Комп'ютерний практикум №2
Робота з файлами для WIN32

Виконав:
Студент 2 курсу ФТІ
групи ФІ-92
Поночевний Назар Юрійович

Мета: вивчити основи роботи з двійковими і текстовими файлами на базі WIN32 API.

Завдання 10:

1. Створити системи каталогів типу FILE11/FILE12/FILE13/ та FILE21/FILE22/FILE23/;
2. Обійти другу систему директорій починаючи з кореневого каталогу та повернутись;
3. Створити файл у каталозі FILE23;
4. Read-Only. Час останньої зміни;
5. Скопіювати з якогось існуючого каталогу групу файлів (як бінарних так і текстових) у вказаній директорії;
6. Знайти файли у вказаній директорії за двома першими символами імені файлу;
7. Відкрити один з текстових файлів і зчитати зміст. Дописати до нього назву лабораторної роботи та номер завдання;
8. В заданому не пустому текстовому файлі підрахувати кількість слів, що мають довжину менше чотирьох букв. Читання файлу йде паралельно (асинхронно) з підрахунком в одному потоці;
9. Встановити розділюване блокування останнього 1 КБ файлу.

Код:

```c
#include <tchar.h>
#include <windows.h>
#include <stdio.h>
#include <strsafe.h>

#define BUFFERSIZE 100
#define TESTSTRLEN 1000
DWORD g_BytesTransferred = 0;

VOID CALLBACK FileIOCompletionRoutine(
    __in  DWORD dwErrorCode,
    __in  DWORD dwNumberOfBytesTransfered,
    __in  LPOVERLAPPED lpOverlapped
);
```

```c
VOID CALLBACK FileIOCompletionRoutine(
    __in  DWORD dwErrorCode,
    __in  DWORD dwNumberOfBytesTransfered,
    __in  LPOVERLAPPED lpOverlapped)
{
    // _tprintf(TEXT("Error code:\t%x\n"), dwErrorCode);
    // _tprintf(TEXT("Number of bytes:\t%x\n"),
dwNumberOfBytesTransfered);
    g_BytesTransferred = dwNumberOfBytesTransfered;
}

int searchFolders(LPCWSTR FolderName)
{
    WIN32_FIND_DATA FileData;
    HANDLE          hSearch;
    DWORD           dwAttrs;
    TCHAR           szDir[MAX_PATH];
    TCHAR           szNewDir[MAX_PATH];
    unsigned long long int numFolder = 0;

    _tprintf(TEXT("Search for files in %s\n"), FolderName);

    StringCchCopy(szDir, MAX_PATH, FolderName);
    StringCchCat(szDir, MAX_PATH, TEXT("\\*"));

    hSearch = FindFirstFile(szDir, &FileData);
    if (hSearch == INVALID_HANDLE_VALUE)
    {
        _tprintf(TEXT("No files found in %s\n"), FolderName);
        return 0;
    }

    while (TRUE)
    {
        StringCchCopy(szNewDir, MAX_PATH, FolderName);
        StringCchCat(szNewDir, MAX_PATH, TEXT("\\"));
        StringCchCat(szNewDir, MAX_PATH, FileData.cFileName);

        dwAttrs = GetFileAttributes(szNewDir);
        if (dwAttrs == INVALID_FILE_ATTRIBUTES) return 1;

        if (dwAttrs & FILE_ATTRIBUTE_DIRECTORY)
        {
            numFolder++;
            if (numFolder > 2)
```

```c
                searchFolders(szNewDir);
        }

        if (!FindNextFile(hSearch, &FileData))
        {
            if (GetLastError() == ERROR_NO_MORE_FILES)
            {
                _tprintf(TEXT("All files checked in %s\n"), FolderName);
                break;
            }
            else
            {
                printf("Could not find next file.\n");
                return 1;
            }
        }
    }

    FindClose(hSearch);
    return 0;
}

int _tmain(int argc, TCHAR* argv[])
{
    WIN32_FIND_DATA FileData;
    HANDLE          hFile;
    HANDLE          hSearch;
    DWORD           dwAttrs;
    TCHAR           szNewPath[MAX_PATH];
    TCHAR           sFileName[MAX_PATH];
    FILETIME        ft;
    SYSTEMTIME      st;
    TCHAR           szSearchDir[MAX_PATH];
    TCHAR           szSearchMask[MAX_PATH];
    BOOL            fFinished;
    TCHAR           sTargetFileDirectory[MAX_PATH];

    char            DataBuffer1[] = "Lorem ipsum.\n";
    char            DataBuffer2[] = "ROBOTA Z FAYLAMY dlia WIN32.
Variant 10\n";
    DWORD           dwBytesToWrite;
    DWORD           dwBytesWritten;
    BOOL            bErrorFlag;

    DWORD           dwBytesRead1 = 0;
    char            ReadBuffer1[BUFFERSIZE] = {0};
```

```c
    OVERLAPPED      ol1 = {0};
    DWORD           dwBytesRead2 = 0;
    char            ReadBuffer2[BUFFERSIZE] = {0};
