

# Unicode

---

Série Unicode
Unicode
UCS
UTF-7
UTF-8
UTF-16
UTF-32/UCS-4
SCSU
Punycode

**Unicode** é um padrão que permite aos computadores representar e manipular, de forma consistente, texto de qualquer sistema de escrita existente. Publicado no livro *The Unicode Standard*<sup>[1], [2]</sup> o padrão consiste de pouco mais de 107 mil caracteres<sup>[3]</sup>, um conjunto de diagramas de códigos para referência visual, uma metodologia para codificação e um conjunto de codificações padrões de caracteres, uma enumeração de propriedades de caracteres como caixa alta e caixa baixa, um conjunto de arquivos de computador com dados de referência, além de regras para normalização, decomposição, ordenação alfabética e renderização.

Atualmente, é promovido e desenvolvido pela *Unicode Consortium*, uma organização sem fins lucrativos que coordena o padrão, e que possui o objetivo de um dia substituir esquemas de codificação de caractere existentes pelo Unicode e pelos esquemas padronizados de transformação Unicode (chamado *Unicode Transformation Format*, ou UTF). Seu desenvolvimento é feito em conjunto com a Organização Internacional para Padronização (ISO) e compartilha o repertório de caracteres com o ISO/IEC 10646: o Conjunto Universal de Caracteres (UCS). Ambos funcionam equivalentemente como codificadores de caracteres, mas o padrão Unicode fornece muito mais informação para implementadores, cobrindo em detalhes tópicos como ordenação alfabética e visualização.

Seu sucesso em unificar conjuntos de caracteres levou a um uso amplo e predominante na internacionalização e localização de programas de computador. O padrão foi implementado em várias tecnologias recentes, incluindo XML, Java e sistemas operacionais modernos.

## Visão geral

### Histórico de revisões

Data de lançamento	Versão	ISBN	Sistemas de escrita	Códigos
outubro de 1991	1.0	ISBN 0-201-56788-1	24	7.161
junho de 1992	1.0.1	ISBN 0-201-60845-6	25	28.359
junho de 1993	1.1		24	34.233
julho de 1996	2.0	ISBN 0-201-48345-9	25	38.950
maio de 1998	2.1		25	38.952
setembro de 1999	3.0	ISBN 0-201-61633-5	38	49.259
março de 2001	3.1		41	94.205
março de 2002	3.2		45	95.221
abril de 2003	4.0	ISBN 0-321-18578-1	52	96.447

março de 2005	4.1		59	97.720
julho de 2006	5.0	ISBN 0321480910	64	99.089
abril de 2008	5.1		75	100.713
outubro de 2009	5.2		90	107.361

O Unicode possui o objetivo explícito de transcender as limitações de codificações de carácter tradicionais, como as definidas pelo padrão ISO 8859, que possuem grande uso em vários países mas que permanecem em sua maioria incompatíveis umas com as outras. Várias codificações de carácter tradicionais compartilham um problema comum, ao permitirem processamento bilíngue (geralmente usando caracteres romanos e a língua local), mas não processamento multilíngue (processamento de línguas arbitrárias misturadas umas com as outras).

O Unicode codifica os caracteres em si - grafemas e unidades tais como grafemas - em vez de codificar glifos variantes para tais caracteres. No caso de caracteres chineses, essa estratégia geralmente leva a controvérsias quanto à distinção entre um caractere e seus glifos variantes.

Na área de processamento de texto, o Unicode possui o papel de fornecer um único código - um número e não um glifo - para cada carácter. Em outras palavras, o Unicode representa um carácter em uma forma abstrata e deixa questões sobre renderização (como tamanho, forma, fonte ou estilo) para outro software, como um navegador ou um editor de texto. Esse simples objetivo torna-se complicado pelas concessões feitas pelos desenvolvedores do padrão a fim de encorajar uma adoção mais rápida.

Os 256 primeiros códigos Unicode são idênticos aos do padrão ISO 8859-1, de forma que é trivial converter texto ocidental existente. Diversos caracteres idênticos foram codificados múltiplas vezes em diferentes códigos para preservar distinções usadas por codificações legadas, permitindo assim a conversão de tais codificações para Unicode e vice versa, sem perder qualquer informação. Da mesma forma, enquanto o Unicode permite combinar caracteres, ele também codifica versões pré-compostas da maioria das combinações mais comuns de letra/diacrítico. Por exemplo, o carácter "é" pode ser representado por U+0065 (letra latina "e" minúsculo) combinado com U+0301 (diacrítico "acento agudo"), mas também pode ser representado como U+00E9 (letra latina "e" com diacrítico "acento agudo").

O padrão ainda inclui outros itens relacionados, como propriedades de caracteres, formas de normalização de texto e ordem bidirecional de visualização (para a correta visualização de texto lido da direita à esquerda, como em língua árabe ou hebraica).

Quando se escreve sobre um carácter Unicode, normalmente se usa "U+" seguido de um número hexadecimal que indica o código do carácter.

