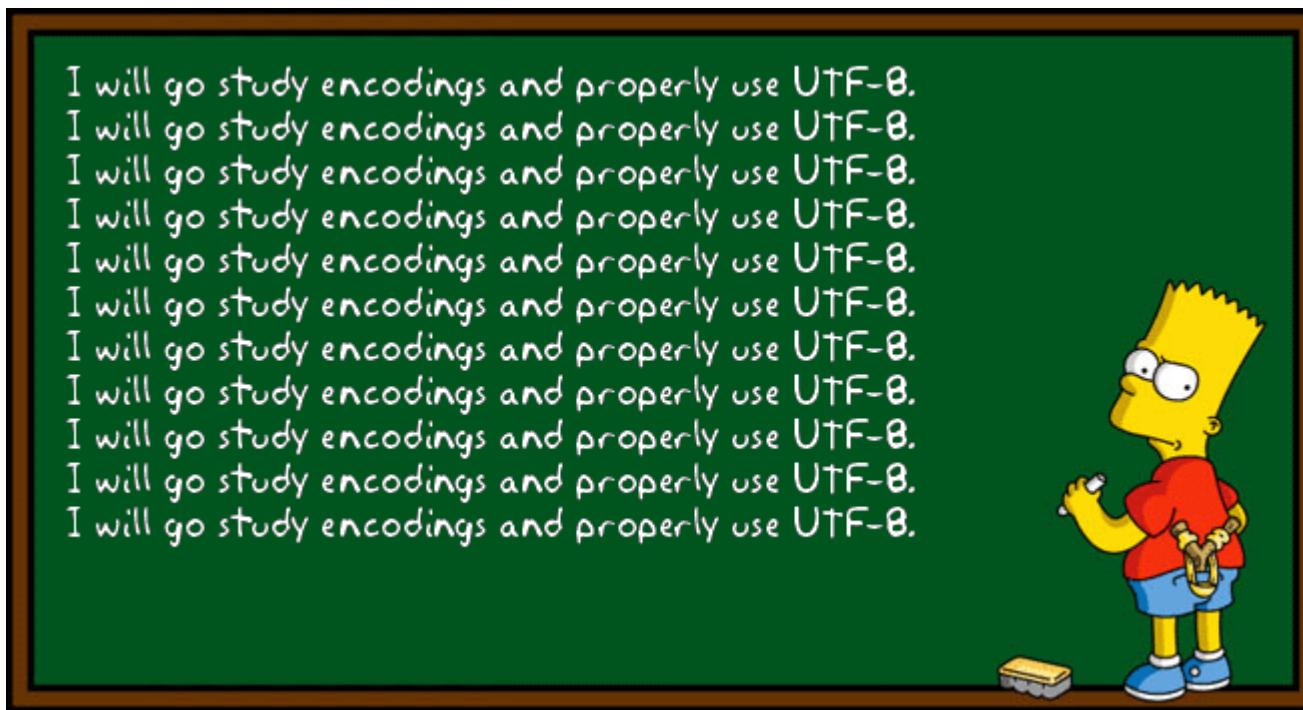
0111

[Search For](#)

Exact Name



<http://www.babelstone.co.uk/Unicode/unicode.html>



<http://perltricks.com/images/building-a-utf8-encoder-in-perl/bart-simpson-utf8.gif>

# Suporte para *multi-charset*

- Mas não é só Unicode que causa problemas...
- Strings que podem ser problemáticas
  - Undefined, undef
  - Null, NULL, (null)
  - Nil, NIL
  - true, false, True, False
  - TRUE, FALSE
  - None
  - hasOwnProperty
  - then
  - \
  - \\



Adam   
@adamjogrady

Em resposta a @SwiftOnSecurity

You don't even need Unicode, the apostrophe in O'Grady, from the basic ASCII charset still breaks government systems and banks to this day. 😊

Traduzir Tweet

10:59 - 4 de mai de 2018

29 Retweets 144 Curtidas



Adam @adamjogrady · 4 de mai

Em resposta a @adamjogrady @SwiftOnSecurity

True story, first day as a programmer for a government department my task was to fix our systems to handle my last name because they broke.

