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 Системного програмування

(назва дисципліни)

на тему: «Вивчення системних викликів Win32 API, функцій збору інформації про систему»

Виконав: студент 3-го курсу групи № 535а

напряму підготовки (спеціальності)

123-«Комп’ютерна інженерія»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва напряму підготовки (спеціальності))

Скринник Є. І.

(прізвище й ініціали студента)

Прийняв: асистент каф.503

Мозговий М.В.

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала: \_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Харків – 2020

**Цель работы:**

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).

**Код первой программы:**

#include <stdio.h>

#include <Windows.h>

#include <locale.h>

// Function that generates system error.

void generate\_error();

// Function that receive information about the current system.

void get\_information();

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

{

setlocale(LC\_ALL, "Russian");

if (argv[1] != NULL) {

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

{

get\_information();

}

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

{

generate\_error();

}

}

else

{

printf("Введите параметры, и попробуйте еще раз!\n");

}

system("pause");

return 0;

}

void generate\_error() {

MEMORYSTATUS memory\_info;

// Receiving info about memory.

GlobalMemoryStatus(&memory\_info);

// ========================================

// Generate an error

// Trying to allocate all virtual memory memory.

if (LocalAlloc(LPTR, memory\_info.dwTotalVirtual) == NULL) {

LPVOID error\_message;

FormatMessage(

FORMAT\_MESSAGE\_ALLOCATE\_BUFFER | FORMAT\_MESSAGE\_FROM\_SYSTEM,

NULL,

GetLastError(),

MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT),

(LPTSTR)&error\_message,

0, NULL);

printf("Системное описания произошедшей ошибки:\n%s", error\_message);

}

return;

}

void get\_information() {

//Contains information about system.

SYSTEM\_INFO system\_info;

//Copy the system information to the structure system\_info.

GetSystemInfo(&system\_info);

// Printing information about system.

printf("System information: \n");

printf("Oem Id: %u\n", system\_info.dwOemId);

printf("Processor architecture: %u\n", system\_info.wProcessorArchitecture);

printf("Page size: %u\n", system\_info.dwPageSize);

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

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

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

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

printf("Processor type: %u\n", system\_info.dwProcessorType);

}

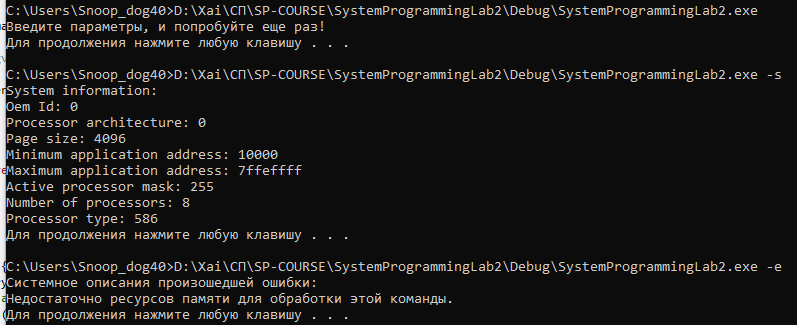
**Результат работы:**

Рисунок 1 – Результат отработки программы в трех случаях, без параметров и с параметрами -s и -e.

**Код второй программы:**

#include <stdio.h>

#include <string.h>

#include <Windows.h>

#include <locale.h>

#define AmountOfBytesToRead 2048

void FromUnicodeToAscii(char\*);

void FromAsciiToUnicode(char\*);

// Функция для создания файла результата, и открытия исходного текста.

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

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

setlocale(LC\_ALL, "Russian");

if (argv[1] != NULL && argv[2] != NULL) {

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

FromUnicodeToAscii(argv[2]);

}

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

FromAsciiToUnicode(argv[2]);

}

else {

printf("Неверные параметры!");

}

}

else {

printf("Введите параметры, и попробуйте еще раз!\n");

}

system("pause");

return 0;

}

void FromUnicodeToAscii(char\* path) {

HANDLE readFile;

HANDLE writeFile;

if (CreateResultAndOpenSourceFiles(path, &readFile, &writeFile)) {

BYTE buff[AmountOfBytesToRead];

DWORD dwBytes;

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

{

if (dwBytes == 0) break;

//сначала вы вызываете MultiByteToWideChar, чтобы получить размер конечной строки (юникод)

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

//указатель на юникод строку, которую необходимо конвертировать

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

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

//Вызов WideCharToMultiByte для получения размера конечной строки (ANSI)

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

//Указатель на буфер, который получает преобразованную строку.