## Origem e desenvolvimento

Entre 1986 e 1987, iniciou-se na Xerox o trabalho de construção dum banco de dados para mapear o relacionamento entre caracteres idênticos dos alfabetos japonês, chinês tradicional e chinês simplificado, a fim de construir uma fonte tipográfica para caracteres chineses estendidos. O grupo de funcionários envolvidos incluía Huan-mei Liao, Nelson Ng, Dave Opstad e Lee Collins. Até então, os utilizadores da Xerox usavam JIS para estender o conjunto original de caracteres chineses. Na mesma época, na Apple se iniciou a discussão sobre um conjunto universal de caracteres. O grupo da Xerox começa uma discussão sobre questões multilíngues com Mark Davis, da Apple. Já em dezembro de 1987 é registrado o primeiro uso documentado do termo "Unicode".

A partir de 1988 começam discussões sobre uma largura fixa ou variável de bytes para a representação dos códigos, e uma das primeiras propostas é o sistema de Davis com uma largura fixa de 16 bits com o nome "High Text", em oposição a "Low Text" para o padrão ASCII. Nos estudos são levados em conta comparações entre o acesso de texto em largura fixa e variável, investigações sobre os requisitos para se utilizar 16 bits em sistemas computacionais e uma estimativa inicial de contagem de todos os caracteres existentes, para definir se 16 bits seriam mesmo o

suficiente.<sup>[4]</sup>

Em abril, os primeiros protótipos começaram a ser construídos na Apple, decidindo-se incorporar suporte ao padrão no TrueType, o padrão de fontes tipográficas da empresa.

Em janeiro de 1989 a Metaphor decide implementar uma codificação 16-bit para suportar internacionalização em seu conjunto de software. Nos meses seguintes, as freqüentes reuniões do grupo Unicode contam com a presença de representantes de empresas como Metaphor, Sun, Adobe, HP e NeXT. Tais reuniões evoluíram até o comitê técnico do Unicode, com a formação da *Unicode Consortium* dois anos após.

Em setembro, o grupo decide usar padrões ISO já existentes para ordenações de sistemas de escrita e nomeação de esquemas. No mês seguinte, o padrão é apresentado para a Microsoft e a IBM, em conjunto com a cooperação entre Apple e Microsoft com o TrueType. O padrão também foi apresentado ao grupo de internacionalização do Unix.

A partir do início de 1990, a Microsoft começa a participar das reuniões do Unicode. Em junho é a vez da IBM começar a participar mais ativamente. No mesmo ano é iniciado o trabalho para a formação de um consórcio ao padrão. Em 3 de janeiro de 1991 a *Unicode Consortium* é fundada, como *Unicode, Inc.* na Califórnia, Estados Unidos da América. No dia 25 é realizada a primeira reunião dos membros, e ainda em janeiro é formado o comitê técnico Unicode. No mês seguinte, um dos primeiros artigos sobre o Unicode aparece no New York Times<sup>[5]</sup>. Atualmente, qualquer empresa ou pessoa disposta a pagar os custos de associação pode se tornar membro da organização; os membros incluem, virtualmente, todas as principais empresas de software e hardware interessadas em padrões de processamento de texto, tais como Adobe Systems, Apple, Google, HP, IBM, Microsoft e Xerox. Outras instituições incluem a Universidade de Berkeley, o governo da Índia e o governo do Paquistão.<sup>[6]</sup>

## Sistemas de escrita suportados

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2D3	◌	⊖	⊕	⊗	⊘	⊙	⊚	⊛	⊜	⊝	⊞	⊟	⊠	⊡	⊢	⊣
2D4	⊤	⊥	⊦	⊧	⊨	⊩	⊪	⊫	⊬	⊭	⊮	⊯	⊰	⊱	⊲	⊳
2D5	⊴	⊵	⊶	⊷	⊸	⊹	⊺	⊻	⊼	⊽	⊾	⊿	⋀	⋁	⋂	⋃
2D6	⋄	⋅	⋆	⋇	⋈	⋉	⋊	⋋	⋌	⋍	⋎	⋏	⋐	⋑	⋒	⋓
2D7																

A codificação Unicode para tfinagh.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0B0		◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
0B1	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
0B2	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
0B3	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
0B4	1															
0B5																
0B6	◌	◌														
0B7	✓	◌														

A codificação Unicode para o alfabeto oriá.