Traduzir Tweet



2



12



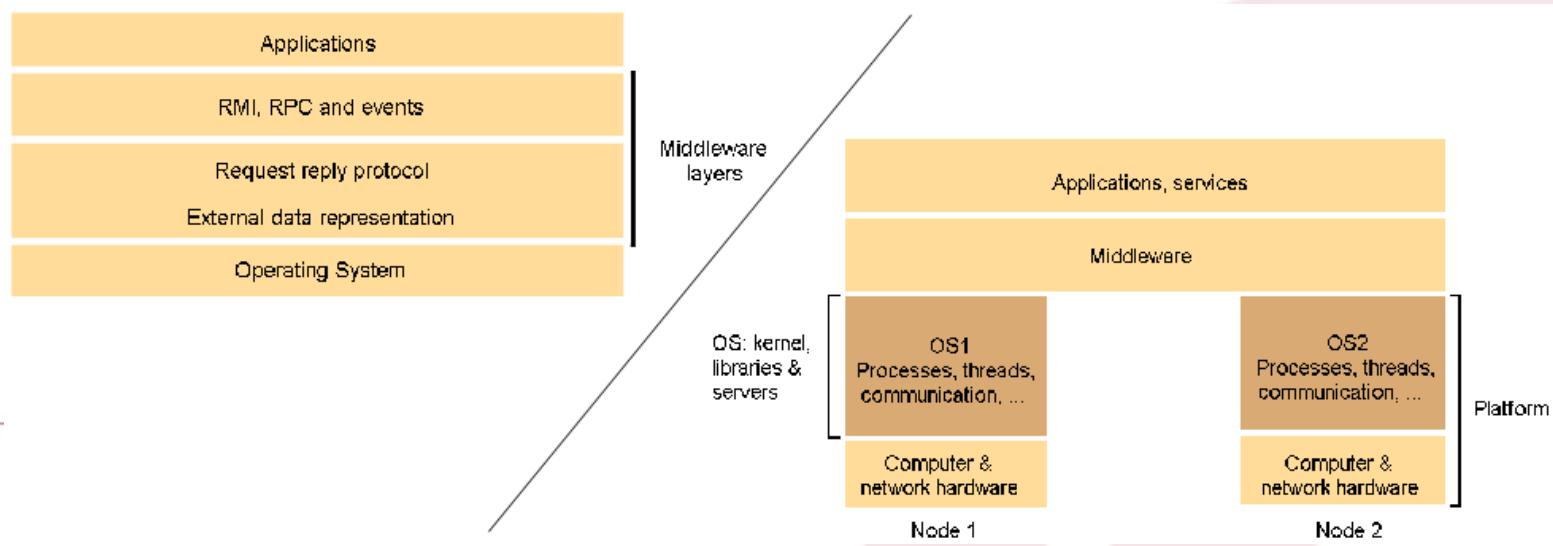
78



- +info: <https://github.com/minimaxir/big-list-of-naughty-strings/blob/master/blns.txt>

# Middleware (1)

- Noção de middleware
  - Interface entre as aplicações e o sistema operativo
  - Camada de abstração dos detalhes de representação dos dados e do processo de comunicação
  - Objetivos
    - Eliminar a heterogeneidade num ambiente distribuído
    - Facilitar a implementação e portabilidade de aplicações

