Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное

учреждение высшего образования

Московский технический университет связи и информатики

Кафедра «Системного программирования»

Лабораторная работа №1. Часть 2.

по дисциплине

«Операционные системы»

Выполнила: студентка группы БСТ2104

Первухина А. А.

Проверила: Алексанян Диана Ашотовна

Москва 2023

**Оглавление**

[Цель работы 3](#_Toc146627773)

[Лабораторное задание 3](#_Toc146627774)

[Код программы 4](#_Toc146627775)

[Результаты работы 9](#_Toc146627776)

[Вывод 9](#_Toc146627777)

# Цель работы:

Получение практических навыков использования Win32 API для исследования процессов

# Лабораторное задание:

Написать программу, которая выполняет следующие действия в интерактивном режиме:

• Принимая дескриптор, имя или полное имя модуля, возвращает другие два элемента в своих выходных параметрах (выполнить задание для своей программы и для любой известной библиотеки).

• Будет выполнять последовательно по шагам следующие задачи:

o Используя функцию GetCurrentProcessId определит идентификатор текущего процесса.

o Используя функцию GetCurrentProcess определит псевдодескриптор текущего процесса.

o Используя функцию DuplicateHandle и значение псевдодескриптора определит дескриптора текущего процесса.

o Используя функцию OpenProcess определит копию дескриптора текущего процесса.

o Закроет дескриптор, полученный функцией DuplicateHandle.

o Закроет дескриптор, полученный функцией OpenProcess.

• Выдаст список перечисления всех процессов, потоков, модулей и их свойства в системе: для Windows NT (и список загруженных драйверов устройств).

# Код программы

#include <iostream>

#include <Windows.h>

#include <stdio.h>

#include <tlhelp32.h>

#include <conio.h>

#include <locale>

#include <psapi.h>

using namespace std;

// Получить дескриптор, имя и полное имя

void getHandelNameFName() {

HMODULE hModule = GetModuleHandle(NULL);

wchar\_t\* FName = new wchar\_t[256];

GetModuleFileName(hModule, FName, 256);

setlocale(LC\_ALL, "rus");

wcout.imbue(locale("rus\_rus.1251"));

wchar\_t \*name = wcsrchr(FName, L'\\');

wchar\_t \*Name = new wchar\_t[256];

wcsncpy\_s(Name, 256, name + 1, 256);

std::wcout << "Handle: " << hModule<< endl<< "Name: "<< Name << endl << "FName: " << FName << endl;

}

// Получить имя и дескриптор, используя полное имя

void getHandleName() {

setlocale(LC\_ALL, "rus");

wcout << "Введите FName: ";

wchar\_t\* FName = new wchar\_t[256];

wcin >> FName;

wchar\_t \*name = wcsrchr(FName, L'\\');

wchar\_t \*Name = new wchar\_t[256];

if (name != NULL) {

wcsncpy\_s(Name, 256, name + 1, 256);

HMODULE hModule = GetModuleHandle(FName);

wcout << "Handle: " << hModule << endl << "Name: " << Name << endl << "FName: " << FName << endl;

}

}

// Получить дескриптор и полное имя, используя имя

void getHandleFName() {

cout << "Введите Name: ";

wchar\_t\* Name = new wchar\_t[256];

wcin >> Name; wchar\_t\* FName = new wchar\_t[256];

GetModuleFileName(GetModuleHandle(Name), FName, 256);

HMODULE hModule = GetModuleHandle(FName);

wcout << "Handle: " << hModule << endl << "Name: " << Name << endl << "FName: " << FName << endl;

}

// Получить полное имя и имя, используя дескриптор

void getNameFName() {

cout << "Введите Handle: ";

wchar\_t\* temp = new wchar\_t[10];

wcin >> temp;

wchar\_t \*endptr;

long num = wcstoul(temp, &endptr, 16);

HMODULE hModule = (HMODULE)num;

wchar\_t\* FName = new wchar\_t[256];

GetModuleFileName(hModule, FName, 256);

wchar\_t \*name = wcsrchr(FName, L'\\');

wchar\_t \*Name = new wchar\_t[256];

wcsncpy\_s(Name, 256, name + 1, 256);

wcout << "Handle: " << hModule << endl << "Name: " << Name << endl << "FName: " << FName << endl;

delete[] temp;

}

// Инфо о текущем процессе, псевдодескрипторе и дескрипторе. Создание копии текущего процесса и закрытие

void Handels() {

setlocale(LC\_ALL, "rus");

wcout.imbue(locale("rus\_rus.1251"));