O Unicode cobre quase todos os sistemas de escritas em uso atualmente, incluindo:

- Alfabeto árabe
- Alfabeto armênio
- Alfabeto bengali
- Braille
- Alfabeto cherokee
- Alfabeto copta
- Alfabeto cirílico
- Devanagari
- Alfabeto ge'ez
- Alfabeto georgiano
- Alfabeto grego
- Alfabeto gujarati
- Alfabeto gurmukhi
- Caracteres chineses
- Hangul
- Alfabeto hebraico
- Hiragana e Katakana
- Alfabeto fonético internacional (AFI)
- Alfabeto khmer
- Alfabeto kannada
- Alfabeto latino
- Alfabeto mongoliano
- Alfabeto birmanês
- N'Ko
- Alfabeto oriá
- Alfabeto siríaco
- Alfabeto tamil
- Alfabeto tailandês
- Alfabeto tibetano
- Tifinagh
- Alfabeto yi
- Bopomofo

O padrão adicionou outros sistemas de escrita e cobrirá mais, incluindo sistemas históricos ou extintos usados primordialmente pela academia, tais como:

- Escrita cuneiforme
- Ogham
- Runas
- Alfabeto deseret
- Alfabeto etrusco
- Alfabeto ugarítico
- Escrita Linear B
- Alfabeto fenício
- Élfico

## Desenvolvimento

Apesar de mais de trinta sistemas de escrita serem incluídos no Unicode, ainda restam outros por codificar. Novas adições de caracteres para sistemas já codificados também ocorrem, tal como símbolos matemáticos e musicais. Michael Everson, Rick McGowan e Ken Whistler mantêm uma lista de tais sistemas e um rascunho de codificação na página oficial do *Unicode Consortium*<sup>[7]</sup>. Para alguns sistemas já adicionados ao rascunho de desenvolvimento, propostas de codificação são feitas até que se encontre um consenso que leve à aprovação. Para outros sistemas, não são feitas propostas até que comunidades acadêmicas envolvidas aprove o repertório de caracteres e outros detalhes.

Entre os sistemas de escrita esperando codificação, estão os hieróglifos egípcios, fenícios, e maias, além do alfabeto babilônico e outros de caráter cuneiforme.

## Mapeamento e codificação

### Armazenamento, transferência e processamento

O armazenamento dos códigos Unicode no processamento de texto apresenta o desafio de a maioria dos programas de computador escritos no mundo ocidental utilizar somente codificações de 8 bits — 1 byte — (como o padrão ASCII), já que o suporte ao Unicode começou somente nos últimos anos. Similarmente, na representação de sistemas de escrita asiáticos, o modelo baseado em ASCII de caracteres de 2 bytes não pode mesmo em teoria codificar mais que 32 768<sup>[8]</sup> caracteres e, na prática, as arquiteturas impõem limites ainda menores. Tais limites são insuficientes mesmo só tendo em conta as necessidades de acadêmicos da língua chinesa.

A lógica interna de muitos programas legados tipicamente permite somente 8 bits para cada carácter, tornando impossível o uso de mais de 256 códigos<sup>[9]</sup> sem processamento especial. Programas 16-bit suportam somente dezenas de milhares de caracteres. Por outro lado, o Unicode já definiu pouco mais de 107 mil caracteres codificados. Desenvolvedores de sistemas já sugeriram diversos mecanismos para implementar o padrão; a escolha de cada um depende do espaço de armazenamento disponível, compatibilidade de código fonte e interoperabilidade com outros sistemas.

O Unicode define dois métodos de mapeamento de códigos Unicode em códigos de implementação, UTF (*Formato de Transformação Unicode*, do inglês *Unicode Transformation Format*) e UCS (*Conjunto Universal de Caracteres*, do inglês *Universal Character Set*). Os números associados aos nomes dos mapeamentos indicam o número de bits por código (no caso de UTF) ou o número de bytes por código (no caso de UCS). UTF-8 e UTF-16 são possivelmente as mais usadas.

A UTF-7 é uma codificação não tão popular usada para a codificação em 7 bits, e é normalmente considerada obsoleta. Pode ser usada quando há restrições a caracteres com o oitavo bit ligado; por exemplo, quando só se podem usar caracteres ASCII válidos. Não faz parte do padrão, sendo apresentada apenas como uma recomendação. Já a UTF-EBCDIC possui largura variável e maximiza a compatibilidade com EBCDIC, mas também não faz parte do padrão.

Por outro lado, a UTF-8 é uma codificação de muito usada, e que maximiza a compatibilidade com ASCII. Utiliza entre um e quatro bytes por código e, sendo compacta para os sistemas latino e compatível com ASCII nos códigos até 127, fornece um padrão *de facto* de codificação para a conversão de textos para o formato Unicode. É usada pelas mais recentes distribuições Linux como uma substituta para codificações legadas na manipulação de texto. A UTF-8 representa uma forma de otimizar o espaço alocado para textos Unicode. Considerando por exemplo um texto escrito

em língua inglesa, percebe-se que raramente são utilizados caracteres fora do escopo do ASCII, isto é, os primeiros 127 códigos Unicode. Isso significa que se for utilizada uma codificação de largura fixa de 16 bits, o segundo byte de cada carácter muito provavelmente estará vazio, nulo, inutilizado. Para arquivos grandes a sobrecarga desse espaço inútil alocado passa a ser relevante. Tendo uma largura variada, o UTF-8 define que caracteres ASCII são representados com somente um byte. Além de otimizar o espaço alocado no caso de caracteres ASCII, isso garante a paridade entre ASCII e UTF-8, facilitando a tradução de texto ASCII legado para o Unicode. Uma propriedade adicional do UTF-8 diz respeito ao truncamento de cadeias de caracteres Unicode. Alguns códigos (predominantemente legados) de processamento de cadeias de caracteres definem que um único byte nulo (0x00) representa o fim da cadeia. Como visto anteriormente, num texto Unicode de largura fixa de 16 bits, o segundo byte de cada carácter é frequentemente nulo em textos latinos. Isso é interpretado incorretamente como o final da cadeia de texto, problema que não acontece com o UTF-8 devido a otimização do espaço eliminando-se os tais caracteres nulos.

