МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №5

з дисципліни

«Операційні системи»

Виконала:

Студентка
групи КН-214

Олескевич Софія

Викладач:

Кривенчук Ю.П.

Львів 2020

Тема. Робота з динамічними бібліотеками в ОС Windows

Мета. Ознайомитися з динамічно-зв’язувальними бібліотеками (Dynamic-link library) в ОС Windows. Навчитися реалізовувати динамічно-зв’язувальні бібліотеки.

Завдання

1. Реалізувати лабораторну роботу №4 у вигляді динамічно-зв’язувальної бібліотеки.

2. Запустити створену бібліотеку з командної стрічки (cmd.exe) за допомогою rundll32.exe.

3. Створити окрему програму і реалізувати статичний зв'язок між програмою та бібліотекою.

4. Створити окрему програму і реалізувати динамічний зв'язок між програмою та бібліотекою.

5. Експортувати головну функцію бібліотеки під іншим іменем.

6. Результати виконання роботи відобразити у звіті.

Реалізація (варіант 19)

DLL2.dll

```
// MathLibrary.cpp : Defines the exported functions for the DLL.
#include "pch.h"
#include <Windows.h>
#include <thread>
#include <iostream>
#include <set>
#include <string>
#include <algorithm>
#include <iomanip>
#include <conio.h>
#include <stdio.h>
#include <locale>
#include <vector>
#include <tchar.h>
#include <atlstr.h>
#include <utility>
#include <limits.h>
#include "Header.h"
using namespace std;
WIN32_FIND_DATA FindFileData, FindFileData1;
DWORD nFileSizeLow;
HANDLE hf, hf1;
```

```
HANDLE Sem;
SYSTEMTIME cm;
vector <string> files;
vector <long int> files_size;
vector <string> res;
int size0, per_process;
struct PARAMETERS { int from, to; };

void find_(int from) {
    WaitForSingleObject(Sem, INFINITE);
    cout << "Thread with ID " << this_thread::get_id() << " has started" << endl;
    ReleaseSemaphore(Sem, 1, NULL);
    int to = from + per_process - 1;
    for (int i = from; i <= to; i++) {
        bool is_file = false;
        // cout << "OBROBKA" << from << endl;
        for (int j = 0; j < files[i].length(); j++) {
            if (files[i][j] == '.') { is_file = true; }
        }

        WaitForSingleObject(Sem, INFINITE);
        if (files_size[i] == size0) {
            cout << files[i] << endl;
        }
        ReleaseSemaphore(Sem, 1, NULL);
        //Sleep(500);
    }
    WaitForSingleObject(Sem, INFINITE);
    cout << "Process with ID " << this_thread::get_id() << " has ended!" << endl;
    ReleaseSemaphore(Sem, 1, NULL);
    //Sleep(500000);
}
void MainF() {

    DWORD ID = 0;
    int cnt = 0, from, to;
    pair <int, int> p;
    static PARAMETERS par;
    hf = FindFirstFile("C:\\files\\*", &FindFileData);
    if (hf != INVALID_HANDLE_VALUE) {
        do {
            bool is_true = true;
            files.push_back(FindFileData.cFileName);
            files_size.push_back(FindFileData.nFileSizeLow);
            for (int j = 0; j < files[cnt].length(); j++) {
                if (files[cnt][j] == '.') { is_true = false; break; }
            }
            cnt++;
            if (is_true) {
                string s = "C:\\" + files[cnt - 1] + "\\*";
                const char* path = s.c_str();
                hf1 = FindFirstFile(path, &FindFileData1);
                if (hf != INVALID_HANDLE_VALUE) {
                    do {
                        files.push_back(FindFileData1.cFileName);
                        files_size.push_back(FindFileData1.nFileSizeLow);
                        cnt++;
                    } while (FindNextFile(hf1, &FindFileData1) != 0);
                    FindClose(hf1);
                }
            }
        } while (FindNextFile(hf, &FindFileData) != 0);
```

```
        FindClose(hf);
    }
    cout << "Number of files: " << endl << cnt << endl;
    cout << "Enter size of file: " << endl;
    cin >> size0;
    cout << "Enter count of processes: " << endl;
    int n;
    cin >> n;
    cout << endl;
    per_process = cnt / n;
    HANDLE* hThreads = new HANDLE[n];
    Sem = CreateSemaphore(NULL, 1, 1, 0);
    for (int i = 0; i < n; i++) {
        from = (i * per_process);
        //cout << from << " ";
        p = make_pair(from, to);
        //cout << "from: " << from << endl << "to:" << to << endl;
        hThreads[i] = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)find_,
(LPVOID)from, 0, &ID);
    }
    int choice;
    _getch();
    do {
        cout << "Would you like to set the priority to one of the thread? " << endl <<
"1. Yes" << endl << "2. Exit" << endl;
        cin >> choice;
        if (choice == 1) {
            int thread;
            cout << "Input the number of thread: ";
            cin >> thread;
            cout << "1. Highest priority " << endl;
            cout << "2. Above normal priority " << endl;
            cout << "3. Normal priority " << endl;
            cout << "4. Below normal priority " << endl;
            cout << "5. Idle priority " << endl;
            int ch; cin >> ch;
            switch (ch) {
            case 1: {SetThreadPriority(hThreads[thread - 1], THREAD_PRIORITY_HIGHEST);
break; }
            case 2: {SetThreadPriority(hThreads[thread - 1],
THREAD_PRIORITY_ABOVE_NORMAL); break; }
            case 3: {SetThreadPriority(hThreads[thread - 1], THREAD_PRIORITY_NORMAL);
break; }
            case 4: {SetThreadPriority(hThreads[thread - 1],
THREAD_PRIORITY_BELOW_NORMAL); break; }
            case 5: {SetThreadPriority(hThreads[thread - 1], THREAD_PRIORITY_IDLE);
break; }
            }
        }
    } while (choice == 1);
    cout << "All threads ended!" << endl << endl;
    //for (int i = 0; i < n; i++) cout << "Thread " << this_thread::get_id()<< "
priority: " << GetThreadPriority(hThreads[i]) << endl;

    for (int i = 0; i < n; i++) {
        CloseHandle(hThreads[i]);
    }
}

void CALLBACK console(HWND hWnd, HINSTANCE hInst, LPSTR  lpszCmdLine, int nCmdShow) {
    AllocConsole();
    freopen("CONIN$", "r", stdin);
```

```
    freopen("CONOUT$", "w", stdout);
    freopen("CONIN$", "r", stderr);
    MainF();
}
```