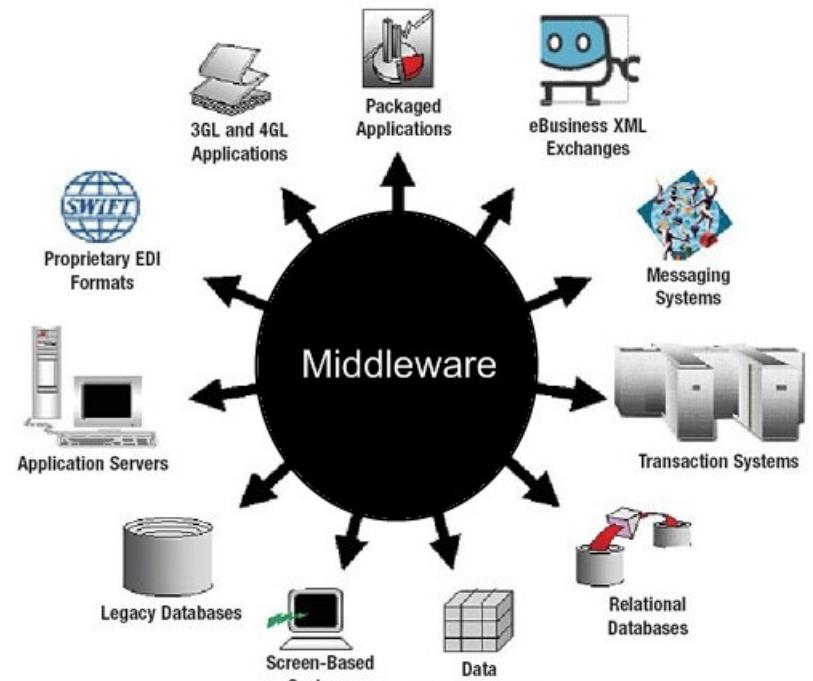


# Middleware (2)

- Mecanismos que possibilitam a comunicação entre aplicações e outras entidades de software

- Exemplos

- Aplicação a comunicar com servidor de bases de dados
  - *Middleware*: ODBC, JDBC
  - O middleware pode abstrair o servidor de bases de dados
  - Possibilita que se troque de servidor (MySQL por Oracle) sem ser necessário alterar a aplicação (na teoria!)
- Motores de jogos
  - Unreal, Unity
- Oracle fusion middleware
  - Software para integração entre aplicações e serviços



<http://bpmtalk.wordpress.com/2009/10/13/an-introduction-to-middleware/>

- XML = eXtensible Markup Language
  - Representação de dados/informação de forma independente da plataforma
  - Uso de texto (formato unicode)
    - <?xml version="1.0" encoding="UTF-8" ?>
  - Assente numa estrutura bem definida
  - Origens: HTML + SGML
    - XML é SGML simplificado
  - Definido pelo W3C através do standard XML 1.0 (já existe XML 1.1)
- Empregues em muitos cenários
  - Formato OpenDocument (open office), Office Open XML (Microsoft's office), iWork (Apple)
- Web services, disponibilização dados online
  - RSS, ATOM, ...

- O XML é uma “meta” linguagem com a qual é possível definir outras linguagens através de *schemas XSD* (substituto dos “DTDs”)
  - XML = HTML - tags apresentação + schema
  - XML schema
    - O schema é definido pelo utilizador/criador e contém as “tags” e hierarquia da informação válidas para a informação que se pretende representar
    - O schema pode (e deve) ser gerado recorrendo ao auxílio de ferramentas – o *schema* usa linguagem XML

# XML vs HTML (1)

- O HTML é uma linguagem de formatação
  - Indica como deve ser mostrado
  - Não estrutura os dados
    - Extrair dados de uma página HTML é um processo casuístico, não estruturado
- O XML é uma linguagem para representação de dados
  - Os tipos de dados são definidos, bem como os seus relacionamentos



**Exemplos >>**

# XML vs HTML (2)

## HTML

- Tags HTML

- Apresentação e estrutura do documento

**<h1>**Unidades curriculares**</h1>**

**<p><i>**Programação Avançada**</i>**

**<br>** Prof. Responsável PA

**<br>** EI - 2º Ano, 1º Semestre**</p>**

**<p><i>**Sistemas Operativos**</i>**

**<br>** Prof. Responsável SO

**<br>** EI - 1º Ano, 2º Semestre**</p>**



## XML

- Tags XML

- conteúdo, "semântica", associadas a uma estrutura (DTD ou Schema)

**<UnidadeCurricular>**

**<nome>**Programação Avançada**</nome>**

**<responsavel>** Prof. PA **</responsavel>**

**<ano>** 2º Ano **</ano>**

**<semestre>** 1º**</semestre>**

**</UnidadeCurricular>**

**<UnidadeCurricular>**

**<nome>**Sistema Operativos**</nome>**

(...)

**<ano>** 1º Ano **</ano>**

**</UnidadeCurricular>**



# XML – Representação de dados binários

- Sendo o XML uma linguagem “textual”, como passar dados binários (imagens, vídeos, etc)?
  - Usar um algoritmo de codificação de dados binários para texto (ex. Base64)
  - Os símbolos usados na codificação são apenas símbolos ASCII básicos
- Base64
  - Representação de dados binários recorrendo a uma alfabeto de 64 símbolos
    - 64 símbolos = 6 bits
    - Dados originais são agrupados em blocos de 6 bits, sendo esse bloco substituído por um símbolo ASCII definido por uma tabela
    - O binário **000000** é substituído por A, o binário **000001** é substituído por B, O binário **000010** é substituído por C, etc.