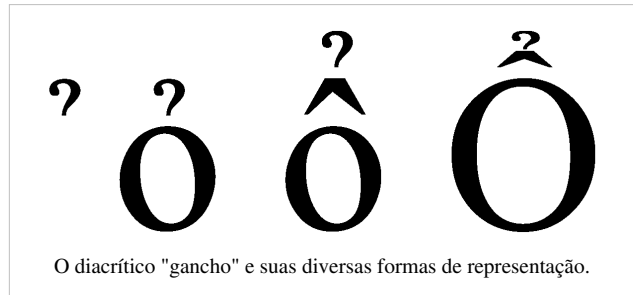
No campo de codificações 16-bit, a UCS-2 é uma codificação de largura fixa que suporta somente o plano básico de línguas, considerado obsoleto. Não faz parte do padrão. Já a UTF-16 é uma codificação 16-bit de largura variável. Ela pode incluir uma ou duas palavras 16-bit para suportar outros caracteres. É usada por várias API, freqüentemente por compatibilidade com códigos desenvolvidos quando o Unicode era baseado em UCS-2. A UTF-16 é o padrão de codificação para a API do sistema operacional Microsoft Windows, da linguagem de programação Java (J2SE, desde a versão 1.5) e dos ambientes .NET. Para caracteres do plano básico de línguas do Unicode, a UCS-2 e a UTF-16 são idênticas. Tanto UCS-2 quanto UTF-16 especificam a marca de ordem de byte (BOM) para ser usada no começo dos arquivos de texto, uma técnica que pode ser usada para a detecção do endiamento de acordo com a arquitetura. Alguns desenvolvedores adotaram a técnica em outras codificações; como no UTF-8, que apesar de não precisar da indicação da ordem do byte, usa a técnica para marcar o texto como sendo Unicode. O BOM é representado pelo código Unicode U+FEFF, e possui a propriedade de desambiguar a ordenação de bytes, independente da codificação Unicode usada. O resultado da má interpretação do endiamento é U+FFFE, um código ilegal. Portanto, o sistema que ler o texto Unicode saberá que deve permutar os bytes se o BOM assim indicar. O código U+FEFF em outros locais do texto senão o início é convertido num símbolo sem aparência nem efeito no texto senão o de prevenir ligaduras tipográficas. Também, as unidades FE e FF nunca aparecem em UTF-8. O mesmo carácter convertido para UTF-8 torna-se a sequência EF BB BF. Entretanto, nem todo texto Unicode possui o BOM.

A UTF-32 é uma codificação 32-bit de largura fixa amplamente usada na representação interna de texto em programas (não durante armazenamento ou transmissão), já que todo sistema operacional Unix que utiliza os compiladores GCC para gerar executáveis utiliza UTF-32 como o padrão de codificação *wide character*. Versões recentes da linguagem de programação Python (desde 2.2) também podem ser configuradas para usar UTF-32 como a representação de Unicode para cadeias de caracteres. A UCS-4 fornece funcionalidade equivalente ao UTF-32, ainda que não faça parte do padrão. Codificações de 32 bits garantem a representação de todos os códigos Unicode sem qualquer tipo de transformação, mas ocupam bastante espaço de armazenamento.

Há também diversas outras codificações disponíveis, algumas delas que representam somente partes de todo o padrão Unicode, como o Windows-1252 (europeu ocidental) e o ISO-8859-1 (Latin-1). Por outro lado, os UTF possuem a capacidade de armazenar todos os códigos Unicode. Para outros subconjuntos do padrão, ver a seção "Subconjuntos padronizados" abaixo.

## Caracteres pré-compostos e caracteres combinados

O Unicode inclui um mecanismo para modificar o formato de caracteres, estendendo o repositório de glifos suportados. Isso cobre o uso de combinação de marcas diacríticas, o que permite que uma letra seja combinada com um acento, por exemplo. Entretanto, por questões de compatibilidade, o padrão também inclui uma quantidade considerável de caracteres pré-compostos, associando o símbolo de uma letra combinada com um acento em um código distinto ao código da letra e ao código do acento. Para vários casos, o utilizador possui várias maneiras de codificar o mesmo caractere. Para padronizar essas opções, o padrão ainda fornece um mecanismo de equivalência canônica.



Um exemplo é o Hangul, o alfabeto coreano. O Unicode fornece um mecanismo para compor sílabas Hangul com seus subcomponentes individuais, conhecidos como Hangul Jamo. Entretanto, o padrão também fornece todas as 11 172 combinações de sílabas Hangul pré-compostas.

