**Міністерство освіти і науки України**

**НТУУ «Київський політехнічний інститут»**

**Фізико-технічний інститут**

**Кафедра інформаційної безпеки**

**Системне програмування**

Комп’ютерний практикум № 2

**РОБОТА З ФАЙЛАМИ**

**Варіант 8**

**Виконав:**

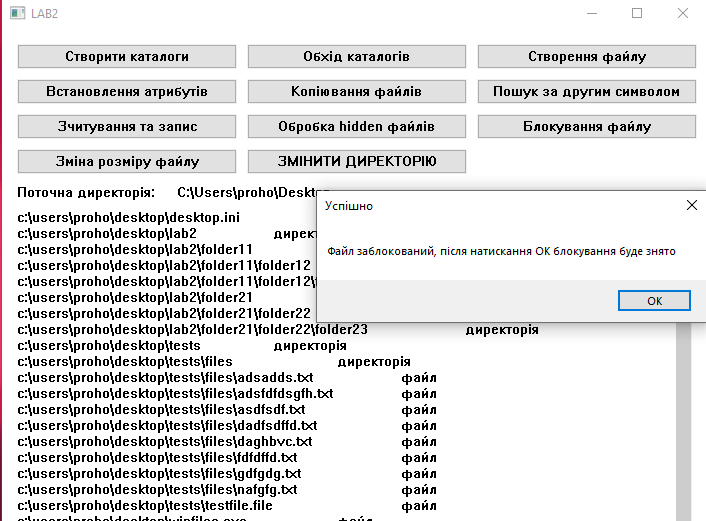
Студент   
 2 курсу ФТІ  
групи ФБ-95  
Прохоренко Ярослав

***Функціонал:***

1. Створення системи каталогів в поточному каталозі
2. Обхід каталогів
3. Створення файлу в поточній директорії
4. Встановлення атрибуту на вибраний файл
5. Копіювання будь-яких файлів з будь-якого каталогу в поточний
6. Зчитування вибраного файлу, та запис в кінець у цей же файл
7. Пошук файлів за другим символом
8. Пошук файлів за першим символом «а», та за атрибутом «хідден», копіювання знайдених файлів в інший каталог
9. Блокування вибраного файлу
10. Видалення останнього рядка файлу шляхом зміни розміру
11. Зміна поточного каталогу

***Скріншоти:***





***Код:***

#pragma comment(linker,"\"/manifestdependency:type='win32' \

name='Microsoft.Windows.Common-Controls' version='6.0.0.0' \

processorArchitecture='\*' publicKeyToken='6595b64144ccf1df' language='\*'\"")

#include "resource.h"

#include <windows.h>

#include <tchar.h>

#include <stdlib.h>

#include <string>

#include <iostream>

#include <sstream>

#include <shlobj.h>

#include <malloc.h>

#include <vector>

#include <iomanip>

#define ID\_BUTTON1 1

#define ID\_BUTTON2 2

#define ID\_BUTTON3 3

#define ID\_BUTTON4 4

#define ID\_BUTTON5 5

#define ID\_BUTTON6 6

#define ID\_BUTTON7 7

#define ID\_BUTTON8 8

#define ID\_BUTTON9 9

#define ID\_BUTTON10 10

#define ID\_BUTTON11 11

#define ID\_BUTTON12 12

#define LOG\_TEXT 13

#define BUFF 1024

HINSTANCE hInst;

TCHAR logSize[**255**];

std::wstring searchLogSize;

HWND logs;

HWND edit;

wchar\_t symbol[**4**];

std::vector<std::wstring> FileNames;

OVERLAPPED olf = { **0** };