# XML – Representação de dados binários

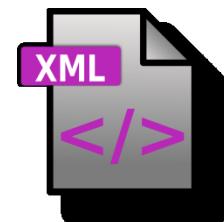
- Sendo o XML uma linguagem “textual”, como passar dados binários (imagens, vídeos, etc)?
  - Usar um algoritmo de codificação de dados binários para texto (ex. Base64)
- **Base64**
  - Empregue para o transporte/armazenamento de dados binários em sistemas que usam ASCII
    - Email (com uso de Multipurpose Internet Mail Extensions – MIME)
  - Representação de dados binários recorrendo a uma alfabeto de 64 símbolos
    - 64 símbolos = 6 bits
    - Dados originais são agrupados em blocos de 6 bits, sendo esse bloco substituído por um símbolo ASCII definido por uma tabela
    - O binário **000000** é substituído por A, o binário **000001** é substituído por B, O binário **000010** é substituído por C, etc.
    - Exemplo (fonte: <http://en.wikipedia.org/wiki/Base64>)



Text content	M	a	n
ASCII	77	97	110
Bit pattern	0 1 0 0 1 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 1 1 1 0		
Index	19	22	5
Base64-encoded	T	W	F
			u

# Base64 (continuação)

- A representação em base64 requer mais bits que a representação nativa
  - 6 bits nativos são mapeados em 8 bits base64
    - Sobrecarga de  $8/6 = 1,33 \rightarrow 33\%$
  - Exemplo
    - Representação de imagem em base64 xml.png
    - `data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAGkAAABtCAIAAAAAtcdh7AAAAAXNSR0IA (...) ASUVORK5CYII`
    - `xml.png` = 7256 bytes, Base64 = 9699 bytes
    - Overhead:  $9699/7256 = 1.3367 \rightarrow 33\%$
    - Sítio para conversão: <http://webcodertools.com/imagetobase64converter/Create>



- XML apenas representa os dados
- Onde está a representação gráfica?
  - Dado o âmbito do XML não ser somente a “Web” foi criada a linguagem XSLT baseada em XML que permite transformar XML em qualquer formato (incluindo HTML)
- XSLT = eXtensible Stylesheet Language for Transformations
  - Permite transformar a estrutura da informação de um documento XML noutra estrutura
  - Exemplos
    - Transformar XML em HTML
    - Transformar XML numa query SQL