Os ideogramas chineses, japoneses e coreanos possuem atualmente códigos apenas para suas formas pré-compostas. Entretanto, a maioria dos ideogramas possuem e combinam elementos mais simples, radicais, que o Unicode poderia decompor, tal como acontece com o Hangul. Tentativas para decompor ideogramas não foram bem sucedidas pois o processo não é tão simples e regular como se esperava.

### Ligaduras

Como em árabe e o devanagari, vários sistemas de escrita possuem regras ortográficas especiais que requerem que certas combinações de letras sejam feitas por ligaduras tipográficas. As regras de formação de ligaduras podem ser complexas, requerendo tecnologias especiais de formatação de texto como o OpenType (da Adobe e Microsoft), Graphite (da SIL International) ou AAT (da Apple). Instruções também são embarcadas em fontes para informar o sistema operacional quanto à impressão correta de diferentes seqüências de caracteres.

## Uso em diversos sistemas

### Sistemas operacionais

O Unicode tornou-se o esquema predominante para o processamento interno de texto, e por vezes também para o armazenamento (apesar de muitos textos ainda estarem armazenados em codificações legadas). As primeiras implementações usavam predominantemente UCS-2, mudando posteriormente para UTF-16 (já que esta codificação é a mais compatível para suportar caracteres fora do plano básico). O sistema mais conhecido nessa situação foi o Windows NT (e seus descendentes Windows 2000 e Windows XP). Os ambientes de *bytecode* das plataformas Java e .NET, o sistema operacional Mac OS X e o ambiente gráfico KDE também uso o UTF-16 para a representação interna.

O UTF-8 (originalmente desenvolvido para o Plan 9) tornou-se a principal codificação para a maioria dos sistemas operacionais *Unix-like* por ser um substituto simples dos conjuntos de caracteres estendidos do ASCII.

## Correio eletrônico

Definido como o padrão da Internet para a extensão do correio eletrônico, o MIME define dois mecanismos diferentes para a codificação de caracteres não-ASCII em e-mails, dependendo de como os caracteres estão no cabeçalhos das mensagens, como no campo "assunto", ou no corpo da mensagem. Em ambos os casos, o conjunto original de caracteres é identificado assim como uma codificação de transferência. Para a transmissão de e-mail Unicode, a codificação de caracteres UTF-8 e a codificação de transferência Base64 são recomendadas. O Base64 garante uma transmissão segura mesmo para servidores de e-mail legados de 7-bit ainda em operação. Caso a mensagem esteja codificada em UTF-7, a codificação de transmissão não é necessária para servidores 7-bit.

Vários aplicativos clientes de correio eletrônico possuem suporte a Unicode no corpo das mensagens. Entretanto, a maioria não envia a mensagem em Unicode por padrão, e poucos sistemas são configurados para exibir todo o repertório do padrão.

O suporte Unicode para o cabeçalho de e-mails é mais problemático, pois diversos padrões devem ser usados para lidar com dados que não sejam ASCII. O RFC 2047 fornece suporte para a codificação não ASCII dos campos "assunto" e "nome real". O RFC 3490 fornece suporte para a codificação do domínio do endereço de e-mail (a parte posterior ao "@"). O nome da caixa de e-mail (a parte anterior ao "@") é limitado a um sub-conjunto de caracteres ASCII visíveis, assim como definido pelo RFC 2822.

No suporte Unicode para o corpo de e-mails, mensagens HTML podem usar entidades HTML para usar qualquer caractere Unicode mesmo que a codificação do e-mail está num padrão legado. Para mensagens em texto puro, deve-se usar MIME para especificar uma codificação.

## World Wide Web

Os navegadores já suportam diversas sub-codificações UTF há vários anos, especialmente UTF-8. Os problemas de visualização derivam principalmente de questões relacionadas a fontes tipográficas. Num caso particular, o Internet Explorer não imprime na tela diversos códigos Unicode exceto quando é indicado explicitamente qual a fonte que contém os símbolos.

Todas as recomendações W3C (a organização que padroniza a WWW) usam o Unicode como o conjunto de caracteres de documentos desde o HTML 4.0, sem um método de mapeamento específico; anteriormente utilizava-se o conjunto ASCII 6-bit ISO-8859-1.

Apesar de regras sintáticas poderem afetar a ordem em

que os caracteres podem aparecer, por definição tanto documentos HTML quanto XML (incluindo o XHTML) suportam caracteres da maioria dos códigos Unicode, exceto pela maioria dos códigos de controle C0 e C1, o bloco de código D800-DFFF e qualquer código que termine em FFFE ou FFFF.

