Вивчення системних викликів Win32 API, МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут»»

Факультет радіоелектроніки, комп’ютерних систем та інфокомунікацій

Кафедра комп’ютерних систем, мереж і кібербезпеки

**Лабораторна робота**

з Cистемне програмування

(назва дисципліни)

|  |  |
| --- | --- |
| на тему: | Вивчення системних викликів Win32 API, |
|  | функцій збору інформації про систему |
|  |  |
|  |  |
|  |  |

НОМЕР\_ЗАЧЕТКИ

|  |  |  |
| --- | --- | --- |
| Виконав: | | студент 3 курсу групи № 535а |
| напряму підготовки (спеціальності) | | |
| 123-«Комп’ютерна інженерія» | |
|  | |
| (шифр і назва напряму підготовки (спеціальності)) | |
| Олiйник О. К. | |
| (прізвище й ініціали студента) | |
| Прийняв: | асистент |
|  | Мозговий М.В. |
| (посада, науковий ступінь, прізвище й ініціали) | | |

|  |  |
| --- | --- |
| Національна шкала: |  |
| Кількість балів: |  |
| Оцінка: ECTS |  |

Харків — 2020

# Тема и цель работы

## Цель

1. Изучение системных вызовов Win32 API, позволяющих получить информацию об ошибке.
2. Изучение функций сбора информации о системе.

Постановка задачи

1. Необходимо написать программу, которая бы генерировала ошибку в ходе выполнения системного вызова и выдавала системное описание данной ошибки. Вторым режимом работы данной программы должен быть вывод информации о состоянии системы. Режим запуска программы определяется ключом, передаваемым в командной строке (-e – печать ошибки, -s – печать информации о системе).
2. Написать программу, позволяющую выполнять перекодировку текста из ASCII в Юникод и обратно. В качестве входных данных выступает файл с текстом. Программа при запуске получает параметр командной строки определяющий исходную кодировку файла (-a – ANSI файл, -u – Юникод файл).

## Выполнение работы

Ниже приведен код первой программы.

#include "windows.h"

#include "stdio.h"

#include "string.h"

#include "locale.h"

int main(int argc, char\* argv[])

{

setlocale(LC\_ALL, "rus");

if (argc != 2)

{

printf\_s("Incorrect input!");

}

else

{

if (strcmp(argv[1], "-e") == 0)

{

MEMORYSTATUS ms;

GlobalMemoryStatus(&ms);

HLOCAL memory = LocalAlloc(LPTR, ms.dwTotalPhys \* (3));

if (memory == NULL) {

LPVOID message;

DWORD code = GetLastError();

FormatMessage(FORMAT\_MESSAGE\_ALLOCATE\_BUFFER | FORMAT\_MESSAGE\_FROM\_SYSTEM, NULL, code, MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT), (LPTSTR)&message, 0, NULL);

wprintf(L"\nError code: 0x%x\n", code);

wprintf(L"Сообщение об ошибке: %s\n", (char\*)message);

}

}

else if (strcmp(argv[1], "-s") == 0)

{

MEMORYSTATUSEX ms;

SYSTEM\_INFO proc;

GlobalMemoryStatusEx(&ms);

GetSystemInfo(&proc);

printf\_s("\nCPU Info:");

printf\_s("Number of processors: %u\n", proc.dwNumberOfProcessors);

printf\_s("Processor architectire: %u\n", proc.wProcessorArchitecture);

printf\_s("Processor level: %u\n", proc.wProcessorLevel);

printf\_s("Processor revision: %u\n", proc.wProcessorRevision);

printf\_s("Active processor mask: %u\n", proc.dwActiveProcessorMask);

printf\_s("Page size: %u\n", proc.dwPageSize);

printf\_s("Minimum application address: %lx\n", proc.lpMinimumApplicationAddress);

printf\_s("Maximum application address: %lx\n", proc.lpMaximumApplicationAddress);

printf\_s("Information about memory:\n");

printf\_s("Availble memory: %d bytes;\n", ms.ullAvailPhys);

printf\_s("Total memory: %d bytes;\n", ms.ullTotalPhys);

}

else

{

printf\_s("Incorrect key!");

}

}

return 0;

}

Ниже приведен скриншот работы программы.

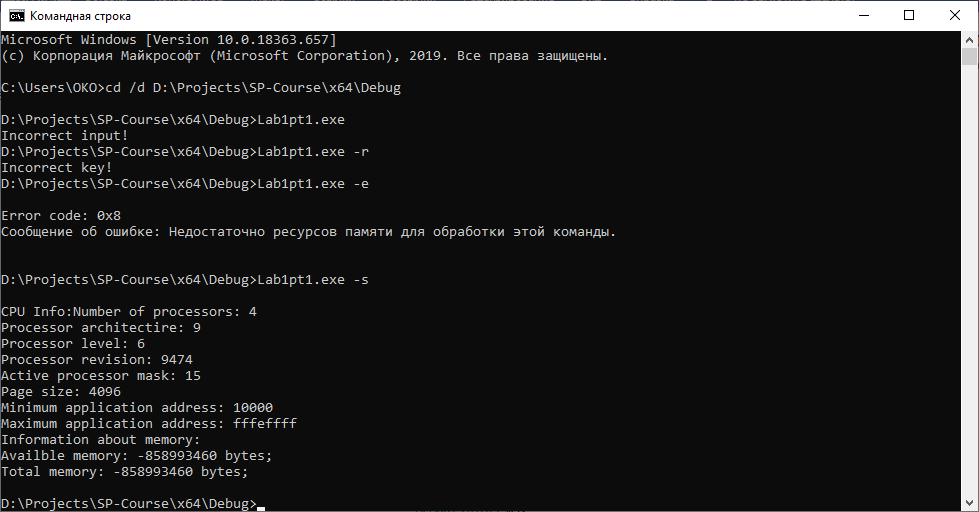


Рисунок 1 – Скриншот работы первой программы

Ниже приведен код второй программы.