# XSLT – exemplo (1)

- XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="dei.xsl"?>
<dei>

    <unidadecurricular>
        <nome>Programação Avançada</nome>
        <professor> Prof. responsável PA</professor>
        <ano> 2º Ano </ano>
        <semestre> 1º</semestre>
    </unidadecurricular>
    <unidadecurricular>
        <nome> Sistema Operativos </nome>
        <professor> Prof. responsável SO</professor>
        <semestre> 2º</semestre>
        <ano> 1º Ano </ano>
    </unidadecurricular>
</dei>
```

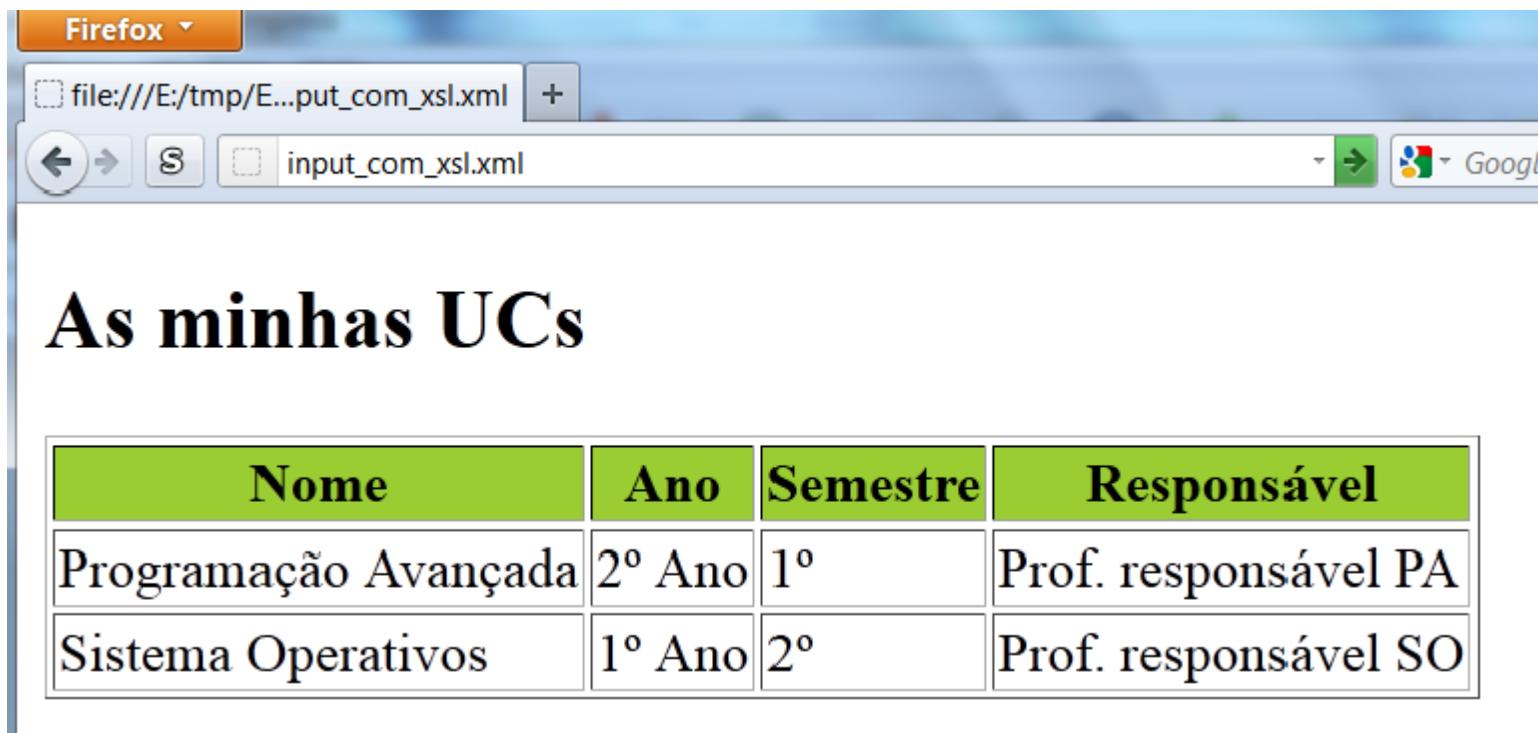
# XSLT – exemplo (2)

- Código XSLT (para formatação do XML anteriormente mostrado)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="">
<xsl:template match="/">
  <html>
    <body>
      <h2>As minhas UCs</h2>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th>Nome</th>
          <th>Ano</th>
          <th>Semestre</th>
          <th>Responsável</th>
        </tr>
        <xsl:for-each select="dei/unidadecurricular">
          <tr>
            <td><xsl:value-of select="nome"/></td>
            <td><xsl:value-of select="ano"/></td>
            <td><xsl:value-of select="semestre"/></td>
            <td><xsl:value-of select="professor"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

# XSLT – exemplo (3)

- Resultado



**As minhas UCs**

Nome	Ano	Semestre	Responsável
Programação Avançada	2º Ano	1º	Prof. responsável PA
Sistema Operativos	1º Ano	2º	Prof. responsável SO

- Existem protocolos distribuídos cuja representação de informação assenta em XML
  - XML-RPC – XML Remote Procedure Call
    - Especificação para plataforma de procedimentos remotos
  - SOAP – Simple Object Access Protocol
    - Empregue na plataforma .NET da Microsoft
    - Versão simplificada do XML-RPC
  - Comunicação entre máquinas: Machine 2 Machine (M2M)
    - Especialmente na IoT
  - Sítios de comércio eletrónico: Business to Business (B2B)
- Outros usos
  - Aplicações
    - libreoffice, microsoft office: Open Document Format (ODF)
  - Programas científicos
    - dados baseados em XML, de modo a tirarem partido da portabilidade

- Alguns utilitários disponibilizam saída em formato XML
  - Exemplo: WEVTUtil
    - Utilitário para acesso, configuração e exportação do sistema de log no Windows
    - `wevtutil gl system /f:XML`
      - Produz saída XML com a indicação da configuração do log *system*