Os caracteres são armazenados em XML e HTML na forma binária do código Unicode (desde que a codificação em uso suporte o código). Alternativamente, pode-se armazená-los como referências numéricas baseadas no seu respectivo código, seguindo o formato `&#valor;` (no qual "valor" é o código em notação decimal) ou `&#xvalor;` (código em notação hexadecimal; note o "x" antes do valor);

Por exemplo, as referências `&#916;`, `&#1049;`, `&#1511;`, `&#1605;`, `&#3671;`, `&#12354;`, `&#21494;`, `&#33865;` e `&#45307;` são visualizadas nos navegadores respectivamente como Δ, Й, ρ, ρ, ω, あ, 叶, 葉 e 𑜋. Se as fontes apropriadas existem, tais símbolos aparecem respectivamente como a letra maiúscula grega "delta", a letra maiúscula cirílica "I curta", a letra árabe "Meem", a letra hebraica "Qof", o numeral tailandês 7, o hiragana



Captura de tela do navegador Firefox 2.0.0.5 ilustrando suporte tanto ao chinês tradicional quanto ao alfabeto latino.

japonês "A", a letra do chinês simplificado "folha", a letra do chinês tradicional "folha" e a sílaba hangul "Nyaelh".

Outro formato para representar caracteres Unicode são as entidades de caractere, um texto que "apelida" um determinado código do padrão. Por exemplo, &mdash;, assim como &#8212; ou &#x2014;, representa o código U+2014, o caractere "—".

Em requisições HTTP, as URL são codificadas obrigatoriamente usando o prefixo "%", geralmente em UTF-8 para representar Unicode.

## Fontes tipográficas

Tanto os padrões TrueType quanto OpenType suportam Unicode, tornando comum a existência de fontes baseadas nessa codificação. Tais formatos de fontes mapeiam códigos Unicode em glifos. Diversas fontes existem no mercado, mas muito poucas suportam a maioria dos códigos Unicode. Fontes Unicode geralmente focam o suporte a ASCII (o básico) e um conjunto particular de códigos, isto é, um conjunto particular de sistema de escrita. O motivo para tal é a falta duma aplicação para a visualização de um grande conjunto de códigos (um programa de computador que contém diversos sistemas de escrita juntos), a quantidade de recursos que as fontes usam do sistema computacional e o fato de os sistemas operacionais e aplicações possuírem inteligência suficiente para obter informação dum glifo a partir dum arquivo diferente se necessário (por exemplo, usando substituição de fontes). A tarefa de desenvolver um conjunto consistente de instruções de visualização para dezenas de milhares de glifos é árdua.


## Subconjuntos padronizados

Diversos subconjuntos do Unicode são padronizados: o Microsoft Windows suporta desde a versão NT 4.0 o WGL-4 com 652 caracteres, que representa todas as línguas européias contemporâneas usando os sistemas de escrita latino, grego ou cirílico. Outros conjuntos padronizados incluem os subconjuntos multilínguas europeus: MES-1 (sistemas latinos somente, 335 caracteres), MES-2 (sistemas latinos, grego e cirílico, 1063 caracteres)<sup>[10]</sup> e MES-3A & MES-3B. Note que o MES-2 inclui todo o MES-1, que por sua vez contém todo o WGL-4.

WGL-4, MES-1 e MES-2			
Linha	Célula(s)	Bloco(s)	Exemplos
00	20–7E	Latino básico (00–7F)	A, i, [, _, @
	A0–FF	Suplemento Latino-1 (80–FF)	Æ, Ç, Ø, ß, ÷
01	00–13, 14–15, 16–2B, 2C–2D, 2E–4D, 4E–4F, 50–7E, 7F	Latino estendido-A (00–7F)	ě, Ĥ, Ŋ, ſ, œ
	8F, 92, B7, DE–EF, FA–FF	Latino estendido-B (80–FF ...)	đ, f, Σ, Ź, y
02	18–1B, 1E–1F	Latino estendido-B (... 00–4F)	
	59, 7C, 92	Extensões IPA (50–AF)	ʈ, ϕ, Ō, ɱ, dʒ
	BB–BD, C6, C7, C9, D6, D8–DB, DC, DD, DF, EE	Letras de modificação de espaço (B0–FF)	+, ˇ, ×, ’, ”
03	74–75, 7A, 7E, 84–8A, 8C, 8E–A1, A3–CE, D7, DA–E1	Grego (70–FF)	σ, Ϛ, Ϝ, Ϟ, Δ
04	00, 01–0C, 0D, 0E–4F, 50, 51–5C, 5D, 5E–5F, 90–91, 92–C4, C7–C8, CB–CC, D0–EB, EE–F5, F8–F9	Cirílico (00–FF)	Д, ІА, ъ, ѓ, ѿ
1E	02–03, 0A–0B, 1E–1F, 40–41, 56–57, 60–61, 6A–6B, 80–85, 9B, F2–F3	Latino estendido adicional (00–FF)	Ƙ, ƕ, Œ, Ū, Ǻ
1F	00–15, 18–1D, 20–45, 48–4D, 50–57, 59, 5B, 5D, 5F–7D, 80–B4, B6–C4, C6–D3, D6–DB, DD–EF, F2–F4, F6–FE	Grego estendido (00–FF)	Ϝ, ϝ, Ϟ, ϟ, Ϡ, ϡ