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

std::wstring szPATH = L"D:**\\**";

std::wstring PATHs[] = {

L"**\\**lab2**\\**" ,

L"**\\**lab2**\\**Folder11",

L"**\\**lab2**\\**Folder11**\\**Folder12",

L"**\\**lab2**\\**Folder11**\\**Folder12**\\**Folder13",

L"**\\**lab2**\\**Folder21",

L"**\\**lab2**\\**Folder21**\\**Folder22",

L"**\\**lab2**\\**Folder21**\\**Folder22**\\**Folder23" };

void DetourToLog(LPWSTR path, std::wstring mask);

void AppendText(HWND hEditWnd, LPCTSTR Text);

INT\_PTR CALLBACK DlgProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);

LPCSTR ReadFromFile(PWSTR path);

DWORD WINAPI FindThreadFunc(LPVOID param);

static int CALLBACK BrowseCallbackProc(HWND hwnd, UINT uMsg, LPARAM lParam, LPARAM lpData)

{

**if** (uMsg == BFFM\_INITIALIZED)

{

std::string tmp = (const char\*)lpData;

std::cout << "path: " << tmp << std::endl;

SendMessage(hwnd, BFFM\_SETSELECTION, TRUE, lpData);

}

**return** **0**;

}

std::wstring BrowseFolder(std::wstring saved\_path)

{

TCHAR path[MAX\_PATH];

const wchar\_t\* path\_param = saved\_path.c\_str();

BROWSEINFO bi = { **0** };

bi.lpszTitle = L"Виберіть директорію...";

bi.ulFlags = BIF\_RETURNONLYFSDIRS | BIF\_NEWDIALOGSTYLE;

bi.lpfn = BrowseCallbackProc;

bi.lParam = (LPARAM)path\_param;

LPITEMIDLIST pidl = SHBrowseForFolder(&bi);

**if** (pidl != **0**)

{

//get the name of the folder **and** put it **in** path

SHGetPathFromIDList(pidl, path);

//free memory used

IMalloc\* imalloc = **0**;

**if** (SUCCEEDED(SHGetMalloc(&imalloc)))

{

imalloc->Free(pidl);

imalloc->Release();

}

**return** path;

}

**return** szPATH;

}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

static TCHAR szWindowClass[] = L"DesktopApp";

static TCHAR szTitle[] = L"LAB2";

WNDCLASSEX wn;

wn.cbSize = sizeof(WNDCLASSEX);

wn.style = CS\_HREDRAW | CS\_VREDRAW;

wn.lpfnWndProc = WndProc;

wn.cbClsExtra = **0**;

wn.cbWndExtra = **0**;

wn.hInstance = hInstance;

wn.hIcon = LoadIcon(hInstance, IDI\_APPLICATION);

wn.hCursor = LoadCursor(NULL, IDC\_ARROW);

wn.hbrBackground = CreateSolidBrush(RGB(**255**, **255**, **255**));

wn.lpszMenuName = NULL;

wn.lpszClassName = szWindowClass;

wn.hIconSm = LoadIcon(wn.hInstance, IDI\_APPLICATION);

**if** (!RegisterClassEx(&wn))

{

MessageBoxA(NULL, "Call to RegisterClassEx failed!", "Error", NULL);

**return** **1**;

}

hInst = hInstance;

HWND hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW | WS\_THICKFRAME, CW\_USEDEFAULT, CW\_USEDEFAULT, **720**,**640**, NULL, NULL, hInstance, NULL);

**if** (!hWnd)

{

MessageBoxA(NULL, "Call to CreateWindow failed!","Error", NULL);

**return** **1**;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

MSG msg;

**while** (GetMessage(&msg, NULL, **0**, **0**))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

**return** static\_cast<int>(msg.wParam);

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {

HDC hdc;

PAINTSTRUCT ps;

TCHAR label1[] = L"Поточна директорія: ";

switch (message)

{

case WM\_CREATE:

{

HWND button1 = CreateWindow(L"button", L"Створити каталоги", WS\_CHILD | WS\_VISIBLE, **15**, **15**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON1, **0**, **0**);

HWND button2 = CreateWindow(L"button", L"Обхід каталогів", WS\_CHILD | WS\_VISIBLE, **245**, **15**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON2, **0**, **0**);

