Unicode®

L

L.1 Introdução

O uso de **codificações de caracteres** inconsistentes (isto é, valores numéricos associados a caracteres) ao desenvolver produtos de softwares globais causa problemas sérios porque os computadores processam informações utilizando números. Por exemplo, o caractere "a" é convertido em um valor numérico de modo que um computador possa manipular esse fragmento de dados. Muitos países e empresas desenvolveram sistemas de codificação incompatíveis com os sistemas de codificação de outros países e empresas. Por exemplo, o sistema operacional Microsoft Windows atribui o valor `0xC0` ao caractere "A com um acento grave", enquanto o sistema operacional Apple Macintosh atribui o mesmo valor a um ponto de interrogação de cabeça para baixo. Isso resulta em representação errônea e possível corrupção de dados.

Na ausência de um padrão universal de codificação de caracteres, os desenvolvedores de softwares globais têm de fazer um extenso processo de localização dos seus produtos antes da distribuição. A **localização** inclui a tradução entre idiomas e adaptação cultural do conteúdo. O processo de localização normalmente inclui modificações significativas no código-fonte (por exemplo, a conversão de valores numéricos e as suposições subjacentes feitas pelos programadores), o que resulta em aumento de custos e atrasos na distribuição do software. Por exemplo, um programador que fala inglês projetaria um produto de software global assumindo que um caractere único poderia ser representado por um byte. Entretanto, quando a localização desses produtos e feita nos mercados asiáticos, as suposições do programador não mais são válidas porque há muito mais caracteres asiáticos e, portanto, a maior parte do código, se não todo ele, precisaria ser reescrita. A localização é necessária em cada distribuição de uma versão. No momento em que ocorre a localização de um produto de software para um mercado em particular, uma versão mais recente, que também precisa ser localizada, pode estar pronta para distribuição. Como resultado, é trabalhoso e caro produzir e distribuir produtos de softwares globais em um mercado em que não há nenhum padrão universal de codificação de caracteres.

Em resposta a essa situação, foi criado o **Padrão Unicode**, um padrão de codificação que facilita a produção e distribuição de softwares. O Padrão Unicode delineia uma especificação para produzir uma codificação consistente dos caracteres e símbolos mundiais. Deve ocorrer a localização de produtos de softwares que tratam texto codificado no Padrão Unicode, mas o processo de localização é mais simples e mais eficiente porque os valores numéricos não precisam ser convertidos e as suposições feitas pelos programadores sobre a codificação de caracteres são universais. O Padrão Unicode é mantido por uma organização sem fins lucrativos chamada **Unicode Consortium**, cujos membros incluem a Apple, IBM, Microsoft, Oracle, Sun Microsystems, Sybase e muitos outros.

Quando o Consortium pensou e desenvolveu o Padrão Unicode, ele queria um sistema de codificação que fosse **universal**, **eficiente**, **uniforme** e **não ambíguo**. Um sistema de codificação universal inclui todos os caracteres comumente utilizados. Um sistema de codificação eficiente permite que arquivos de texto sejam analisados sintaticamente com rapidez. Um sistema de codificação uniforme atribui valores fixos a todos os caracteres. Um sistema de codificação não ambíguo representa um dado caractere de uma maneira consistente. Esses quatro termos são referidos como a base de design do Padrão Unicode.

L.2 Formatos de transformação Unicode

Embora o Unicode incorpore o conjunto de caracteres ASCII limitado (isto é, uma coleção de caracteres), ele inclui um conjunto de caracteres mais abrangente. Em ASCII cada caractere é representado por um byte contendo 0s e 1s. Um byte é capaz de armazenar os números binários de 0 a 255. A cada caractere é atribuído um número entre 0 e 255, portanto sistemas baseados em ASCII suportam somente 256 caracteres, uma pequena fração dos caracteres mundiais. O Unicode estende o conjunto de caracteres ASCII codificando a ampla maioria

dos caracteres mundiais. O Padrão Unicode codifica caracteres em um espaço numérico uniforme entre 0 a 10FFFF hexadecimal. Uma implementação expressará esses números em um dos vários formatos de transformação, selecionando o que melhor se ajusta ao aplicativo particular disponível.

