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++»

Виконав: студент 3-го курсу групи №525ст2
напряму підготовки (спеціальності)
123-«Комп'ютерна інженерія»
(шифр і назва напряму підготовки (спеціальності))
Золотопуп А.С.
(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:

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

Постановка задачи:

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).

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 программы:

```
#include <iostream>
#include <windows.h>
#include <stdio.h>
#include <tchar.h>
#include <strsafe.h>
#include <string>
#include <locale>
#include <codecvt>
#include <fstream>

std::string unicode2ansi(const std::wstring& wstr)
{
int size_needed = WideCharToMultiByte(CP_ACP, 0, &wstr[0], -1, NULL, 0,
NULL, NULL);
std::string strTo(size_needed, 0);
WideCharToMultiByte(CP_ACP, 0, &wstr[0], (int)wstr.size(), &strTo[0],
size_needed, NULL, NULL);
return strTo;
}

std::wstring ansi2unicode(const std::string& str)
{
int size_needed = MultiByteToWideChar(CP_ACP, 0, &str[0], (int)str.size(),
NULL, 0);
std::wstring wstrTo(size_needed, 0);
```

```
MultiByteToWideChar(CP_ACP, 0, &str[0], (int)str.size(), &wstrTo[0],
size_needed);
return wstrTo;
}
void generate_error()
{
MEMORYSTATUS memory_info;
GlobalMemoryStatus(&memory_info);
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("System description error:\:\n%s", error_message);
}
return;
}
void get_information()
{
SYSTEM_INFO system_info;
GetSystemInfo(&system_info);
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);
}

int main(int argc, char* argv[])
{
if (argv[1] != NULL) {
        if (strcmp(argv[1], "-s") == 0)
```

```
        {
                get_information();
        }
        if (strcmp(argv[1], "-e") == 0)
        {
                generate_error();
        }
}
else
{
        printf("Print parameter and try again!\n");
}
std::locale loc(std::locale(), new std::codecvt_utf16<wchar_t>);
std::string name;
std::cout << "Input file name:";
std::cin >> name;
name = "Inputs//" + name;
if (argc > 1 && (std::string(argv[1]) == "-a")) {
        std::ifstream file(name);
        std::basic_ofstream<wchar_t> ofs("Unicode.txt");
        ofs.imbue(loc);
        ofs << (wchar_t)0xfeff;
        std::string str;
        while (std::getline(file, str))
        {
                ofs << ansi2unicode(str);
        }
}
else {
        std::wifstream file(name);
        file.imbue(loc);
        file.seekg(2);
        std::ofstream ofs("ANSI.txt");
        std::wstring str;
        while (std::getline(file, str))
        {
                ofs << unicode2ansi(str);
        }
}
std::cout << "Done!" << std::endl;

system("pause");
return 0;
}
```

Результат работы:

```
C:\Users\nonst\source\repos\laba1SP\Debug>laba1SP.exe
Print parameter and try again!
Input file name:1.txt
Done!
Для продолжения нажмите любую клавишу . . .

C:\Users\nonst\source\repos\laba1SP\Debug>laba1SP.exe -e
System description error::
□□5□4□>□A□B□0□B□>□G□=□>□ Input file name:2.txt
Done!
Для продолжения нажмите любую клавишу . . .

C:\Users\nonst\source\repos\laba1SP\Debug>laba1SP.exe -s
System information:
Oem Id: 0
Processor architecture: 0
Page size: 4096
Minimum application address: 10000
Maximum application address: 7ffeffff
Active processor mask: 4095
Number of processors: 12
Processor type: 586
Input file name:1.txt
Done!
Для продолжения нажмите любую клавишу . . .

C:\Users\nonst\source\repos\laba1SP\Debug>laba1SP.exe -a
Input file name:2.txt
Done!
Для продолжения нажмите любую клавишу . . .

C:\Users\nonst\source\repos\laba1SP\Debug>laba1SP.exe -a
Input file name:1.txt
Done!
Для продолжения нажмите любую клавишу . . .
```

Выводы:

В результате выполнения данной лабораторной работы были изучены системные вызовы Win32 API, позволяющие получить информацию об ошибке, а также функции сбора информации о системе. Так же работа с кодировками и перекодировками.