```
c:\>wevtutil gl system /f:xml
<?xml version="1.0" encoding="UTF-8"?>
<channel name="system" enabled="true" type="Admin" owningPublisher="" isolation="System" channelAccess="0:BAG:SYD:(A;;0x
f0007;;;SY)(A;;0x7;;;BA)(A;;0x3;;;B0)(A;;0x5;;;S0)(A;;0x1;;;IU)(A;;0x3;;;SU)(A;;0x1;;;S-1-5-3)(A;;0x2;;;S-1-5-33)(A;;0x1
;;;S-1-5-32-573)" xmlns="http://schemas.microsoft.com/win/2004/08/events">
  <logging>
    <logFileName>%SystemRoot%\System32\Winevt\Logs\system.evtx</logFileName>
    <retention>false</retention>
    <autoBackup>false</autoBackup>
    <maxSize>20971520</maxSize>
  </logging>
  <publishing>
    <fileMax>1</fileMax>
  </publishing>
</channel>
```

# XML - problemas

- Demasiadamente verboso
  - Rácio conteúdo / suporte XML é baixo
  - Pouco legíveis para o operador humano
  - Exemplo
    - Cor azul: sobrecarga do XML

```
<UnidadeCurricular>
  <nome>Programação Avançada</nome>
  <responsavel> Prof. PA </responsavel>
  <ano> 2º Ano </ano>
  <semestre> 1º</semestre>
</UnidadeCurricular>
```

- Computacionalmente oneroso
  - Processamento pesado (validação)

- JSON – JavaScript Object Notation
  - Norma aberta assente na representação em modo texto de informação
    - RFC 4627 / Julho 2006 (<http://tools.ietf.org/html/rfc4627>)
  - Empregue pela linguagem JavaScript para a representação estruturas de dados
  - O formato JSON é empregue para a serialização/transmissão entre servidor e aplicação web
  - Alternativa ao XML
    - Mais simples, menos “verboso”
  - Empregues por muitas API web
    - Twitter, youtube, google maps, ...