    OVERLAPPED      ol2 = {0};

    BOOL fLockSuccess = FALSE;
    StringCchCopy(sTargetFileDirectory, MAX_PATH,
TEXT("FILE21\\FILE22\\FILE23"));

    if (argc != 3)
    {
        _tprintf(TEXT("Usage: %s <search dir> <search mask>\n"),
argv[0]);
        return 1;
    }

    // Create new directories.

    if (!CreateDirectory(TEXT("FILE11"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }
    if (!CreateDirectory(TEXT("FILE11\\FILE12"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }
    if (!CreateDirectory(TEXT("FILE11\\FILE12\\FILE13"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }

    if (!CreateDirectory(TEXT("FILE21"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }
    if (!CreateDirectory(TEXT("FILE21\\FILE22"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }
    if (!CreateDirectory(TEXT("FILE21\\FILE22\\FILE23"), NULL))
    {
```

```c
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }
    printf("Directories created\n\n");

    // Enumerate FILE21

    searchFolders(TEXT("FILE21"));

    // Create file in FILE23

    StringCchCopy(sFileName, MAX_PATH, sTargetFileDirectory);
    StringCchCat(sFileName, MAX_PATH, TEXT("\\TempFile.txt"));

    hFile = CreateFile(sFileName,
        GENERIC_WRITE,
        0,
        NULL,
        CREATE_NEW,
        FILE_ATTRIBUTE_NORMAL,
        NULL);

    if (hFile == INVALID_HANDLE_VALUE)
    {
        _tprintf(TEXT("Unable to create file \"%s\" for write.\n"),
sFileName);
        return 1;
    }

    _tprintf(TEXT("\nFile \"%s\" succesfully created.\n"), sFileName);

    // Write temp text to temp file

    dwBytesToWrite = (DWORD)strlen(DataBuffer1);
    dwBytesWritten = 0;
    bErrorFlag = FALSE;

    bErrorFlag = WriteFile(
        hFile,
        DataBuffer1,
        dwBytesToWrite,
        &dwBytesWritten,
        NULL);

    if (FALSE == bErrorFlag)
        printf("Unable to write to file.\n");
```

```c
    else
    {
        if (dwBytesWritten != dwBytesToWrite)
            printf("Error: dwBytesWritten != dwBytesToWrite\n");
        else
            _tprintf(TEXT("Wrote %d bytes to %s successfully.\n"),
dwBytesWritten, sFileName);
    }

    // Set attributes

    GetSystemTime(&st);
    SystemTimeToFileTime(&st, &ft);
    SetFileTime(hFile,
        (LPFILETIME) NULL,
        (LPFILETIME) NULL,
        &ft);

    dwAttrs = GetFileAttributes(sFileName);
    if (!(dwAttrs & FILE_ATTRIBUTE_READONLY))
    {
        SetFileAttributes(sFileName,
            dwAttrs | FILE_ATTRIBUTE_READONLY);
        SetFileAttributes(sFileName,
            dwAttrs | FILE_ATTRIBUTE_NORMAL);
    }

    _tprintf(TEXT("\nRead-Only and LastWriteFileTime attrs for file
\"%s\" changed.\n"), sFileName);

    CloseHandle(hFile);

    // Copy "*.txt" and "*.exe" files to FILE23 folder

    StringCchCopy(szSearchDir, MAX_PATH, argv[1]);
    StringCchCat(szSearchDir, MAX_PATH, TEXT("\\*.txt"));

    hSearch = FindFirstFile(szSearchDir, &FileData);
    if (hSearch == INVALID_HANDLE_VALUE)
    {
        printf("No *.txt files found.\n");
        return 1;
    }

    fFinished = FALSE;
    while (!fFinished)
```

```c
    {
        StringCchPrintf(szNewPath, sizeof(szNewPath) /
sizeof(szNewPath[0]), TEXT("%s\\%s"), sTargetFileDirectory,
FileData.cFileName);

        if (!CopyFile(FileData.cFileName, szNewPath, FALSE))
        {
            _tprintf(TEXT("Could not copy file %s to %s\n"),
FileData.cFileName, szNewPath);
            return 1;
        }

        if (!FindNextFile(hSearch, &FileData))
        {
            if (GetLastError() == ERROR_NO_MORE_FILES)
            {
                _tprintf(TEXT("\nCopied %s to %s\n"), szSearchDir,
sTargetFileDirectory);
                fFinished = TRUE;
            }
            else
            {
                printf("Could not find next file.\n");
                return 1;
            }
        }
    }

    FindClose(hSearch);

    StringCchCopy(szSearchDir, MAX_PATH, argv[1]);
    StringCchCat(szSearchDir, MAX_PATH, TEXT("\\*.exe"));

    hSearch = FindFirstFile(szSearchDir, &FileData);