HWND button3 = CreateWindow(L"button", L"Створення файлу", WS\_CHILD | WS\_VISIBLE, **475**, **15**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON3, **0**, **0**);

HWND button4 = CreateWindow(L"button", L"Встановлення атрибутів", WS\_CHILD | WS\_VISIBLE, **15**, **50**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON4, **0**, **0**);

HWND button5 = CreateWindow(L"button", L"Копіювання файлів", WS\_CHILD | WS\_VISIBLE, **245**, **50**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON5, **0**, **0**);

HWND button6 = CreateWindow(L"button", L"Пошук за другим символом", WS\_CHILD | WS\_VISIBLE, **475**, **50**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON6, **0**, **0**);

HWND button7 = CreateWindow(L"button", L"Зчитування та запис", WS\_CHILD | WS\_VISIBLE, **15**, **85**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON7, **0**, **0**);

HWND button8 = CreateWindow(L"button", L"Обробка hidden файлів", WS\_CHILD | WS\_VISIBLE, **245**, **85**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON8, **0**, **0**);

HWND button9 = CreateWindow(L"button", L"Блокування файлу", WS\_CHILD | WS\_VISIBLE, **475**, **85**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON9, **0**, **0**);

HWND button11 = CreateWindow(L"button", L"Зміна розміру файлу", WS\_CHILD | WS\_VISIBLE, **15**, **120**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON11, **0**, **0**);

HWND button12 = CreateWindow(L"button", L"ЗМІНИТИ ДИРЕКТОРІЮ", WS\_CHILD | WS\_VISIBLE, **245**, **120**, **220**, **25**, hWnd, (HMENU)ID\_BUTTON12, **0**, **0**);

logs = CreateWindowW(L"EDIT", L"", WS\_CHILD | WS\_VISIBLE | ES\_MULTILINE | WS\_VSCROLL | EM\_SCROLLCARET, **15**, **180**, **675**, **400**, hWnd, (HMENU)LOG\_TEXT, **0**, **0**);

TCHAR logSize[**255**] = { **0** };

SendDlgItemMessage(logs, EM\_SETREADONLY, TRUE, **0**, **0**);

}**break**;

case WM\_PAINT:

{

hdc = BeginPaint(hWnd, &ps);

TextOut(hdc, **15**, **155**, label1, \_countof(label1));

const wchar\_t\* wPATH = szPATH.c\_str();

TextOut(hdc, **175**, **155**, wPATH, szPATH.size());

EndPaint(hWnd, &ps);

} **break**;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case ID\_BUTTON1:

**if** (HIWORD(wParam) == BN\_CLICKED)

{

**for** (int i = **0**; i < \_countof(PATHs); i++)

{

**if** (CreateDirectory((LPCWSTR)(szPATH + PATHs[i]).c\_str(), NULL))

{

AppendText(logs, L"**\r\n**Успішно створено");

AppendText(logs, L"**\t\t**");

AppendText(logs, PATHs[i].c\_str());

}

}

}

**break**;

case ID\_BUTTON2:

**if** (HIWORD(wParam) == BN\_CLICKED)

{

AppendText(logs, L"**\r\n**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ПОЧАТОК ОБХОДУ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***\r\n**");

DetourToLog((LPWSTR)(szPATH.c\_str()), L"\*.\*");

AppendText(logs, L"**\r\n**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ОБХІД ЗАВЕРШЕНО \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***\r\n**");

}

**break**;

case ID\_BUTTON3:

**if** (HIWORD(wParam) == BN\_CLICKED)

{

std::wstring fileName = L"**\\**File22.file";

std::wstring szNamePath = szPATH + fileName;

HANDLE file = CreateFile(szNamePath.c\_str(), GENERIC\_READ, **0**, NULL, CREATE\_NEW, FILE\_ATTRIBUTE\_NORMAL, NULL);

AppendText(logs, L"**\r\n**Успішно створено");