Header.h

```
// MathLibrary.h - Contains declarations of math functions
#pragma once
#ifdef Header_EXPORTS
#define Header_API __declspec(dllexport)
#else
#define Header_API __declspec(dllimport)
#endif
extern "C" Header_API void MainF();
extern "C" Header_API void main_function();
extern "C" Header_API void CALLBACK console(HWND hWnd, HINSTANCE hInst, LPSTR
lpszCmdLine, int nCmdShow);
//extern "C" Header_API void find_();
```

Dll2.def

```
LIBRARY DLL2.DLL
EXPORTS
	main_function = MainF
	console
```

Dynamic

```
// os555.cpp : This file contains the 'main' function. Program execution begins and
ends there.
// dynamic
#include <windows.h>
#include <iostream>
#include <limits.h>
using namespace std;
typedef int(CALLBACK* LPFNDLLFUNC1)();
int main()
{
	HINSTANCE hDLL;
	LPFNDLLFUNC1 lpfnDllFunc1;
	HRESULT hrReturnVal;

	hDLL =
LoadLibrary("C:\\Users\\soles\\VisualProjects\\repos\\Dll2\\Debug\\Dll2.dll");

	if (hDLL != NULL)
	{
		lpfnDllFunc1 = (LPFNDLLFUNC1)GetProcAddress(hDLL, "main_function");

		if (NULL != lpfnDllFunc1)
			hrReturnVal = lpfnDllFunc1();
		else
			hrReturnVal = ERROR_DELAY_LOAD_FAILED;

	}
}
```

Static

```
// os5.3.cpp : This file contains the 'main' function. Program execution begins and
ends there.
//static
#include <iostream>
#include <windows.h>
#include "Header.h"
int main()
{
    main_function();
    return 0;
}
```

Висновок:

В ході роботи над цією лабораторною я навчилась створювати власну бібліотеку, реалізовувати статичний та динамічний зв’язки між програмою та бібліотекою. Також реалізувала експорт перевизначивши ім’я.