DWORD dCurrentId = GetCurrentProcessId();

HANDLE hCurrentPseudo = GetCurrentProcess();

HANDLE hCurrentIdDup;

DuplicateHandle(hCurrentPseudo, hCurrentPseudo, hCurrentPseudo, &hCurrentIdDup, 0, FALSE, DUPLICATE\_SAME\_ACCESS);

HANDLE hCurrentIdOpen = OpenProcess(PROCESS\_DUP\_HANDLE, TRUE, dCurrentId);

BOOL fClosedDup = CloseHandle(hCurrentIdDup);

BOOL fClosedOpen = CloseHandle(hCurrentIdOpen);

wcout << "Идентификатор текущего процесса: " << dCurrentId << "\nПсевдодескриптор текущего процесса: " << hCurrentPseudo << "\nДескриптор, полученный функцией DuplicateHandle: " << hCurrentIdDup << "\nДескриптор, полученный функцией OpenProcess: " << hCurrentIdOpen << "\nЗакрываем дескриптор, полученный функцией OpenProcess: " << fClosedOpen << "\nЗакрываем дескриптор, полученный функцией DuplicateHandle: " << fClosedDup << endl;

}

// Инфо о всех процессах и вывод атрибутов

void ShowProcesses() {

setlocale(LC\_ALL, "rus");

wcout.imbue(locale("rus\_rus.1251"));

HANDLE Snapshot = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

wcout << "\nSnapshot ID = " << Snapshot;

PROCESSENTRY32 entry;

entry.dwSize = sizeof(PROCESSENTRY32);

BOOL fRepeat = Process32First(Snapshot, &entry);

int i = 0;

while (fRepeat) {

i++;

wcout << "\n\nth32ProcessID = " << entry.th32ProcessID;

wcout << "\nth32DefaultHeapID = " << entry.th32DefaultHeapID;

wcout << "\nth32ModuleID = " << entry.th32ModuleID;

wcout << "\nth32ParentProcessID = " << entry.th32ParentProcessID;

wcout << "\ncntUsage = " << entry.cntUsage;

wcout << "\ncntThreads = " << entry.cntThreads;

wcout << "\npcPriClassBase = " << entry.pcPriClassBase;

wcout << "\ndwFlags = " << entry.dwFlags;

wcout << "\nszExeFile = " << entry.szExeFile;

fRepeat = Process32Next(Snapshot, &entry);

}

CloseHandle(Snapshot);

}

// Инфо о всех потоках системы текущего процесса и их атрибуты

void ShowThreads() {

setlocale(LC\_ALL, "rus");

wcout.imbue(locale("rus\_rus.1251"));

HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS\_SNAPTHREAD, 0);

wcout << "\nSnapshot ID = " << hSnapshot;

THREADENTRY32 entry;

entry.dwSize = sizeof(THREADENTRY32);

BOOL fRepeat = Thread32First(hSnapshot, &entry);

DWORD dwCurentProcessId = GetCurrentProcessId();

while (fRepeat) {

if (entry.th32OwnerProcessID == dwCurentProcessId) {

wcout << "\n\nth32ThreadID = " << entry.th32ThreadID;

wcout << "\nth32OwnerProcessID = " << entry.th32OwnerProcessID;

wcout << "\ncntUsage = " << entry.cntUsage;

wcout << "\ntpBasePri = " << entry.tpBasePri;

wcout << "\ntpDeltaPri = " << entry.tpDeltaPri;

wcout << "\ndwFlags = " << entry.dwFlags;

}

fRepeat = Thread32Next(hSnapshot, &entry);

}

CloseHandle(hSnapshot);

}

// Инфо о всех модулях системы и их атрибуты

void ShowModules() {

setlocale(LC\_ALL, "rus");

wcout.imbue(locale("rus\_rus.1251"));

HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS\_SNAPMODULE, 0);

wcout << "\nSnapshot ID = " << hSnapshot;

MODULEENTRY32 entry; entry.dwSize = sizeof(MODULEENTRY32);