AppendText(logs, L"**\t\t**");

AppendText(logs, fileName.c\_str());

CloseHandle(file);

}

**break**;

case ID\_BUTTON4:

{

std::wstring fileName = L"**\\**File22.file";

std::wstring szNamePath = szPATH + PATHs[**5**] + fileName;

**if** (HIWORD(wParam) == BN\_CLICKED)

{

OPENFILENAME ofn;

wchar\_t nameFile[**260**] = { **0** };

wchar\_t szFile[**260**] = { **0** };

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFile = szFile;

ofn.nMaxFile = sizeof(szFile);

ofn.nFilterIndex = **1**;

ofn.lpstrFileTitle = nameFile;

ofn.nMaxFileTitle = **256**;

ofn.lpstrInitialDir = NULL;

ofn.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST | OFN\_EXPLORER;

GetOpenFileName(&ofn);

wchar\_t full[MAX\_PATH];

DWORD attributes = GetFileAttributes(ofn.lpstrFile);

**if** (!(attributes & FILE\_ATTRIBUTE\_SYSTEM))

{

SetFileAttributes(ofn.lpstrFile, FILE\_ATTRIBUTE\_SYSTEM);

AppendText(logs, L"**\r\n**System атрибут був доданий до файлу**\t**");

AppendText(logs, ofn.lpstrFile);

AppendText(logs, L"**\r\n**");

}

**else** {

AppendText(logs, L"**\r\n**Виникла помилка! Атрибут вже встановлений, або файл не створений**\r\n**");

}

WIN32\_FIND\_DATA wfd = { **0** };

FindFirstFile(ofn.lpstrFile, &wfd);

AppendText(logs, L"Ім'я файлу: ");

AppendText(logs, wfd.cFileName);

AppendText(logs, L"**\r\n**Встанолвені атрибути: ");

**if** (wfd.dwFileAttributes & FILE\_ATTRIBUTE\_SYSTEM)

{

AppendText(logs, L"System ");

}

**if** (wfd.dwFileAttributes & FILE\_ATTRIBUTE\_HIDDEN)

{

AppendText(logs, L"Hidden ");

}

**if** (wfd.dwFileAttributes & FILE\_ATTRIBUTE\_ARCHIVE)

{

AppendText(logs, L"ARCHIVE ");

}

}

}**break**;

case ID\_BUTTON5:

{

FileNames.clear();

OPENFILENAME ofn;

wchar\_t nameFile[**260**] = { **0** };

wchar\_t szFile[**260**] = { **0** };

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFile = szFile;

ofn.nMaxFile = sizeof(szFile);

ofn.nFilterIndex = **1**;

ofn.lpstrFileTitle = nameFile;

ofn.nMaxFileTitle = **256**;

ofn.lpstrInitialDir = NULL;

ofn.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST | OFN\_ALLOWMULTISELECT | OFN\_EXPLORER;

**if** (GetOpenFileName(&ofn) == **0**)//не выбран файл

{

**return** **0**;

}

**else** //выбран файл

{

int nOffset = ofn.nFileOffset;

**if** (nOffset > lstrlen(ofn.lpstrFile))

{//более **1** файла

**while** (ofn.lpstrFile[nOffset])

{

FileNames.push\_back(ofn.lpstrFile + nOffset);

nOffset = nOffset + wcslen(ofn.lpstrFile + nOffset) + **1**;

}

}

}

wchar\_t full[MAX\_PATH];

std::wstring to;

**for** (int i = **0**; i < FileNames.size(); i++)

{

GetFullPathName(FileNames[i].c\_str(), MAX\_PATH, (LPWSTR)full, NULL);

to = szPATH + L"**\\**" + FileNames[i];

**if** (CopyFile(full, to.c\_str(), NULL))

{

AppendText(logs, L"**\r\n**успішно скопійовано**\t\t**");

AppendText(logs, full);

AppendText(logs, L"**\r\n**");