# JSON exemplo (#1)

Serviço GeolP, disponível via http em **ipinfo.io/IP**

- substituir IP por um endereço IP
- Exemplo:
  - 194.210.216.8 – um dos endereços IP do IPLeiria.pt
  - wget <http://ipinfo.io/194.210.216.8>

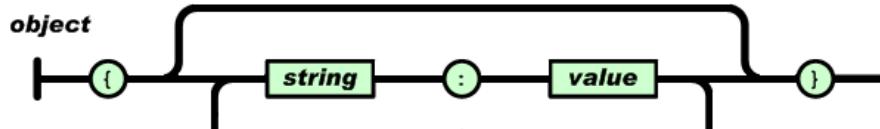
```
{  
  "ip": "194.210.216.8",  
  "city": "Moura",  
  "region": "Leiria",  
  "country": "PT",  
  "loc": "39.7196,-8.8155",  
  "org": "AS1930 Fundacao para a Ciencia e a Tecnologia, I.P.",  
  "postal": "7860mour"  
}
```

Notas:

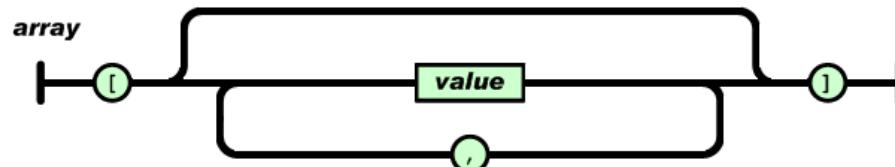
- String em JSON usam "" (aspas)
- Objeto é delimitado por { e }

# JSON - elementos

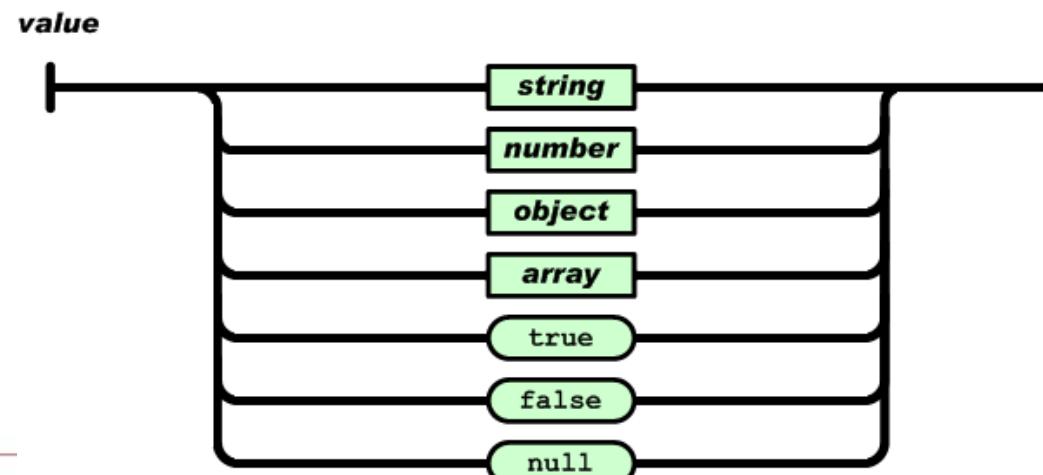
- Object { ... }



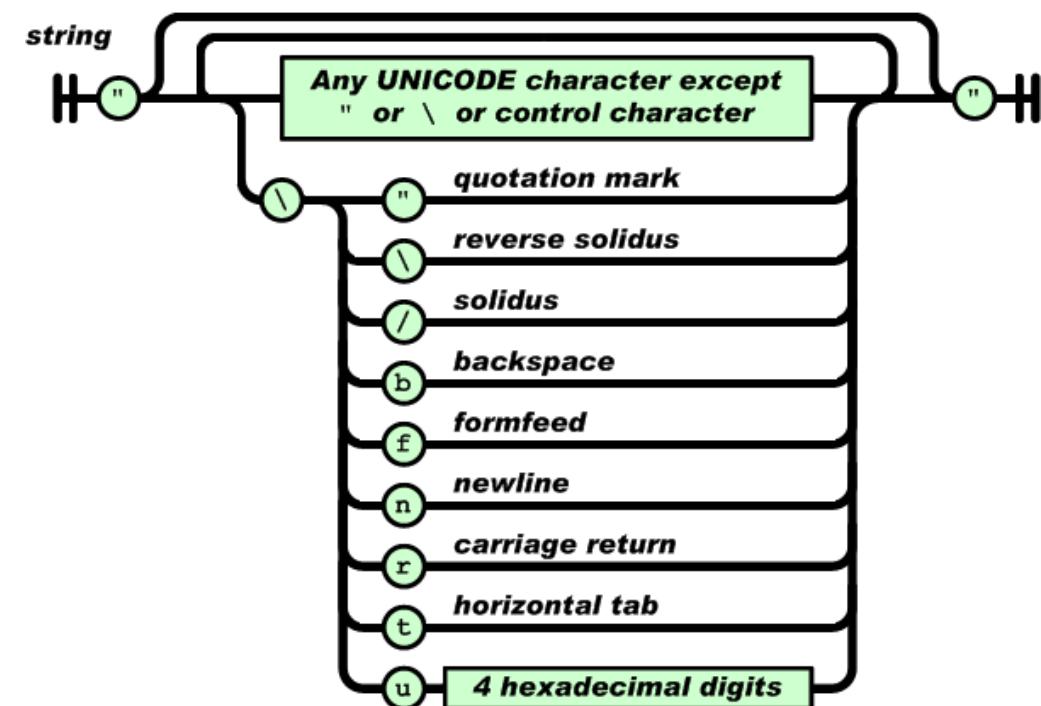
- Array [ ... ]



- Value



- String



- Fonte: <http://json.org>

Mais exemplos JSON >>

# JSON exemplo (#2)

- Exemplo multinível

```
{"widget": {  
    "debug": "on",  
    "window": {  
        "title": "Sample Konfabulator Widget",  
        "name": "main_window",  
        "width": 500,  
        "height": 500  
    },  
    "image": {  
        "src": "Images/Sun.png",  
        "name": "sun1",  
        "hOffset": 250,  
        "vOffset": 250,  
        "alignment": "center"  
    },  
    "text": {  
        "data": "Click Here",  
        "size": 36,  
        "style": "bold",  
        "name": "text1",  
        "hOffset": 250,  
        "vOffset": 100,  
        "alignment": "center",  
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"  
    }  
}}
```

# JSON exemplo (#3)

- Exemplo google maps (fonte: <https://www.sitepoint.com/google-maps-json-file/>)