BOOL fRepeat = Module32First(hSnapshot, &entry);

while (fRepeat) {

wcout << "\n\nth32ModuleID = " << entry.th32ModuleID;

wcout << "\nth32ProcessID = " << entry.th32ProcessID;

wcout << "\nGlblcntUsage = " << entry.GlblcntUsage;

wcout << "\nProccntUsage = " << entry.ProccntUsage;

wcout << "\nmodBaseAddr = " << entry.modBaseAddr;

wcout << "\nmodBaseSize = " << entry.modBaseSize;

wcout << "\nhModule = " << entry.hModule;

wcout << "\nszModule = " << entry.szModule;

wcout << "\nszExePath = " << entry.szExePath;

fRepeat = Module32Next(hSnapshot, &entry);

}

CloseHandle(hSnapshot);

}

// Инфо о загруженных драйверов устройств

void ShowDrivers() {

DWORD cbNeeded;

HMODULE hModules[1024];

// Получаем список загруженных модулей (включая драйверы) для текущего процесса

EnumProcessModules(GetCurrentProcess(), hModules, sizeof(hModules), &cbNeeded);

int numDrivers = cbNeeded / sizeof(HMODULE);

for (int i = 0; i < numDrivers; i++) {

TCHAR szDriverPath[MAX\_PATH];

if (GetModuleFileNameEx(GetCurrentProcess(), hModules[i], szDriverPath, MAX\_PATH)) {

std::wcout << L"Драйвер " << i + 1 << L": " << szDriverPath << std::endl;

}

}

}

int main() {

setlocale(LC\_ALL, "rus");

int a = -1;

while (a != 0) {

cout << "1. Получить Name, FName, Handle" << endl

<< "2. Получить FName и Handle по Name" << endl

<< "3. Получить Name и Handle по FName" << endl

<< "4. Получить Name и FName по Handle" << endl

<< "5. Получить идентификатор, псевдодескриптор, дескриптор и копию дескриптора текущего процесса" << endl

<< "6. Получить список всех процессов" << endl

<< "7. Получить список всех потоков" << endl

<< "8. Получить список всех модулей" << endl

<< "9. Получить список загруженных драйверов" << endl

<< "0. Выход" << endl << endl;

cout << "> ";

cin >> a;

switch (a) {

case 0: {

break; }

case 1: {

getHandelNameFName();

cout << endl; break; }

case 2: {

getHandleFName();

cout << endl; break; }

case 3: {

getHandleName();

cout << endl; break; }

case 4: {

getNameFName();

cout << endl; break; }

case 5: {

Handels();

cout << endl; break; }

case 6: {

cout <<\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Процессы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

ShowProcesses();

cout << endl << endl; break; }

case 7: {

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Потоки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

ShowThreads();

cout << endl << endl; break; }

case 8: {

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Модули\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

ShowModules();

cout << endl << endl; break; }

case 9: {

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Драйверы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

ShowDrivers();

cout << endl << endl; break; }

}

}

return 0;

}

# Результаты работы

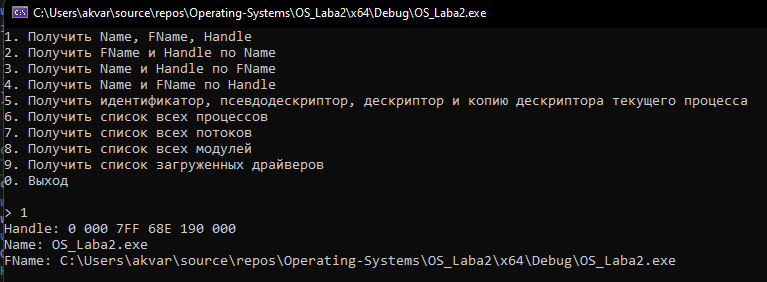
Выведем имя, полное имя и дескриптор текущего процеса.

Рисунок 1 – вывод информации

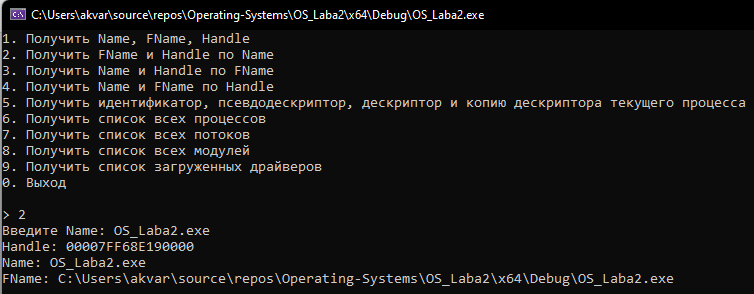
****Выведем полное имя и дескриптор, указав имя текущего процесса