}

**else** {

AppendText(logs, L"помилка при копіюванні**\t\t**");

AppendText(logs, full);

AppendText(logs, L"**\r\n**");

}

}

}**break**;

case ID\_BUTTON6:

{

**if** (HIWORD(wParam) == BN\_CLICKED)

{

HWND hWndDialog = CreateDialog(hInst, MAKEINTRESOURCE(IDD\_DIALOG1), hWnd, (DLGPROC)DlgProc);

ShowWindow(hWndDialog, SW\_SHOW);

}

}**break**;

case ID\_BUTTON7:

{

**if** (HIWORD(wParam) == BN\_CLICKED)

{

HRESULT hr = CoInitializeEx(NULL, COINIT\_APARTMENTTHREADED |

COINIT\_DISABLE\_OLE1DDE);

**if** (SUCCEEDED(hr))

{

IFileOpenDialog\* pFileOpen;

// Create the FileOpenDialog object.

hr = CoCreateInstance(CLSID\_FileOpenDialog, NULL, CLSCTX\_ALL,

IID\_IFileOpenDialog, reinterpret\_cast<void\*\*>(&pFileOpen));

**if** (SUCCEEDED(hr))

{

// Show the Open dialog box.

hr = pFileOpen->Show(NULL);

// Get the file name **from** **the** **dialog** **box.**

**if** (SUCCEEDED(hr))

{

IShellItem\* pItem;

hr = pFileOpen->GetResult(&pItem);

**if** (SUCCEEDED(hr))

{

PWSTR pszFilePath;

hr = pItem->GetDisplayName(SIGDN\_FILESYSPATH, &pszFilePath);

**if** (SUCCEEDED(hr))

{

SetWindowTextA(logs, ReadFromFile(pszFilePath));

MessageBoxW(NULL, L"Файл був зчитаний в лог. Після натискання ОК до нього будуть дозаписані рядки.", L"Успішно", MB\_OK);

HANDLE FileHandle;

FileHandle = CreateFile(pszFilePath, GENERIC\_WRITE, **0**, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, **0**);

const char\* data = "**\r\n**ФБ-95 Прохоренко";

DWORD dwBytesToWrite = strlen(data);

DWORD dwBytesWritten;

WriteFile(FileHandle, data, dwBytesToWrite, &dwBytesWritten, &olf);

CloseHandle(FileHandle);

SetWindowTextA(logs, ReadFromFile(pszFilePath));

CoTaskMemFree(pszFilePath);

}

pItem->Release();

}

}

pFileOpen->Release();

}

CoUninitialize();

}

}

}**break**;

case ID\_BUTTON8:

{

**if** (HIWORD(wParam) == BN\_CLICKED)

{

CreateThread(NULL, **0**, FindThreadFunc, NULL, **0**, **0**);

//WaitForSingleObject(FindThread, INFINITE);

}

}**break**;

case ID\_BUTTON9:

{

**if** (HIWORD(wParam) == BN\_CLICKED)

{

OPENFILENAME ofn;

wchar\_t nameFile[**260**] = { **0** };

wchar\_t szFile[**260**] = { **0** };

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFile = szFile;

ofn.nMaxFile = sizeof(szFile);

ofn.nFilterIndex = **1**;

ofn.lpstrFileTitle = nameFile;

ofn.nMaxFileTitle = **256**;

ofn.lpstrInitialDir = NULL;

ofn.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST | OFN\_EXPLORER;

GetOpenFileName(&ofn);

HANDLE FileHandle;

FileHandle = CreateFile(ofn.lpstrFile, GENERIC\_READ, **0**, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, **0**);

DWORD lpNumberOfBytesRead;

DWORD dwFileSize = GetFileSize(FileHandle, NULL);

WCHAR\* lpszText = new WCHAR[dwFileSize / **2** + **1**];

lpszText[dwFileSize / **2**] = **0**;

ReadFile(FileHandle, lpszText, dwFileSize, &lpNumberOfBytesRead, NULL);

