Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
EEB4A91CF3A1B6B70AF7B4BFBB
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
7FF39FA2A3D4BB7144
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
E68FA1CAAF71E183BDF5B180AF
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
F0A8A687C4ADE3A2ADDCB522
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
F4A7BD8BF098898DE588952E52
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
F491A18FE7B795D3B3F3B6939BEE9791
```

.

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

.

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
F7B3AD93EAA0A7C9BDF0B39D93F6998099
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
DDAFF586B78DEBA5ADE78DBB51
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

U Positivo - UTFPR - PUCPr – 30/08/2024 - 18:21:37.6
Prof Dr P Kantek (pkantek@gmail.com)
Unicode e UTF-8 VIVX480a, V: 1.20 75860 JOAO PEDRO DE BRITO DUARTE 24PRG404 - 9 _____ / _____ / _____

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
C98740EFA39DD6BFD589
```

.

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

.

404-75860 - /

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
C4B34CE79DB17EE28AB7
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

U Positivo - UTFPR - PUCPr – 30/08/2024 - 18:21:37.6
Prof Dr P Kantek (pkantek@gmail.com)
Unicode e UTF-8 VIVX480a, V: 1.20 75884 JOAO VICTOR GOMES VILELA GER 24PRG404 - 11 _____ / _____ / _____

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
F3AA8EB3D985F38F97B7F7B7B3892D
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

U Positivo - UTFPR - PUCPr – 30/08/2024 - 18:21:37.6
Prof Dr P Kantek (pkantek@gmail.com)
Unicode e UTF-8 VIVX480a, V: 1.20 75989 LUCAS ANTOSZCZYSZEN
24PRG404 - 12 _____ / _____ / _____

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
C391EF8397F6898F89EAB395F58291BD
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

U Positivo - UTFPR - PUCPr – 30/08/2024 - 18:21:37.6
Prof Dr P Kantek (pkantek@gmail.com)
Unicode e UTF-8 VIVX480a, V: 1.20 75891 LUIZ FELIPE GUEDES
BUCCHERI 24PRG404 - 13 _____ / _____ / _____

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
E983937DF3B69BB7F082BA9FF5BFAD91
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
7DDCB9CD97EFAB87F4829FAF
```

.

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

.

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
E2A6BFE88E87F492A499EAA5A1DFAB
```

.

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

.

U Positivo - UTFPR - PUCPr – 30/08/2024 - 18:21:37.6
Prof Dr P Kantek (pkantek@gmail.com)
Unicode e UTF-8 VIVX480a, V: 1.20 75927 MATHEUS AITA FABRICIO
DE CAR 24PRG404 - 16 _____ / _____ / _____

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
4DC589DDB5E4BF8DEE9FBB
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

U Positivo - UTFPR - PUCPr – 30/08/2024 - 18:21:37.6
Prof Dr P Kantek (pkantek@gmail.com)
Unicode e UTF-8 VIVX480a, V: 1.20 75934 MATHEUS LEANDRO DE BITENCOUR 24PRG404 - 17 _____ / _____ / _____

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
D593E8828F5EF09C90A54B
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
65F1A48F9BF58CBC81C8AD18
```

.

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

.

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
E3A0BBE197A3F0A3B7ABC69D71
```

.

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

.

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
D0AB42DBA15FE4A6A1
```

.

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec

.

U Positivo - UTFPR - PUCPr – 30/08/2024 - 18:21:37.6
Prof Dr P Kantek (pkantek@gmail.com)
Unicode e UTF-8 VIVX480a, V: 1.20 75972 REBECA CABRAL DOS SANTOS 24PRG404 - 21 _____ / _____ / _____

Como trocar informações ?

A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o código EBCDIC. Isto evoluiu para o código ASCII (início de 1960, comite ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, aparecia como uma série de caracteres malucos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham como ser representados no código ASCII.

Unicode

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 30 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos (hieróglifos egípcios) ou mesmo idiomas muito pouco usados (Linear B por exemplo). O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa (são eles: Área privada: U+E000..U+F8FF com 6,400 caracteres; Área suplementar A: U+F0000..U+FFFFD com 65,534 caracteres e Área suplementar B: U+100000..U+10FFFD com 65,534 caracteres).

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjunto de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

UTF=Unicode Transformation Formats

Diversos mecanismos foram sugeridos para implementar o código Unicode, com a preocupação sempre presente de compatibilidade reversa. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C. O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. Suas grandes qualidades são:

- Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
- Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
- Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
- Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Veja-se um resumo do exposto

bits	início	fim	tamanho	byte1	byte2	byte3	byte4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Veja-se agora alguns exemplos de caracteres unicode de 1, 2, 3 e 4 bytes

Character		Binary code point
$	U+0024	0100100
¢	U+00A2	000 10100010
€	U+20AC	00100000 10101100
𤭢	U+24B62	00010 01001011 01100010

Binary UTF-8	Hexadecimal UTF-8
00100100	24
11000010 10100010	C2 A2
11100010 10000010 10101100	E2 82 AC
11110000 10100100 10101101 10100010	F0 A4 AD A2

Uma tabela de conversão binário hexadecimal para ajudar:

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

Exemplos

1. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
F2B1B88DF4BAA8A7E192A9EF92A5F4A584BF
```

Seus codepoints em hexadecimal

```
0B1E0D 13AA27 14A9 F4A5 12513F
```

E em decimal

```
728589 1288743 5289 62629 1200447
```

2. Seja o seguinte conjunto de 5 caracteres UNICODE codificados em UTF-8.

```
39F6B2B6A503E68FAFF59A84B1
```

Seus codepoints em hexadecimal

```
39 1B2DA5 03 63EF 15A131
```

E em decimal

```
57 1781157 3 25583 1417521
```

☞ Para você fazer

A seguir um conjunto de 5 caracteres UNICODE codificados em UTF-8. Você deve examiná-los e descobrir quais os 5 codepoints primeiro escritos em hexadecimal e depois em decimal (é só uma simples conversão).

```
DEA7C28136E391B9EFBD93
```

1 em hexa	2 em hexa	3 em hexa	4 em hexa	5 em hexa
1 em dec	2 em dec	3 em dec	4 em dec	5 em dec