Рисунок 2 – Получение дескриптора и полного имени

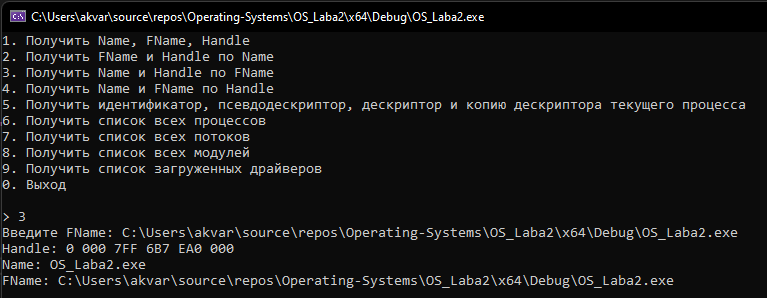
Выведем имя и дескриптор, указав только полное имя.

Рисунок 3 – получение имени и дескриптора

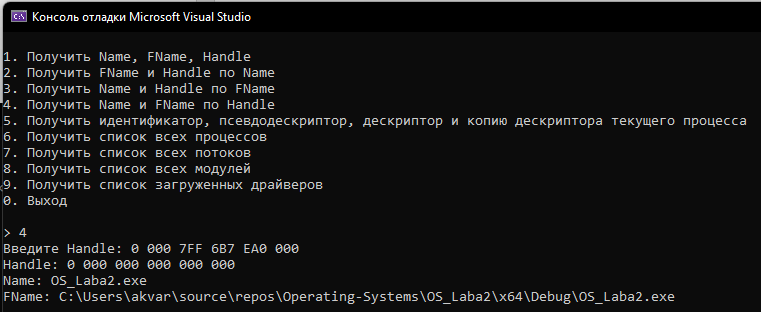
Выведем имя и полное имя, указав только дескриптор.

Рисунок 4 – Получение имени и полного имени

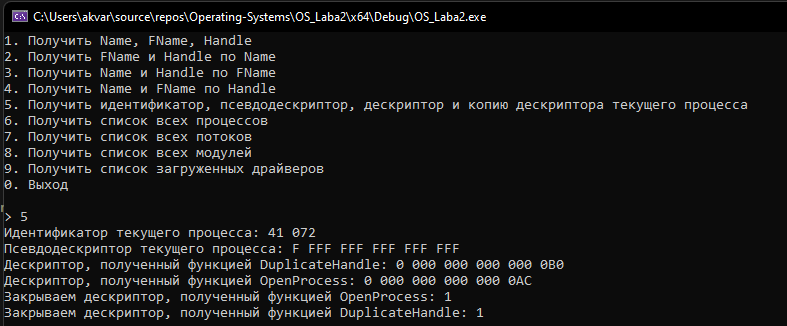
Получим идентификатор, псевдодескриптор, дескриптор и копию дескриптора текущего процесса.

Рисунок 5 – Получение информации

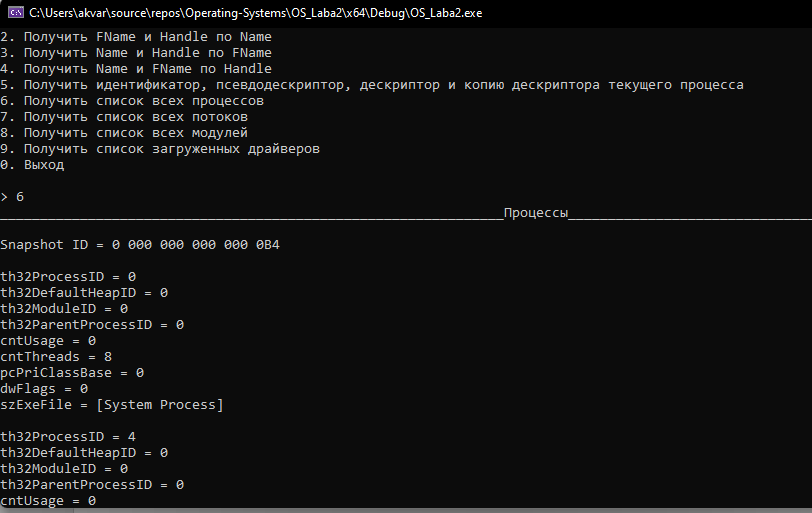
Получим список всех процессов на компьютере.