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("Ошибка записи!");

break;

}

}

CloseHandle(readFile);

CloseHandle(writeFile);

}

return;

}

void FromAsciiToUnicode(char\* path) {

HANDLE readFile;

HANDLE writeFile;

if (CreateResultAndOpenSourceFiles(path, &readFile, &writeFile)) {

BYTE buff[AmountOfBytesToRead];

DWORD dwBytes;

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

{

if (dwBytes == 0) break;

//Определение размера строки Unicode

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

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

//Конечная перекодировка

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

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

printf("Ошибка записи!");

break;

}

}

CloseHandle(readFile);

CloseHandle(writeFile);

}

return;

}

bool CreateResultAndOpenSourceFiles(char\* path, HANDLE\* readFile, HANDLE\* writeFile) {

bool Result = true;

\*readFile = CreateFileA(path, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

\*writeFile = CreateFileA("C:\\Users\\Snoop\_dog40\\Desktop\\recording\_result.txt", GENERIC\_WRITE, FILE\_SHARE\_READ, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (\*readFile == INVALID\_HANDLE\_VALUE || \*writeFile == INVALID\_HANDLE\_VALUE) Result = false;

return Result;

}

**Результат работы:**

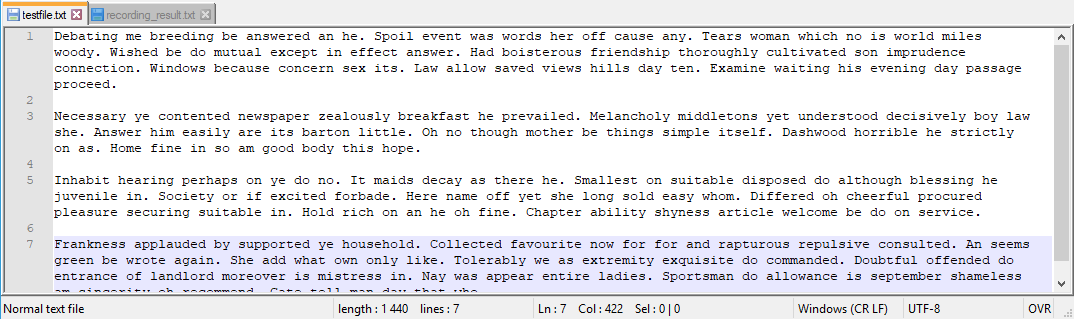
****

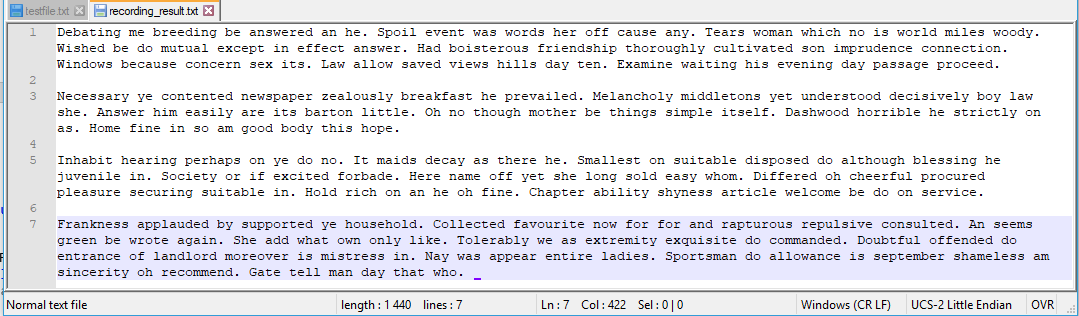
Рисунок 2 – Исходный файл в ASCII кодировке.****

Рисунок 3 – Результат перекодирования в Юникод.