DWORD dwPos = SetFilePointer(FileHandle, **0**, NULL, FILE\_END);

LockFile(FileHandle, dwPos, **0**, lpNumberOfBytesRead, **0**);

MessageBoxW(NULL, L"Файл заблокований, після натискання ОК блокування буде знято", L"Успішно", MB\_OK);

CloseHandle(FileHandle);

}

}**break**;

case ID\_BUTTON11:

{

**if** (HIWORD(wParam) == BN\_CLICKED)

{

OPENFILENAME ofn;

wchar\_t nameFile[**260**] = { **0** };

wchar\_t szFile[**260**] = { **0** };

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFile = szFile;

ofn.nMaxFile = sizeof(szFile);

ofn.nFilterIndex = **1**;

ofn.lpstrFileTitle = nameFile;

ofn.nMaxFileTitle = **256**;

ofn.lpstrInitialDir = NULL;

ofn.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST | OFN\_EXPLORER;

GetOpenFileName(&ofn);

HANDLE FileHandle;

FileHandle = CreateFile(ofn.lpstrFile, GENERIC\_READ | GENERIC\_WRITE,

FILE\_SHARE\_WRITE | FILE\_SHARE\_READ, NULL, OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL, **0**);

DWORD BufSize = GetFileSize(FileHandle, NULL);

char\* buf = new char[**1** + BufSize];

buf[BufSize] = **0**;

DWORD dwBytes = **1**;

std::string all;

int c = **0**;

**for** (int i = **0**; i < BufSize; i++)

{

SetFilePointer(FileHandle, -i, NULL, FILE\_END);

ReadFile(FileHandle, &buf[i], sizeof(char), &dwBytes, NULL);

c++;

**if** (buf[i] == '**\n**')

{

SetFilePointer(FileHandle, -(i+**1**), NULL, FILE\_END);

SetEndOfFile(FileHandle);

**break**;

}

}

CloseHandle(FileHandle);

}

}**break**;

case ID\_BUTTON12:

**if** (HIWORD(wParam) == BN\_CLICKED)

{

std::wstring path\_ = BrowseFolder(szPATH);

szPATH = path\_;

InvalidateRect(hWnd, NULL, TRUE);

}

}

**break**;

case WM\_DESTROY:

PostQuitMessage(**0**);

**break**;

default:

**return** DefWindowProc(hWnd, message, wParam, lParam);

**break**;

}

**return** **0**;

}

DWORD WINAPI FindThreadFunc(LPVOID param)

{

WIN32\_FIND\_DATA wfd = { **0** };

SetCurrentDirectory(szPATH.c\_str());

CreateDirectory((LPCWSTR)(szPATH + L"**\\**CopiedHiddenFiles").c\_str(), NULL);

HANDLE search = FindFirstFile(L"a\*", &wfd);

int counter = **0**;

do

{

**if** (wfd.dwFileAttributes & FILE\_ATTRIBUTE\_HIDDEN)

{

wchar\_t full[MAX\_PATH];

std::wstring to;

GetFullPathName(wfd.cFileName, MAX\_PATH, (LPWSTR)full, NULL);

to = szPATH + L"**\\**CopiedHiddenFiles" + L"**\\**" + wfd.cFileName;

**if** (CopyFile(full, to.c\_str(), NULL))

{

AppendText(logs, L"**\r\n**успішно скопійовано**\t\t**");

AppendText(logs, full);

AppendText(logs, L"**\r\n**");

}

**else** {

AppendText(logs, L"**\r\n**помилка при копіюванні**\t\t**");

AppendText(logs, full);

AppendText(logs, L"**\r\n**");

}

counter++;

}

} **while** (FindNextFile(search, &wfd));

FindClose(search);

AppendText(logs, L"**\r\n**Знайдено прихованих файлів і скопійовано: ");

std::wstring result = std::to\_wstring(counter);

AppendText(logs, result.c\_str());