Рисунок 6 – список всех процессов

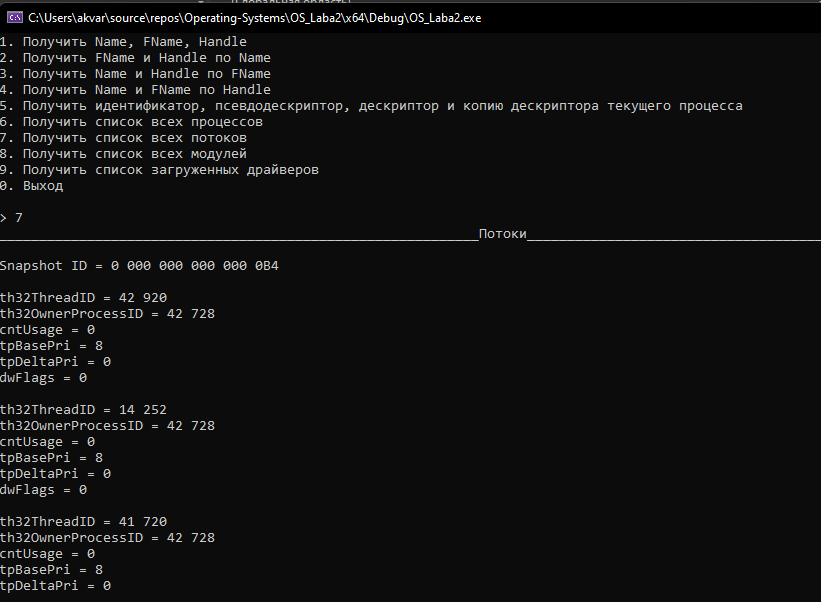
Получим список всех потоков на компьютере.

Рисунок 7 – список всех потоков

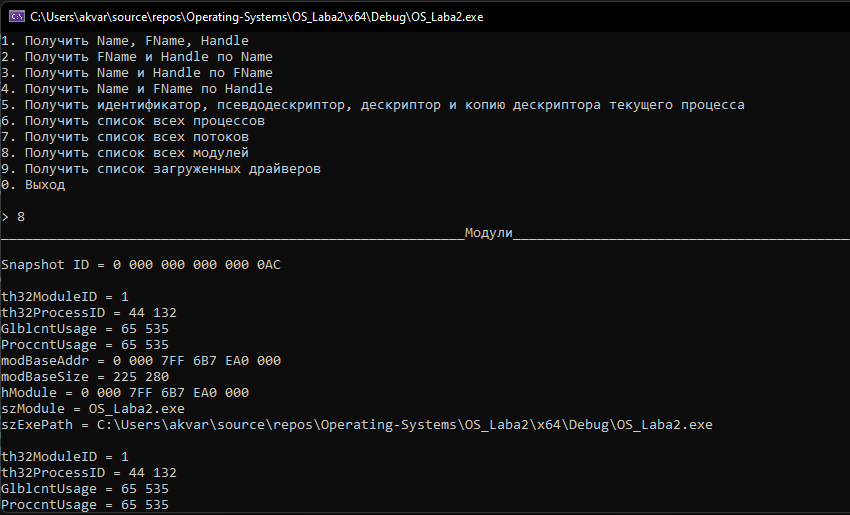
Получим список всех установленных модулей на компьютере.

Рисунок 8 – список всех модулей

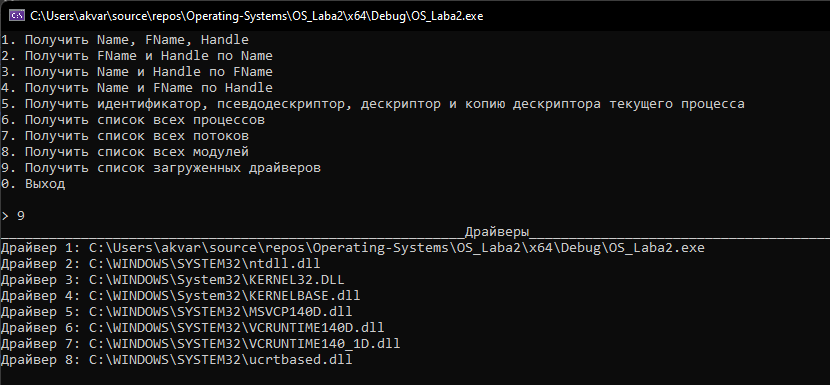
Получим список всех загруженных драйверов.

Рисунок 9 – список загруженных драйверов

# Вывод

Я получила практические навыки использования Win32 API для исследования процессов.