Esses três formatos em uso são chamados **UTF-8**, **UTF-16** e **UTF-32**. O UTF-8, uma forma de codificação de largura variável, requer de um a quatro bytes para expressar cada caractere Unicode. Dados em UTF-8 consistem em bytes de 8 bits (sequências de um, dois, três ou quatro bytes dependendo do caractere que está sendo codificado) é bem adequado para sistemas baseados em ASCII quando há uma predominância de caracteres de um byte (o ASCII representa caracteres como um byte). Atualmente, o UTF-8 é amplamente implementado em sistemas UNIX e em bancos de dados.

A forma de codificação UTF-16 de largura variável expressa os caracteres Unicode em unidades de 16 bits (isto é, como dois bytes adjacentes ou um inteiro curto em muitas máquinas). A maioria dos caracteres Unicode é expressa em uma única unidade de 16 bits. Entretanto, caracteres com valores acima do hexadecimal FFFF são expressos com um par ordenado de unidades de 16 bits chamado **substitutos**. Os substitutos são inteiros de 16 bits no intervalo ente D800 a DFFF, que são unicamente utilizados para o propósito de "escapar" para caracteres numerados mais altos. Aproximadamente um milhão de caracteres pode ser expresso dessa maneira. Embora um par de substituto exija 32 bits para representar caracteres, ele apresenta espaço eficiente para utilizar essas unidades de 16 bits. Os substitutos são caracteres raros nas implementações atuais. Muitas implementações de tratamento de strings são escritas em termos do UTF-16. [*Nota:* Os detalhes e o código de exemplo para tratamento UTF-16 estão disponíveis no site da Web do Unicode Consortium em `www.unicode.org`.]

As implementações que requerem uso significativo de caracteres raros ou scripts inteiros codificados acima do hexadecimal FFFF devem utilizar o UTF-32, uma forma de codificação de largura fixa de 32 bits que normalmente exige duas vezes mais memória do que os caracteres codificados em UTF-16. A principal vantagem da forma de codificação UTF-32 de largura fixa é que ela expressa todos os caracteres uniformemente, portanto é fácil tratá-la em arrays.

Há poucas diretrizes que afirmam quando utilizar uma forma de codificação particular. A melhor forma de codificação a utilizar depende do sistema de computador e do protocolo de negócios, não dos próprios dados. Em geral, a forma de codificação UTF-8 deve ser utilizada onde sistemas de computador e protocolos de negócios requerem que os dados sejam tratados em unidades de 8 bits, particularmente em sistemas legados sendo atualizados, pois isso frequentemente simplifica modificações nos programas existentes. Por essa razão, o UTF-8 tornou-se a forma de codificação preferida na Internet. Da mesma forma, o UTF-16 é a forma de codificação preferida nos aplicativos Microsoft Windows. Há probabilidade de o UTF-32 vir a ser mais amplamente utilizado no futuro à medida que mais caracteres são codificados com valores acima do hexadecimal FFFF. O UTF-32 requer tratamento menos sofisticado que o UTF-16 na presença de pares substitutos. A Figura L.1 mostra as diferentes maneiras como as três formas de codificação tratam a codificação de caracteres.

Caractere	UTF-8	UTF-16	UTF-32
LETRA MAIÚSCULA LATINA A	0x41	0x0041	0x00000041
LETRA GREGA MAIÚSCULA ALFA	0xCD 0x91	0x0391	0x00000391
IDEOGRAMA CJK 4E95 UNIFICADO	0xE4 0xBA 0x95	0x4E95	0x00004E95
LETRA ITÁLICA ANTIGA A	0xF0 0x80 0x83 0x80	0xDC00 0xDF00	0x00010300

Figura L.1 | Correlação entre as três formas de codificação.