20	13–14, 15, 17, 18–19, 1A–1B, 1C–1D, 1E, 20–22, 26, 30, 32–33, 39–3A, 3C, 3E	Pontuação geral (00–6F)	% <sub>oo</sub> , !, ?, ‡, “
	44, 4A, 7F, 82	Superescritos e subscritos (70–9F)	<sup>4</sup> , <sub>Θ</sub> , =, (, )
	A3–A4, A7, AC, AF	Símbolos de moeda (A0–CF)	ℳ, €, ₣, £, \$
21	05, 13, 16, 22, 26, 2E	Símbolos <i>letterlike</i> (00–4F)	ℐ, ℒ, ℔, °C, □
	5B–5E	Formas de números (50–8F)	⅔, viii, □, □, 1/
	90–93, 94–95, A8	Setas (90–FF)	□, □, □, ↗, □
22	00, 02, 03, 06, 08–09, 0F, 11–12, 15, 19–1A, 1E–1F, 27–28, 29, 2A, 2B, 48, 59, 60–61, 64–65, 82–83, 95, 97	Operadores matemáticos (00–FF)	⊗, ∫, □, Δ, ∅
23	02, 0A, 20–21, 29–2A	Símbolos técnicos miscelâneos (00–FF)	□, □, □, □, □
25	00, 02, 0C, 10, 14, 18, 1C, 24, 2C, 34, 3C, 50–6C	Desenho de caixas (00–7F)	-->, T, !, J, —
	80, 84, 88, 8C, 90–93	Elementos de bloco (80–9F)	■, [□], —, [□], ■
	A0–A1, AA–AC, B2, BA, BC, C4, CA–CB, CF, D8–D9, E6	Formas geométricas (A0–FF)	▨, ◆, ○, ◁, ◆
26	3A–3C, 40, 42, 60, 63, 65–66, 6A, 6B	Símbolos miscelâneos (00–FF)	☞, ☎, ☠, ☪, ☯
F0	(01–02)	Áreas de uso privado (00–FF ...)	
FB	01–02	Formas de apresentação alfabética (00–4F)	
FF	FD	Área especial	

Quando o programa de computador de visualização não puder visualizar e processar o código Unicode corretamente, geralmente, imprime somente um retângulo, uma área em aberto ou o caractere Unicode de substituição (U+FFFD, ) para indicar que o caractere não é reconhecido. Alguns sistemas tentam retornar mais informações sobre os caracteres não reconhecidos. A fonte *LastResort* da Apple imprime um glifo substituto indicando o bloco Unicode do caractere. Já a fonte Unidode da SIL imprime uma caixa com o valor hexadecimal do caractere.

## Métodos de entrada

Por questões evidentes, um teclado de computador não pode ter uma tecla distinta para cada caractere Unicode existente: sua superfície teria que ser muito grande, tornando-o inutilizável. Por isso, diversos sistemas operacionais fornecem alternativas para digitar qualquer código do repertório Unicode.

No Microsoft Windows (desde a versão 2000), o utilitário "Mapa de Caracteres" fornece controles de edição para toda tabela até U+FFFF. Programas de processamento de texto como o Microsoft Word possuem um controle similar embarcado, através de inserção de símbolo. Quando se sabe o código desejado, pode-se simular a digitação de um caractere Unicode através da combinação Alt++#, no qual # representa o código hexadecimal até FFFF. Por exemplo, Alt++F+1 produzirá o caractere "ñ". Esse método também funciona em vários outros aplicativos Windows, mas não em aplicações que usam o controle de caixa de edição padrão do sistema.

Utilizadores do Apple Macintosh possuem uma funcionalidade similar, no Mac OS X e no Mac OS 8.5 ou superior: Option+#, no qual # é o código Unicode em hexadecimal de quatro dígitos.

O ambiente gráfico GNOME fornece um utilitário tal qual o "Mapa de Caracteres" que mostra os caracteres ordenados pelo bloco Unicode ou pelo sistema de escrita. Quando sabe-se o código desejado, a produção do caractere pode ser feita de acordo com a norma ISO 14755: Ctrl+Shift+#, no qual # é o código Unicode em

hexadecimal. Deve-se adicionar ao código o caractere "U" caso se esteja usando a versão 2.15 ou superior.

O interpretador de comandos do Linux permite que caracteres Unicode sejam produzidos pela combinação Alt+#, em que # é o código Unicode em decimal digitado no teclado numérico. Para funcionar, o modo Unicode deve ser ativado e uma fonte suportada deve ser usada. A norma ISO 14755 (descrita acima) também é implementada.

A partir da versão 7.5 do navegador Opera, é permitido produzir caracteres Unicode diretamente do campo de texto através da combinação Alt+#.

No editor de texto Vim a produção pode ser feita com a combinação Ctrl+V+u+#.

[1] A última versão (julho de 2007) deste livro é:

The Unicode Consortium. *The Unicode Standard: Version 5.0*. 5 ed. [S.l.]: Addison-Wesley Professional, 2006. 1472 p. ISBN 0321480910

[2] O título, em português, significa "O padrão Unicode".

