Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра програмування та захисту інформації

**Звіт**

з виконаної лабораторної роботи № 7

дисципліни “ Системне програмування”

на тему

«**ПРОГРАМУВАННЯ В WIN32 АРІ В ОС WINDOWS**»

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив :

Викладач

Константинова K.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницкий- 2017

/\*\* закрытие

СloseHandle(lpHandle);

\*/

/\*\* Удаление

BOOL DeleteFile(

LPCTSTR lpFileName; // полный путь до файла

);

\*/

#include <windows.h>

#include <iostream>

using namespace std;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Отображение ошибки

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Error(string operation)

{

cerr << operation <<"operation failed.\nThe last error code: " << GetLastError() << endl;

return;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Корректное завершение

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Close(HANDLE \*hFile)

{

CloseHandle(hFile);

cout << "Operation successful" << endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Удаление файла по указанному пути

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void DeleteFiles(char \*path)

{

if (!DeleteFile(path))

Error("Delete file");

else

cout << "file is deleted\n";

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Создание файла и запись значений цикла

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void CreateAndWriteData(HANDLE \*hFile,char \*path)

{

// создаем файла для записи данных

\*hFile = CreateFile( path, GENERIC\_WRITE, 0, NULL, CREATE\_NEW, FILE\_ATTRIBUTE\_NORMAL, NULL );

// проверка открытости

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

Error("Create file ");

for(char i = '0'; i < '9'; i++ )

{

DWORD dwBytesWrite;

if (!WriteFile( \*hFile, &i, sizeof(i), &dwBytesWrite, (LPOVERLAPPED)NULL ))

Error("Write file ");

dwBytesWrite++;

}

Close(hFile);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Чтение содержимого файла и отображение в консоли

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ReadAndDisplay(HANDLE \*hFile,char \*path)

{

// создаем файла для записи данных

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

// проверка открытости

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

Error("Create file ");

for(;;)

{

DWORD dwBytesRead;

char n;

if (!ReadFile( \*hFile, &n, sizeof(n), &dwBytesRead, (LPOVERLAPPED)NULL))

Error("Read file ");

if (dwBytesRead == 0)

break;

else cout << n << endl;

}

CloseHandle(hFile);

cout << "The file is opened and read\n";

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Копирование файла с указанного пути в пункт назначения

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void CopyFiles(char \*src, char \*dst)

{

if (!CopyFile( src, dst, FALSE ))

Error("Copy file ");

cout << "The file is copied\n";

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Перемещение файла с указанного пути в пункт назначения

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MoveFiles(char \*src, char \*dst)

{

if (!MoveFile( src, dst ))

Error("Move file ");

cout << "The file is moved\n";

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Замещение файла с указанного пути в пункт назначения

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ReplaceFiles(char \*path, char \*repl, char \*resr)

{

if (!ReplaceFile( path, repl, resr, REPLACEFILE\_IGNORE\_MERGE\_ERRORS,0,0))

Error("Replace file ");

cout << "The file is replaced\n";

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Чтение и Изменение размера файла

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ReadAndEditFileSize(HANDLE \*hFile,char \*path, int derived)

{

DWORD dwFileSize; // размер файла

long p; // указатель позиции

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

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

Error("Create file ");

dwFileSize = GetFileSize(\*hFile, NULL);

if (dwFileSize == -1)

Error("Get file size");

cout << "Old file size is : " << dwFileSize << endl;

dwFileSize /= derived;

p = SetFilePointer(\*hFile,dwFileSize, NULL, FILE\_BEGIN);

if (p == -1)

Error("Set file size");

if (!SetEndOfFile(\*hFile))

Error("Set file size");

dwFileSize = GetFileSize(\*hFile, NULL);

if (dwFileSize == -1)

Error("Get file size");

cout << "New file size is : " << dwFileSize << endl;

Close(hFile);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Блокировка и разблокировка файла

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void LockAndUnLock(HANDLE \*hFile,char \*path)

{

DWORD dwFileSize; // размер файла

long p; // указатель позиции

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

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

Error("Create file ");

dwFileSize = GetFileSize(\*hFile, NULL);

if (dwFileSize == -1)

Error("Get file size");

if (!LockFile(\*hFile,0,0,dwFileSize,0))

Error("Lock file ");

cout << "Now file locked\n";

if (!UnlockFile(\*hFile,0,0,dwFileSize,0))

Error("Unlock file ");

cout << "Now file unlocked\n";

Close(hFile);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Получение информации о файле

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void GetFileInfo(HANDLE \*hFile, char \*path)

