Sistemas Operativos II

Engenharia Informática - DEIS

2022/23

Ficha nº 1 - Unicode e Processos

Âmbito da matéria

Assume-se que os alunos conhecem as funções da biblioteca *standard* do C e que têm conhecimento da classe *standard* do C++ para entrada e saída de dados para a consola.

Ao longo desta ficha, será possível:

- Trabalhar com caracteres UNICODE na consola;
- Criar processos utilizando a Windows API e esperar pelo seu término.

Referência bibliográfica

- Capítulos 2 e 6 do Livro Windows System Programming (da Bibliografia) (pgs. 34-37, 181-187, 192-195)
- MSDN: Strings & Unicode e Processes

Exercícios

- 1. Crie um projeto com recurso ao Visual Studio seguindo as seguintes alíneas.
 - a) Construa um projeto vazio do tipo Win32 Console Application no Visual Studio. Durante o processo de criação do projeto, garanta que não está a usar nenhum código inicial de template, nem ficheiros pré-compilados. O projeto deverá estar completamente vazio. Este tipo de projeto é o que vai usar durante algum tempo nestas aulas. Qualquer variação no tipo de projeto (por exemplo, usar header files pré-compilados) irá introduzir entropia no uso imediato, prejudicando o acompanhamento das aulas. Verifique se entendeu o processo de criação e as definições possíveis do IDE.
 - b) Acrescente ao projeto um ficheiro .c com uma função *main* na qual imprime a mensagem "Olá Mundo!". Não se preocupe para já se o 'á' for mal impresso. Garanta apenas que o programa corre.
 - c) Mais tarde pode ser necessário que consiga localizar os ficheiros executáveis dos seus projetos. Usando a linha de comandos do sistema (programa cmd.exe), identifique e explore as diretorias do seu projeto e verifique onde está o ficheiro executável resultante. Veja as diretorias *Debug* e/ou *Release*. Experimente correr o programa a partir da linha de comandos: obviamente, o IDE só é necessário para a construção do programa e não para a sua execução.

- 2. Compile o programa **ex2.c** (código disponível na listagem no fim da ficha). Para isso, defina um projeto *Win32 Console Application (C++)* no *Visual Studio*.
 - a) Execute o programa e confirme que os caracteres acentuados aparecem de forma errada.
 - b) Defina a propriedade do projecto *Character Set* como **UNICODE**. Execute o programa e confirme que, mesmo com esta alteração, o texto que é impresso na consola não apresenta corretamente os caracteres específicos da língua portuguesa (acentos, cedilhas, etc.).
- Construa um novo projeto para compilar o programa ex3.c (código disponível na listagem no fim da ficha). O projeto será do tipo Win32 Console Application (C++), como os anteriores, mas com a propriedade Character Set definida como UNICODE.
 - a) Execute o programa e verifique que, tanto o texto definido no código como o que é inserido pelo utilizador, já aparecem corretamente, mesmo quando têm caracteres especiais da língua.
 - b) De forma a permitir a portabilidade para ASCII, verifique que ao alterar a mesma propriedade *Character Set* para *Multi-Byte*, o projeto continua a compilar.
- 4. Construa um novo projeto para compilar o programa **ex4.c** (código disponível na listagem no fim da ficha). O projeto será do tipo *Win32 Console Application (C++)*, como os anteriores, mas com a propriedade *Character Set* como **UNICODE**. Este exercício é equivalente ao anterior, mas explora o comportamento das bibliotecas de C++ (*cin* e *cout*) em vez das de C (*gets*, *printf*, etc.). Desta forma, repita as alíneas do exercício 3 neste exercício.
- 5. Crie um projeto novo e, utilizando as funções da *Windows API CreateProcess* e *GetModuleFileName* e estruturas do tipo *PROCESS_INFORMATION* e *STARTUP_INFO*, execute as seguintes ações:
 - a) Faça com que o programa apresente no ecrã o nome do ficheiro executável que lhe corresponde.
 - b) Modifique o programa de forma a que peça ao utilizador o nome de um outro programa (ficheiro executável) que é posteriormente executado no contexto de um novo processo, mantendo-se o seu programa em execução e repetindo o comportamento até o utilizador desejar sair.
 - c) Acrescente a possibilidade de o utilizador especificar um argumento a passar ao outro programa. Teste essa funcionalidade mandando executar o programa *notepad* passando-lhe o nome de um ficheiro de texto. Se tudo estiver a funcionar, o *notepad* é executado abrindo o ficheiro indicado.
 - d) Modifique o programa de forma a que deixe de pedir nome de programas a executar e simplesmente se execute novamente mais 3 vezes. Cada nova execução lança a seguinte (apenas uma de cada vez), criando uma cadeira de execuções que pára na quarta execução. O comportamento é automático e não requer a intervenção do utilizador. Cada execução deve indicar uma mensagem que permita distingui-la das restantes.

Listagem de Programas

ex2.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#define MAX 256

int main(int argc, char* argv[]) {
        char result[MAX] = "Olá! Este programa ainda não representa UNICODE\n";
        printf("Frase:%s Tamanho:%d\n", result, strlen(result));
        getchar();
        return 0;
}
```

ex3.c

#endif do{

#include <windows.h>

#include <tchar.h>
#include <fcntl.h>
#include <io.h>
#include <stdio.h>
#define MAX 256

int _tmain(int argc, LPTSTR argv[]) {
 TCHAR str[MAX], result[MAX] = TEXT("Olá! Este programa é para aceitar UNICODE.
 Insira \'fim\' para sair\n");
 unsigned int i;
 //UNICODE: Por defeito, a consola Windows não processa caracteres wide.
 //A maneira mais fácil para ter esta funcionalidade é chamar _setmode:
#ifdef UNICODE
 _setmode(_fileno(stdin), _O_WTEXT);
 setmode(_fileno(stdout), _O_WTEXT);

_tprintf(result);
fflush(stdin);

//Retirar \n

//Maiúsculas

return 0;

_fgetts(str, MAX, stdin);

 $str[_tcslen(str) - 1] = '\0';$

} while (tcsicmp(TEXT("FIM"), str));

stprintf s(result,MAX, TEXT("Frase:%s, Tamanho:%d\n"),str, tcslen(str));

ex4.c

```
#include <windows.h>
#include <tchar.h>
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <string>
using namespace std;
//Permitir que o mesmo código possa funcionar para ASCII ou UNICODE
#ifdef UNICODE
#define tcout wcout
#define tcin wcin
#define tstring wstring
#else
#define tcout cout
#define tcin cin
#define tstring string
#endif
int tmain(int argc, LPTSTR argv[]){
 tstring str = TEXT("Olá! Este programa é para aceitar UNICODE. Insira \'fim\'
para sair\n");
 //UNICODE: Por defeito, a consola Windows não processe caracteres wide.
  //A maneira mais fácil para ter esta funcionalidade é chamar setmode:
#ifdef UNICODE
  _setmode(_fileno(stdin), _O_WTEXT);
  _setmode(_fileno(stdout), _O_WTEXT);
  tcout << str;
  do {
      getline(tcin, str);
      //Maiúsculas
      for (unsigned int i = 0; i < str.length(); i++)</pre>
            str[i] = _totupper(str[i]);
      tcout << TEXT("Frase: ") << str << TEXT("Tamanho:") << str.length() << endl;</pre>
  } while (str != TEXT("FIM"));
  return 0;
```