ExitThread(**0**);

}

LPCSTR ReadFromFile(PWSTR path)

{

HANDLE FileHandle;

FileHandle = CreateFile(path, GENERIC\_READ , **0**, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, **0**);

DWORD lpNumberOfBytesRead;

DWORD dwFileSize = GetFileSize(FileHandle, NULL);

WCHAR\* lpszText = new WCHAR[dwFileSize / **2** + **1**];

lpszText[dwFileSize / **2**] = **0**;

ReadFile(FileHandle, lpszText, dwFileSize, &lpNumberOfBytesRead, NULL);

CloseHandle(FileHandle);

olf.Offset += lpNumberOfBytesRead;

**return** (LPCSTR)lpszText;

}

void DetourToLog(LPWSTR path, std::wstring mask)

{

WIN32\_FIND\_DATA wfd = { **0** };

SetCurrentDirectory(path);

HANDLE search = FindFirstFile(mask.c\_str(), &wfd);

**if** (search == INVALID\_HANDLE\_VALUE)

**return**;

do

{

LPWSTR sizeTemp = (LPWSTR)calloc(BUFF + **1**, sizeof(WCHAR));

GetCurrentDirectory(BUFF, sizeTemp);

wcscat\_s(sizeTemp, BUFF, L"**\\**");

wcscat\_s(sizeTemp, BUFF, wfd.cFileName);

LPWSTR sizeTemp\_ = (LPWSTR)calloc(wcslen(sizeTemp) + **1**, sizeof(WCHAR));

wcscpy\_s(sizeTemp\_, wcslen(sizeTemp) + **1**, sizeTemp);

**if** (wcscmp(wfd.cFileName, L".") &&

wcscmp(wfd.cFileName, L".."))

{

wcscat\_s(sizeTemp, BUFF, L"**\t\t**");

**if** (((wfd.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY)) &&

(!(wfd.dwFileAttributes & FILE\_ATTRIBUTE\_REPARSE\_POINT)))

{

wcscat\_s(sizeTemp, BUFF, L"Директорія");

wcscat\_s(sizeTemp, BUFF, L"**\r\n**");

AppendText(logs, sizeTemp);

SendMessageA(logs, EM\_LINESCROLL, **0**, **1**);

DWORD iCount = **0**;

DetourToLog(sizeTemp\_, mask);

SetCurrentDirectory(path);

}

**else**

{

wcscat\_s(sizeTemp, BUFF, L"Файл");

wcscat\_s(sizeTemp, BUFF, L"**\r\n**");

DWORD iCount = **0**;

AppendText(logs, sizeTemp);

SendMessageA(logs, EM\_LINESCROLL, **0**, **1**);

}

}

free(sizeTemp);

free(sizeTemp\_);

} **while** (FindNextFile(search, &wfd));

FindClose(search);

}

INT\_PTR CALLBACK DlgProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_INITDIALOG:

{

edit = CreateWindowW(L"EDIT", L"a", WS\_CHILD | WS\_VISIBLE, **130**, **13**, **25**, **18**, hWnd, **0**, **0**, **0**);

}**break**;

case WM\_COMMAND:

{

switch (LOWORD(wParam))

{

case IDOK:

GetWindowText(edit, symbol, **4**);

std::wstring mask;

mask += L"?";

mask += symbol;

mask += L"\*";

DetourToLog((LPWSTR)(szPATH.c\_str()),mask);

DestroyWindow(hWnd);

**break**;

}

}**break**;

default:

**return** DefWindowProc(hWnd, message, wParam, lParam);

}

**return** **0**;

}

void AppendText(HWND hEditWnd, LPCTSTR Text)

{

int idx = GetWindowTextLength(hEditWnd);

SendMessage(hEditWnd, EM\_SETSEL, (WPARAM)idx, (LPARAM)idx);

SendMessage(hEditWnd, EM\_REPLACESEL, **0**, (LPARAM)Text);

}