[3] *The Unicode Standard: A Technical Introduction* ([http://www.unicode.org/standard/principles.html#What\\_Characters](http://www.unicode.org/standard/principles.html#What_Characters)) (em inglês).

Página oficial do Unicode (18 de julho de 2007). Página visitada em 20 de julho de 2007.

[4] 16 bits conseguem representar 65536 símbolos distintos, ou  $2^{16}$

[5] Andrew Pollack (20 de fevereiro de 1991). *Universal Computer Code Due* (<http://query.nytimes.com/gst/fullpage.html?res=9D0CE7DF1E3BF933A15751C0A967958260>) (em inglês). The New York Times. Página visitada em 20 de julho de 2007.

[6] *The Unicode Consortium Members* (<http://www.unicode.org/consortium/memblogo.html>) (em inglês). Página oficial do Unicode (17 de julho de 2007). Página visitada em 20 de julho de 2007.

[7] *Roadmaps to Unicode* (<http://www.unicode.org/roadmaps/>) (em inglês). Página oficial do Unicode (10 de dezembro de 2004). Página visitada em 20 de julho de 2007.

[8] 2 bytes = 16 bits;  $65\,536 = 2^{16}$ ; como o espaço de 65 000 valores possui sinal, somente a parte positiva é usada: 32 768 valores.

[9]  $256 = 2^8$ ; isso significa que oito bytes podem representar 256 valores diferentes.

[10] Markus Kuhn (7 de novembro de 1998). *Multilingual European Character Set 2 (MES-2) Rationale* (<http://www.cl.cam.ac.uk/~mgk25/ucs/mes-2-rationale.html>) (em inglês). Área pessoal de Markus Kuhn no sítio da Universidade de Cambridge. Página visitada em 20 de julho de 2007.

## Ligações externas

- Sítio oficial da *Unicode Consortium* (<http://www.unicode.org>) (em inglês)
- Uma wiki com imagens de todos os caracteres Unicode (<http://www.decodeunicode.org/>) (em alemão e em inglês)
- Busca de caracteres Unicode pelo nome (<http://www.fileformat.info/info/unicode/char/search.htm>) (em inglês)
- Todos os caracteres Unicode impressos em uma imagem (<http://www.ianalbert.com/misc/unichart.php>) (em inglês)
- libUniCode-plus Wiki (<http://sourceforge.net/projects/libunicode-plus/>) (em inglês) - criação e manipulação de tabelas Unicode
- Unicode Input Tool/Converter (<https://addons.mozilla.org/pt-BR/firefox/addon/unicode-input-toolconverter/>) (em inglês) - Complemento para o Firefox com suporte ao português.

# Fontes e Editores da Página

**Unicode** *Fonte:* <http://pt.wikipedia.org/w/index.php?oldid=26267928> *Contribuidores:* !Silent, Capmo, Cesarius, ChristianH, Ciacchi, Crolidge, Edgard.magalhaes, EuTuga, Get It, Gfc, Henrique Moreira, Joãoofcf, Leonardo.stabile, LeonardoG, LeonardoRob0t, Manuel Anastácio, RafaAzevedo, Ricvelozo, Ródi, Xandi, 18 edições anónimas

# Fontes, Licenças e Editores da Imagem

**Ficheiro:Tifinagh Unicode table.svg** *Fonte:* [http://pt.wikipedia.org/w/index.php?title=Ficheiro:Tifinagh\\_Unicode\\_table.svg](http://pt.wikipedia.org/w/index.php?title=Ficheiro:Tifinagh_Unicode_table.svg) *Licença:* Public Domain *Contribuidores:* Prince Kassad

**Ficheiro:Oriya font.png** *Fonte:* [http://pt.wikipedia.org/w/index.php?title=Ficheiro:Oriya\\_font.png](http://pt.wikipedia.org/w/index.php?title=Ficheiro:Oriya_font.png) *Licença:* Public Domain *Contribuidores:* -

**Ficheiro:Hook (diacritic).svg** *Fonte:* [http://pt.wikipedia.org/w/index.php?title=Ficheiro:Hook\\_\(diacritic\).svg](http://pt.wikipedia.org/w/index.php?title=Ficheiro:Hook_(diacritic).svg) *Licença:* Public Domain *Contribuidores:* GJo

**Ficheiro:Firefox2.0.0.5 screenshot zh-hant.png** *Fonte:* [http://pt.wikipedia.org/w/index.php?title=Ficheiro:Firefox2.0.0.5\\_screenshot\\_zh-hant.png](http://pt.wikipedia.org/w/index.php?title=Ficheiro:Firefox2.0.0.5_screenshot_zh-hant.png) *Licença:* desconhecido *Contribuidores:* AVRS, FedericoMP, Jcb, Mardus, Pfctdayelise, Remember the dot, Sevela.p, Shack, Yarnalgo

# Licença

---

Creative Commons Attribution-Share Alike 3.0 Unported  
[//creativecommons.org/licenses/by-sa/3.0/](http://creativecommons.org/licenses/by-sa/3.0/)