L.3 Caracteres e glifos

O Padrão Unicode consiste em caracteres — componentes escritos (isto é, alfabetos, números, marcas de pontuação, marcas de acento etc.) que podem ser representados por valores numéricos. Um exemplo desse caractere é a LETRA MAIÚSCULA LATINA A, U+0041. Na primeira representação do caractere, **U+*aaaa*** é um **valor de código**, em que U + se refere a valores de código Unicode, em oposição a outros valores hexadecimais. O *aaaa* representa um número de quatro dígitos hexadecimais de um caractere codificado. Os valores de código são combinações de bits que representam caracteres codificados. Os caracteres são representados utilizando **glifos** — várias formas, fontes e tamanhos para exibir caracteres. Não há nenhum valor de código para glifos no Padrão Unicode. Os exemplos de glifos são mostrados na Figura L.2.

O Padrão Unicode inclui os alfabetos, ideogramas, lista de sílabas, marcas de pontuação, **diacríticos**, operadores matemáticos e outros recursos que abrangem os idiomas escritos e manuscritos mundiais. Um diacrítico é uma marca especial adicionada ao caractere para distingui-lo de outra letra ou para indicar um acento (por exemplo, em espanhol, o til "~" acima do caractere de "n"). Atualmente, o Unicode fornece valores de código para 96.382 representações de caracteres, com mais de 878.000 valores de código reservados para expansão futura.

Figura L.2 | Vários glifos do caractere A.

L.4 Vantagens e desvantagens do Unicode

O Padrão Unicode tem várias vantagens significativas que promovem seu uso. Uma delas é o impacto que ele tem sobre o desempenho da economia internacional. O Unicode padroniza os caracteres para os sistemas mundiais de escrita de acordo com um modelo uniforme que promove a transferência e compartilhamento de dados. Os programas desenvolvidos com esse esquema mantêm sua exatidão porque cada caractere tem uma única definição (isto é, a é sempre U+0061, % sempre é U+0025). Isso permite que as empresas administrem as altas demandas dos mercados internacionais, processando vários sistemas de escrita ao mesmo tempo. Todos os caracteres podem ser gerenciados de uma maneira idêntica, evitando assim qualquer confusão causada por diferentes arquiteturas de códigos de caracteres. Além disso, gerenciar dados de uma maneira consistente elimina a corrupção de dados, pois os dados podem ser classificados, pesquisados e manipulados utilizando um processo consistente.

Outra vantagem do Padrão Unicode é a portabilidade (isto é, softwares que podem executar em computadores distintos ou em diferentes sistemas operacionais). A maioria dos sistemas operacionais, bancos de dados, linguagens de programação (incluindo o Java e linguagens .NET da Microsoft) e navegadores da Web atualmente suportam, ou estão planejamento suportar, o Unicode.

Uma desvantagem do Padrão Unicode é a quantidade de memória requerida pelo UTF-16 e UTF-32. Os conjuntos de caracteres ASCII têm comprimento de 8 bits, portanto requerem menos espaço de armazenamento do que os conjuntos de caracteres Unicode, de 16 bits padrão. O **conjunto de caracteres de dois bytes** (**Double-Byte Character Set — DBCS**) codifica caracteres asiáticos com um ou dois bytes por caractere. O **conjunto de caracteres de multibyte** (**Multibyte Character Set — MBCS**) codifica caracteres com um número variável de bytes por caractere. Nessas instâncias, as formas de codificação UTF-16 ou UTF-32 podem ser utilizadas com pouco impacto sobre a memória e desempenho.

Outra desvantagem do Unicode é que, embora ele inclua mais caracteres do que qualquer outro conjunto de caracteres em uso comum, ele ainda não codifica todos os caracteres escritos mundiais. Além disso, o UTF-8 e UTF-16 são formas de codificação de largura variável, assim os caracteres ocupam diferentes quantidades de memória.

