Primeiro Trabalho de Software Básico (2008.2)
Conversão entre Representações de Caracteres

Data de Entrega: 3 de outubro 2008

1 Introdução

Caracteres de um texto em geral são representados em C pelo tipo char, que tipicamente num PC tem o tamanho de 1 byte e pode armazenar apenas 256 valores diferentes. Enquanto a codificação *ASCII* utiliza apenas inteiros de 0 até 127, uma codificação chamada *ISO Latin1* usa a faixa de valores 128-255 para caracteres acentuados e alguns símbolos. Porém, 256 valores não são suficientes para armazenar conjuntamente caracteres de outras linguas como chinês, japonês, árabe, hebraico, etc. Para lidar com essa questão, foi criada a codificação Unicode, que atribui números a caracteres. Atualmente o Unicode é capaz de representar os caracteres de todos os idiomas conhecidos, além de símbolos e outros tipos de caracteres.

Cada caracter em Unicode tem um código, na faixa de 0 a 0x10FFFF. Assim como acontece com a representação de números inteiros negativos, onde o complemento a dois é uma forma de codificação, existem algumas formas de codificação de caracteres Unicode. Para este trabalho, as codificações de interesse são UTF-32 e UTF-8.

A codificação UTF-32 é simplesmente a representação numérica do código do caracter, usando um inteiro de 4 bytes. Essa codificação é suficiente para representar todos os códigos previstos no Unicode, não sendo necessária maior elaboração.

Já no formato UTF-8, os caracteres têm tamanho variável. Um caracter tem tamanho mínimo de 8 bits; porém, se seu código necessitar de mais espaço para ser representado, o formato UTF-8 irá utilizar mais de um byte para representá-lo. Abaixo é apresentada a faixa dos números e o número de bytes necessários para representar cada faixa do código Unicode (onde x é o valo de um bit):

Código Unicode	Representação UTF-8
U+0000 - U+007F	0xxxxxxx
U+0080 - U+07FF	110xxxxx 10xxxxxx
U+0800 - U+FFFF	1110xxxx 10xxxxxx 10xxxxxx
U+10000 - U+10FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Note que:

- O bit x da extrema direita é o bit menos significativo.
- Apenas a menor representação possível de um caracter deve ser utilizada.
- Em uma sequência de vários bytes, o número de **1**s no início do primeiro byte indica o número de bytes da sequência.

Exemplos

1. O sinal © tem código Unicode U+00A9, que em binário é 1010 1001. Logo, sua codificação em UTF-8 será 11000010 10101001 = 0xC2 0xA9.
2. O sinal ≠ tem código Unicode U+2260, que em binário é 0010 0010 0110 0000. Logo, sua codificação em UTF-8 será 11100010 10001001 10100000 = 0xE2 0x89 0xA0.

2 O Trabalho

O trabalho consiste em implementar, na linguagem C, um programa que realiza a transformação entre estas três representações: UTF-32, UTF-8 e Latin1. Ou seja, o seu programa deve abrir um arquivo binário contendo um texto codificado em uma representação fonte e gerar um arquivo binário com o mesmo texto codificado em uma representação destino. A representação fonte e destino, bem como os nomes dos arquivos fonte e destino, serão fornecidos como argumentos da linha de comando, como ilustrado abaixo:

```
converte utf32 latin1 arq-fonte arq-dest
```

Mais adiante apresentamos um modelo de `main.c` para acessar o valor dos argumentos da linha de comando do programa.

No caso do formato UTF-32, os quatro primeiros bytes do arquivo indicam se os valores no arquivo estão em little-endian ou big-endian. A sequência 0x00 0x00 0xFE 0xFF indica big-endian; a sequência 0xFF 0xFE 0x00 0x00 indica little-endian. A saída codificada em UTF-32 deve sempre usar a ordenação little-endian (e portanto começar com a sequência 0xFF 0xFE 0x00 0x00).

A estrutura básica do trabalho pode seguir o seguinte modelo:

- Ler nos parâmetros do programa as codificações de entrada e de saída.
- Abrir os arquivos de entrada e de saída.
- Enquanto o arquivo de entrada não terminar, ler um caracter desse arquivo (segundo a codificação do arquivo) e escrever esse caracter no arquivo de saída (segundo sua codificação).

3 Observações

- O trabalho pode ser feito em grupo de dois alunos. Caso alunos de turmas diferentes queiram formar um grupo, isso deve ser avisado aos dois professores com antecedência mínima de 15 dias da entrega dos trabalhos.
- Os grupos poderão ser chamados para apresentações orais (demonstrações) do trabalho. O não comparecimento de um (ou dos dois) membros do grupo no dia/hora marcada implicarão em nota mínima para os faltantes.
- Os trabalhos deverão ser entregues por email para o professor da turma e em papel (no escaninho do professor da turma). O email deverá ter como anexo um único arquivo, o fonte do seu programa.
- Coloque nas primeiras linhas do arquivo fonte, como comentário, os nomes dos integrantes do grupo na seguinte forma:

  ```
  /* Nome_do_Aluno1 Matrícula1 */
  /* Nome_do_Aluno2 Matrícula2 */
  ```

  Isso é fundamental, pois nosso programa de correção vai obter os dados dos membros do grupo a partir desse comentário no código fonte.
- Na documentação impressa, faça um pequeno relatório explicando o que está funcionando e, principalmente, o que não está funcionando. (Se o programa tiver bugs, você saber disso é sempre um atenuante.)
- Casos de cola serão punidos com nota zero para todos os alunos integrantes dos grupos envolvidos.
- Em caso de dúvidas, não invente. Pergunte!

4 Modelo de Main

```
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

enum Code {utf8, utf32, latin1};

void error (const char *fmt, ...) {
  va_list argp;
  va_start(argp, fmt);
  vfprintf(stderr, fmt, argp);
  va_end(argp);
  exit(1);
}

enum Code getcode (const char *codename) {
  if (strcmp(codename, "latin1") == 0) return latin1;
  if (strcmp(codename, "utf8") == 0) return utf8;
  if (strcmp(codename, "utf32") == 0) return utf32;
  error("código '%s' inválido: os códigos aceitos são "
                  "'latin1', 'utf8' e 'utf32'\n", codename);
  return 0;
}

int main (int argc, char *argv[]) {
  FILE *in, *out;
  enum Code codein, codeout;
  if (argc != 5)
    error("uso correto: %s code-in code-out file-in file-out\n", argv[0]);
  codein = getcode(argv[1]);
  codeout = getcode(argv[2]);
  in = fopen(argv[3], "rb");
  if (in == NULL)
    error("não foi possível abrir arquivo '%s'\n", argv[3]);
  out = fopen(argv[4], "wb");
  if (out == NULL)
    error("não foi possível abrir arquivo '%s'\n", argv[4]);
  /* ... resto do programa ... */
  fclose(in);
  fclose(out);
  return 1;
}
```