**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**. Файли, що відображаються в пам'ять в ОС Windows

**Мета.** Ознайомитися з відображенням файлів в оперативну пам’ять в ОС Windows.

Навчитися реалізовувати відображення файлів в оперативну пам’ять.

**Завдання.**

1. Реалізувати відображення файлу в оперативну пам'ять.

2. Необхідно модифікувати лабораторну роботу №4 таким чином, щоб всі результати роботи програми записувалися через відображення файлів в оперативній пам’яті.

3. Виконати завдання, з використанням цього відображення.

4. Результати виконання роботи відобразити у звіті.

Код:

```
#define _CRT_SECURE_NO_WARNINGS
#define i i
#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>
using namespace std;
WIN32_FIND_DATA FindFileData, FindFileData1;
DWORD nFileSizeLow, ID;
HANDLE Sem, hf, hf1, hFile;
vector <string> files;
vector <long int> files_size;
vector <string> rez;
int size0, per_process, symbol=0;
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;
        for (int j = 0; j < files[i].length(); j++) {
            if (files[i][j] == '.') { is_file = true; }
        }
        WaitForSingleObject(Sem, INFINITE);
        if (files_size[i] == size0)
        {
```

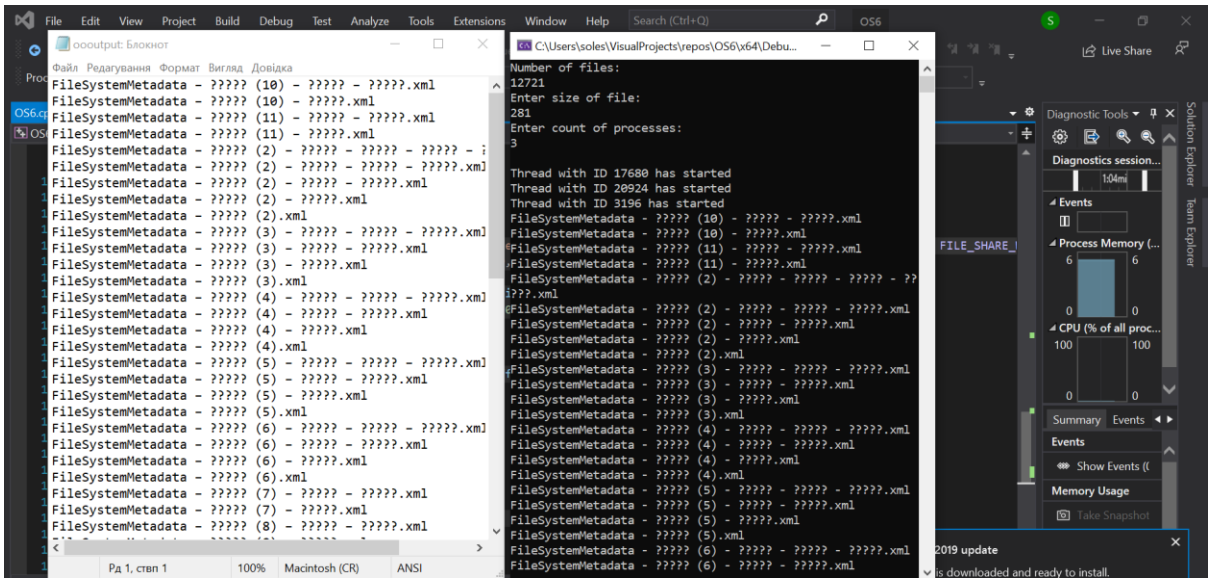
```
            symbol += files[i].length() + 1;
            rez.push_back(files[i]);
            cout << files[i] << endl;
        }
        ReleaseSemaphore(Sem, 1, NULL);
    }
    WaitForSingleObject(Sem, INFINITE);
    cout << "Process with ID " << this_thread::get_id() << " has ended!" << endl;
ReleaseSemaphore(Sem, 1, NULL);
}
int main(int argv, char* argc[]) {
    setlocale(LC_CTYPE, "");
    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;
    HANDLE* hThreads = new HANDLE[n];
    per_process = cnt / n;
    Sem = CreateSemaphore(NULL, 1, 1, 0);
    for (int i = 0; i < n; i++) {
        from = (i * per_process);
        p = make_pair(from, to);
        hThreads[i] = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)find_,
(LPVOID)from, 0, &ID);
    }
    cout << "All threads ended!" << endl << endl;
    cout << "REZ VECTOR SIZE " << rez.size() << endl;
    HANDLE hMap;
    LPBYTE pData = NULL;
    HANDLE hFile =
CreateFile("C:\\Users\\soles\\VisualProjects\\repos\\OS6\\oooutput.txt", GENERIC_READ
| GENERIC_WRITE, FILE_SHARE_WRITE, nullptr, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL,
nullptr);
```

```
    hMap = CreateFileMapping(hFile, NULL, PAGE_READWRITE, 0, symbol, NULL);
    if (hMap) {
    DWORD dwSize = GetFileSize(hFile, NULL); CloseHandle(hFile);
    pData = (LPBYTE)MapViewOfFile(hMap, FILE_MAP_WRITE, 0, 0, 0);
    if (pData) {
        int offset = 0;
        for (int z = 0; z < rez.size(); z++, offset++) {
            for (int zz = 0; zz < rez[z].length(); zz++, offset++) {
                *(pData + offset) = rez[z][zz];

            }
            *(pData + offset) = '\r\n';
        }
    }
    UnmapViewOfFile(pData); } CloseHandle(hMap);

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

Результати виконання:




Висновок: В ході роботи над цією лабораторною, я зуміла реалізувати відображення файлів в оперативну пам’ять в ОС Windows.