L.5 Utilizando o Unicode

Várias linguagens de programação (por exemplo, C, Java, JavaScript, Perl, Visual Basic) fornecem algum nível de suporte para o Padrão Unicode. O aplicativo mostrado na Figuras L.3 imprime o texto "Bem-vindo ao Unicode!" em oito diferentes idiomas: inglês, russo, francês, alemão, japonês, português, espanhol e chinês tradicional.

```
1  // Figura L.3: UnicodeJFrame.java
2  // Demonstrando como utilizar o Unicode em programas Java.
3  import java.awt.GridLayout;
4  import javax.swing.JFrame;
5  import javax.swing.JLabel;
6
7  public class UnicodeJFrame extends JFrame
8  {
9     // construtor cria JLabels para exibir Unicode
10    public UnicodeJFrame()
11    {
12       super( "Demonstrating Unicode" );
13
14       setLayout( new GridLayout( 8, 1 ) ); // configura o layout de frame
15
16       // cria JLabels utilizando o Unicode
17       JLabel englishJLabel = new JLabel( "\u0057\u0065\u006C\u0063" +
18          "\u006F\u006D\u0065\u0020\u0074\u006F\u0020Unicode\u0021" );
19       englishJLabel.setToolTipText( "This is English" );
20       add( englishJLabel );
21
22       JLabel chineseJLabel = new JLabel( "\u6B22\u8FCE\u4F7F\u7528" +
23          "\u0020\u0020Unicode\u0021" );
24       chineseJLabel.setToolTipText( "This is Traditional Chinese" );
25       add( chineseJLabel );
```

```
26
27          JLabel cyrillicJLabel = new JLabel( "\u0414\u043E\u0431\u0440" +
28             "\u043E\u0020\u043F\u043E\u0436\u0430\u043B\u043E\u0432" +
29             "\u0430\u0442\u044A\u0020\u0432\u0020Unicode\u0021" );
30          cyrillicJLabel.setToolTipText( "This is Russian" );
31          add( cyrillicJLabel );
32
33          JLabel frenchJLabel = new JLabel( "\u0042\u0069\u0065\u006E\u0076" +
34             "\u0065\u006E\u0075\u0065\u0020\u0061\u0075\u0020Unicode\u0021");
35          frenchJLabel.setToolTipText( "This is French" );
36          add( frenchJLabel );
37
38          JLabel germanJLabel = new JLabel( "\u0057\u0069\u006C\u006B\u006F" +
39             "\u006D\u006D\u0065\u006E\u0020\u007A\u0075\u0020Unicode\u0021");
40          germanJLabel.setToolTipText( "This is German" );
41          add( germanJLabel );
42
43          JLabel japaneseJLabel = new JLabel( "Unicode\u3078\u3087\u3045" +
44             "\u3053\u305D\u0021" );
45          japaneseJLabel.setToolTipText( "This is Japanese" );
46          add( japaneseJLabel );
47
48          JLabel portugueseJLabel = new JLabel( "\u0053\u00E9\u006A\u0061" +
49             "\u0020\u0042\u0065\u006D\u0076\u0069\u006E\u0064\u006F\u0020" +
50             "Unicode\u0021" );
51          portugueseJLabel.setToolTipText( "This is Portuguese" );
52          add( portugueseJLabel );
53
54          JLabel spanishJLabel = new JLabel( "\u0042\u0069\u0065\u006E" +
55             "\u0076\u0065\u006E\u0069\u0064\u0061\u0020\u0061\u0020" +
56             "Unicode\u0021" );
57          spanishJLabel.setToolTipText( "This is Spanish" );
58          add( spanishJLabel );
59       } // fim do construtor UnicodeJFrame
60    } // fim da classe UnicodeJFrame
```

Figura L.3 | Aplicativo Java que utiliza a codificação Unicode.

```
1  // Figura L.4: Unicode.Java
2  // Exibe Unicode.
3  import javax.swing.JFrame;
4
5  public class Unicode
6  {
7     public static void main( String[] args )
8     {
9        UnicodeJFrame unicodeJFrame = new UnicodeJFrame();
10       unicodeJFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11       unicodeJFrame.setSize( 350, 250 );
12       unicodeJFrame.setVisible( true );
13    } // fim do método main
14 } // fim da classe Unicode
```

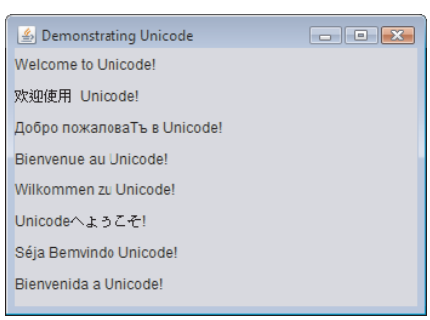

Figura L.4 | Exibindo Unicode.