#define AmountOfBytesToRead 2048

#include "windows.h"

#include "stdio.h"

#include "string.h"

#include "locale.h"

void FromUnicodeToAscii(char\*);

void FromAsciiToUnicode(char\*);

bool CreateResultAndOpenSourceFiles(char\*, HANDLE\*, HANDLE\*);

int main(int argc, char\* argv[])

{

if (argc != 3)

{

printf\_s("Incorrect input!");

}

else

{

if (strcmp(argv[1], "-a") == 0)

{

HANDLE readFile;

HANDLE writeFile;

bool Result = true;

readFile = CreateFileA(argv[2], GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

writeFile = CreateFileA("output.txt", GENERIC\_WRITE, FILE\_SHARE\_READ, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (readFile != INVALID\_HANDLE\_VALUE && writeFile != INVALID\_HANDLE\_VALUE){

BYTE buff[AmountOfBytesToRead];

DWORD dwBytes;

while (ReadFile(readFile, buff, AmountOfBytesToRead, (LPDWORD)&dwBytes, NULL))

{

if (dwBytes == 0) break;

int uLength = MultiByteToWideChar(CP\_UTF8, 0, (LPCCH)buff, dwBytes, NULL, 0);

wchar\_t\* uStr = new wchar\_t[uLength];

MultiByteToWideChar(CP\_UTF8, 0, (LPCCH)buff, dwBytes, uStr, uLength);

if (!WriteFile(writeFile, uStr, uLength \* sizeof(wchar\_t), (LPDWORD)&dwBytes, NULL)) {

printf("Save error!");

break;

}

}

CloseHandle(readFile);

CloseHandle(writeFile);

}

}

else if (strcmp(argv[1], "-u") == 0)

{

HANDLE readFile;

HANDLE writeFile;

bool Result = true;

readFile = CreateFileA(argv[2], GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

writeFile = CreateFileA("output.txt", GENERIC\_WRITE, FILE\_SHARE\_READ, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (readFile != INVALID\_HANDLE\_VALUE && writeFile != INVALID\_HANDLE\_VALUE){

BYTE buff[AmountOfBytesToRead];

DWORD dwBytes;

while (ReadFile(readFile, buff, AmountOfBytesToRead, (LPDWORD)&dwBytes, NULL))

{

if (dwBytes == 0) break;

int uLength = MultiByteToWideChar(CP\_UTF8, 0, (LPCCH)buff, dwBytes, NULL, 0);

wchar\_t\* uStr = new wchar\_t[uLength];

MultiByteToWideChar(CP\_UTF8, 0, (LPCCH)buff, dwBytes, uStr, uLength);

int aLength = WideCharToMultiByte(CP\_ACP, 0, uStr, uLength, NULL, 0, NULL, NULL);

char\* aStr = new char[aLength];

WideCharToMultiByte(CP\_ACP, 0, uStr, uLength, aStr, aLength, NULL, NULL);

if (!WriteFile(writeFile, aStr, aLength \* sizeof(char), &dwBytes, NULL)) {

printf("Save error!");

break;

}

}

CloseHandle(readFile);

CloseHandle(writeFile);

}

}

else

{

printf\_s("Incorrect key!");

}

}

return 0;

}

Ниже приведен скриншот работы программы.

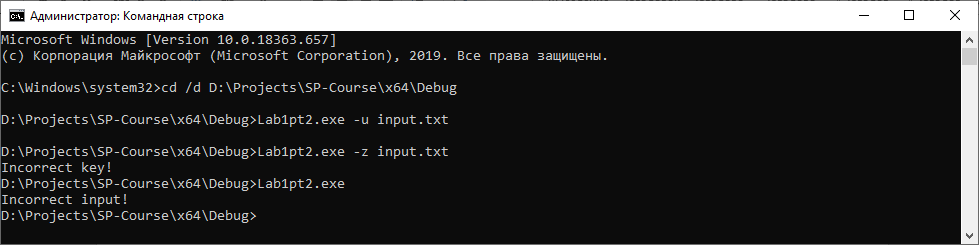


Рисунок 2 – Скриншот работы второй программы

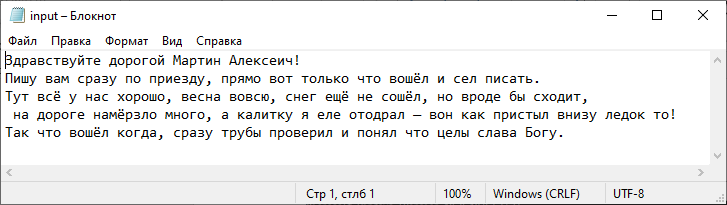


Рисунок 3 – Исходный файл в Юникод кодировке

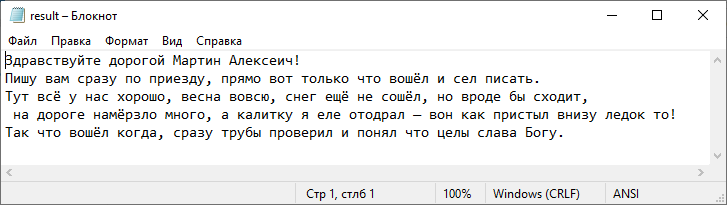


Рисунок 4 – Результат перекодирования в ASCII

**Выводы**

На лабораторной работе мы повторили основы работы со средой разработки Visual Studio и языка программирования С/С++, а также написали программы, требуемые в задании к лабораторной работе.