{

BY\_HANDLE\_FILE\_INFORMATION bhFileInfo;

SYSTEMTIME st; // системное время

DWORD dwFileType; // тип файла

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

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

Error("Create file ");

dwFileType = GetFileType(\*hFile);

switch (dwFileType)

{

case FILE\_TYPE\_CHAR : cout << "Char type file \n"; break;

case FILE\_TYPE\_DISK : cout << "Disk type file \n";break;

case FILE\_TYPE\_UNKNOWN :cout << "Unknown type file \n";break;

case FILE\_TYPE\_PIPE : cout << "Pipe type file \n"; break;

}

if(!GetFileInformationByHandle(\*hFile, &bhFileInfo))

Error("Get file information");

if (!FileTimeToSystemTime(&(bhFileInfo.ftCreationTime), &st))

Error("Get file information");

cout << "file creation time in system format : " << endl

<< "\t Year: " << st.wYear << endl

<< "\t Month: " << st.wMonth << endl

<< "\t DayOfWeek: " << st.wDayOfWeek<< endl

<< "\t Day: " << st.wDay << endl

<< "\t Hour: " << st.wHour << endl

<< "\t Minute: " << st.wMinute << endl

<< "\t Second: " << st.wSecond << endl

<< "\t MilliSecond: " << st.wMilliseconds << endl;

Close(hFile);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Установка указателя на файл

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void SetFilePointAndRead(HANDLE \*hFile,char \*path)

{

long n; // для номера записи

long p; // для указателя позиции

DWORD dwBytesRead; // количество прочитаных байт

int m; // причитанное число

// открытие файла для чтения

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

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

Error("Create file ");

cout << "Input a number from 0 to 9 : ";

cin >> n;

p = SetFilePointer(\*hFile, n\*sizeof(int),NULL, FILE\_BEGIN);

if (p == -1)

Error("Set file pointer");

cout << "File pointer : " << p << endl;

if (!ReadFile(\*hFile, &m, sizeof(m), &dwBytesRead, (LPOVERLAPPED)NULL))

Error("Read file ");

cout << "The Read number : " << m << endl;

Close(hFile);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Чтение аттрибутов файла

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ReadFileAtrribut(char \*path)

{

DWORD attribut;

attribut = GetFileAttributes(path);

if(attribut == -1)

Error( "Get attribut ");

if(attribut == FILE\_ATTRIBUTE\_NORMAL)

cout << "This is a normal file." << endl;

else

{

cout << "This is a not normal file." << endl;

return ;

}

if(!SetFileAttributes(path, FILE\_ATTRIBUTE\_HIDDEN))

Error("Set attribut");

cout << "Now the file is hidden." << endl;

if(!SetFileAttributes(path, FILE\_ATTRIBUTE\_NORMAL))

Error("Set attribut");

cout << "Now the file is again normal." << endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Получение типа исполнения файла

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void GetBinaryTypeOfFile(char \*path)

{

DWORD type;

// Визначаємо тип файлу

if (!GetBinaryType (path, &type))

Error("Get binary file ");

if (type == SCS\_32BIT\_BINARY)

cout <<"The file is Win32 based application." <<endl;

else

cout <<"The file is not Win32 based application." <<endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Освобождение буффера записи

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void FlushBuffer( HANDLE \*hFile, char \* path)

{

\*hFile = CreateFile( path, GENERIC\_WRITE, FILE\_SHARE\_READ, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL );

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

Error("Create");

int len = 10;

for (int i = 0; i < len ; i++)

{

DWORD dwBytesWrite;

if (!WriteFile( \*hFile, &i, sizeof(i), &dwBytesWrite, (LPOVERLAPPED)NULL))

Error("Write ");

if(i == len/2)

if(!FlushFileBuffers(\*hFile))

Error("Flush ");

}

Close(hFile);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()

{

HANDLE hFile;

char path[] = "D:\\demo\_file.txt";

char dst[] = "D:\\demo\_file\_copy.txt";

char move\_path[] = "G:\\move\_files.txt";

//DeleteFiles(path);

//CreateAndWriteData(&hFile,path);

//ReadAndDisplay(&hFile,path);

//CopyFiles(path, dst);

//MoveFiles(path, move\_path);

//ReplaceFiles(path, dst, move\_path);

//ReadAndEditFileSize(&hFile,path,2);

//LockAndUnLock(&hFile,path);

//GetFileInfo(&hFile,path);

//ReadFileAtrribut(path);

//SetFilePointAndRead(&hFile, path);

//GetBinaryTypeOfFile(path);

//FlushBuffer(&hFile,path);

return 0;

}