A classe `UnicodeJFrame` (Figura L.3) usa sequências de escape para representar os caracteres. Uma sequência de escape na forma `\uyyyy`, na qual *yyyy* representa o valor do código de hexadecimal de quatro dígitos. As linhas 17–18 contêm a série de sequências de escape necessária para exibir "Welcome to Unicode!" em inglês. A primeira sequência de escape (`\u0057`) é igualada ao caractere "W", a segunda sequência de escape (`\u0065`) é igualada ao caractere "e" e assim por diante. A sequência de escape `\u0020` (linha 18) é a codificação para o caractere de espaço em branco. As sequências de escape `\u0074` e `\u006F` são igualadas à palavra "to". Observe que "Unicode" não é codificado porque é uma marca comercial registrada e não tem nenhuma tradução equivalente na maioria dos idiomas. A linha 18 também contém a sequência de escape `\u0021` para o ponto de exclamação (`!`).

As linhas 22–56 contêm as sequências de escape para os outros sete idiomas. O site da Web do Unicode Consortium contém um link para gráficos de códigos que listam os valores de códigos Unicode de 16 bits. Os caracteres nos idiomas inglês, francês, alemão, português e espanhol estão localizados no bloco Basic Latin, os caracteres em japonês encontram-se no bloco Hiragana, os caracteres em russo, no bloco agana e os caracteres chineses tradicionais, no bloco Cyrillic CJK Unified. A seção a seguir discute esses blocos.

L.6 Intervalos de caracteres

O Padrão Unicode atribui valores de código, que variam de `0000` (Basic Latin) a `E007F` (Tags), para os caracteres escritos mundiais. Atualmente, há valores de códigos para 96.382 caracteres. Para simplificar a pesquisa de um caractere e seu valor de código associado, geralmente o Padrão Unicode agrupa valores de código por **script** e função (isto é, caracteres latinos são agrupados em um bloco, operadores matemáticos são agrupados em outro bloco etc.). De modo geral, um script é um sistema único de escrita utilizado por múltiplos idiomas (por exemplo, o script Latin é utilizado pelo inglês, francês, espanhol etc.). A página Code Charts no site Unicode Consortium lista todos os blocos definidos e os seus respectivos valores de código. A Figura L.4 lista alguns blocos (script) do site e o seu intervalo de valores de código.

Script	Intervalo dos valores de código
Arabic	U+0600–U+06FF
Basic Latin	U+0000–U+007F
Bengali (Índia)	U+0980–U+09FF
Cherokee (América indígena)	U+13A0–U+13FF
Ideogramas unificados de China, Japão e Coréia (Ásia Oriental)	U+4E00–U+9FFF
Cyrillic (Rússia e Europa Oriental)	U+0400–U+04FF
Ethiopic (Etiópia)	U+1200–U+137F
Greek (Grécia)	U+0370–U+03FF
Hangul Jamo (Coreia)	U+1100–U+11FF
Hebrew	U+0590–U+05FF
Hiragana (Japão)	U+3040–U+309F
Khmer (Camboja)	U+1780–U+17FF
Lao (Laos)	U+0E80–U+0EFF
Mongolian	U+1800–U+18AF
Myanmar	U+1000–U+109F
Ogham (Irlanda)	U+1680–U+169F
Runic (Alemanha e Escandinávia)	U+16A0–U+16FF
Sinhala (Sri Lanka)	U+0D80–U+0DFF
Telugu (Índia)	U+0C00–U+0C7F
Thai	U+0E00–U+0E7F

Figura L.5 | Alguns intervalos de caracteres.