```
{  
  "markers": [  
    {  
      "name": "Rixos The Palm Dubai",  
      "position": [25.1212, 55.1535],  
    },  
    {  
      "name": "Shangri-La Hotel",  
      "location": [25.2084, 55.2719]  
    },  
    {  
      "name": "Grand Hyatt",  
      "location": [25.2285, 55.3273]  
    }  
}
```

# JSON – exemplo

- ExifTool
  - Aplicação que acede aos metadados específicos de certos tipos de ficheiros
    - Imagens (EXIF)
    - PDF
    - Ficheiros office
    - Etc.
  - <https://sno.phy.queensu.ca/~phil/exiftool/>
- Opção para disponibilizar dados em formato JSON
  - -j
- exifTool -j ficheiro
- Exemplo

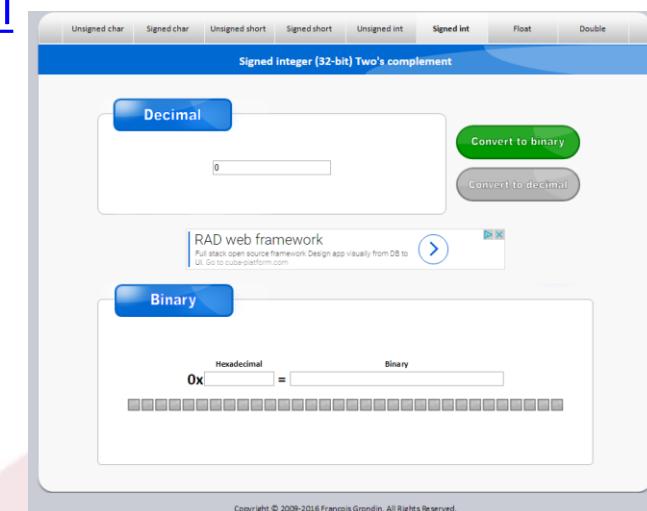
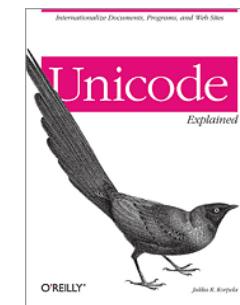
```
exiftool -j ficheiro.pdf
```

```
[{"SourceFile": "Writing endian-independent code in C.pdf", "ExifToolVersion": 10.65, "FileName": "Writing endian-independent code in C.pdf", "Directory": ".", "FileSize": "61 kB", "FileModifyDate": "2007:12:17 19:03:20+00:00", "FileAccessDate": "2017:09:11 16:14:10+01:00", "FileCreateDate": "2017:09:11 16:14:08+01:00", "FilePermissions": "rw-rw-rw-", "FileType": "PDF", "FileTypeExtension": "pdf", "MIMEType": "application/pdf", "PDFVersion": 1.4, "Linearized": "No", "PageCount": 8, "Producer": "GPL Ghostscript 8.54", "CreateDate": "2007:12:17 18:02:22+00:00", "ModifyDate": "2007:12:17 18:02:22", "Title": "Writing endian-independent code in C", "Creator": "PDFCreator Version 0.9.3", "Author": "Owner", "Subject": ""}]
```

# Bibliografia



- Korpela, J. (2006). *Unicode explained.*" O'Reilly Media, Inc.", ISBN: 978-0596101213
- Recursos online
  - Conversor binário / decimal / tipos de dados
    - [http://www.binaryconvert.com/convert\\_signed\\_int.html](http://www.binaryconvert.com/convert_signed_int.html)
- ✓ XML
  - ✓ <http://www.xml.com/>
  - ✓ <http://www.w3.org/XML/>
- ✓ JSON
  - ✓ <http://tools.ietf.org/html/rfc4627>
- Erard M.(2017.10.18), “How the Appetite for Emojis Complicates the Effort to Standardize the World’s Alphabets”, NYTimes, (<http://nyti.ms/2yJfiCg>)



The screenshot shows a web-based binary converter. At the top, there are tabs for Unsigned char, Signed char, Unsigned short, Signed short, Unsigned int, Signed int, Float, and Double. Below these, under the "Signed integer (32-bit) Two's complement" tab, there are two main input fields: "Decimal" (containing "0") and "Binary" (containing "0x"). Between them is a "Convert to binary" button. Below these fields, there is a note about RAD web framework and a copyright notice at the bottom: "Copyright © 2009-2016 François Grondin. All Rights Reserved."