    if (hSearch == INVALID_HANDLE_VALUE)
    {
        printf("No *.exe files found.\n");
        return 1;
    }

    fFinished = FALSE;
    while (!fFinished)
    {
        StringCchPrintf(szNewPath, sizeof(szNewPath) /
sizeof(szNewPath[0]), TEXT("%s\\%s"), sTargetFileDirectory,
FileData.cFileName);
```

```c
        if (!CopyFile(FileData.cFileName, szNewPath, FALSE))
        {
            _tprintf(TEXT("Could not copy file %s to %s\n"),
FileData.cFileName, szNewPath);
            return 1;
        }

        if (!FindNextFile(hSearch, &FileData))
        {
            if (GetLastError() == ERROR_NO_MORE_FILES)
            {
                _tprintf(TEXT("Copied %s to %s\n"), szSearchDir,
sTargetFileDirectory);
                fFinished = TRUE;
            }
            else
            {
                printf("Could not find next file.\n");
                return 1;
            }
        }
    }

    FindClose(hSearch);

    // Find files in FILE23 by a mask

    StringCchCopy(szSearchMask, MAX_PATH, sTargetFileDirectory);
    StringCchCat(szSearchMask, MAX_PATH, TEXT("\\"));
    StringCchCat(szSearchMask, MAX_PATH, argv[2]);

    printf("\nSearching files...\n");
    hSearch = FindFirstFile(szSearchMask, &FileData);
    if (hSearch == INVALID_HANDLE_VALUE)
    {
        _tprintf(TEXT("No %s files found.\n"), argv[2]);
        return 1;
    }

    fFinished = FALSE;
    while (!fFinished)
    {
        StringCchPrintf(szNewPath, sizeof(szNewPath) /
sizeof(szNewPath[0]), TEXT("%s\\%s"), sTargetFileDirectory,
FileData.cFileName);
```

```c
        _tprintf(TEXT("Found: %s\n"), szNewPath);

        if (!FindNextFile(hSearch, &FileData))
        {
            if (GetLastError() == ERROR_NO_MORE_FILES)
            {
                _tprintf(TEXT("All %s files checked in %s\n"), argv[2],
sTargetFileDirectory);
                fFinished = TRUE;
            }
            else
            {
                printf("Could not find next file.\n");
                return 1;
            }
        }
    }

    FindClose(hSearch);

    // Read TempFile.txt and append Lab Info

    StringCchCopy(sFileName, MAX_PATH, sTargetFileDirectory);
    StringCchCat(sFileName, MAX_PATH, TEXT("\\TempFile.txt"));

    hFile = CreateFile(sFileName,
        GENERIC_READ,
        FILE_SHARE_READ,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED,
        NULL);

    if (hFile == INVALID_HANDLE_VALUE)
    {
        _tprintf(TEXT("Unable to open file \"%s\" for read.\n"),
sFileName);
        return 1;
    }

    if (FALSE == ReadFileEx(hFile, ReadBuffer1, BUFFERSIZE - 1, &ol1,
FileIOCompletionRoutine))
    {
        printf("Unable to read from file.\n GetLastError=%08x\n",
GetLastError());
```

```c
        CloseHandle(hFile);
        return 1;
    }
    SleepEx(5000, TRUE);
    dwBytesRead1 = g_BytesTransferred;

    if (dwBytesRead1 > 0 && dwBytesRead1 <= BUFFERSIZE - 1)
    {
        ReadBuffer1[dwBytesRead1] = '\0'; // NULL character

        _tprintf(TEXT("\nData read from %s (%d bytes): \n"), sFileName,
dwBytesRead1);
        printf("%s", ReadBuffer1);
    }
    else if (dwBytesRead1 == 0)
        _tprintf(TEXT("No data read from file %s\n"), sFileName);
    else
        printf("\n ** Unexpected value for dwBytesRead ** \n");

    CloseHandle(hFile);

    hFile = CreateFile(sFileName,
        FILE_APPEND_DATA,
        0,
        NULL,
        OPEN_ALWAYS,
        FILE_ATTRIBUTE_NORMAL,
        NULL);

    if (hFile == INVALID_HANDLE_VALUE)
    {
        _tprintf(TEXT("Unable to create file \"%s\" for write.\n"),
sFileName);
        return 1;
    }

    dwBytesToWrite = (DWORD)strlen(DataBuffer2);
    dwBytesWritten = 0;
    bErrorFlag = FALSE;

    bErrorFlag = WriteFile(
        hFile,
        DataBuffer2,
        dwBytesToWrite,
        &dwBytesWritten,
        NULL);
```

```c
    if (FALSE == bErrorFlag)
        printf("Unable to write to file.\n");
    else
    {
        if (dwBytesWritten != dwBytesToWrite)
            printf("Error: dwBytesWritten != dwBytesToWrite\n");
        else
            _tprintf(TEXT("Wrote (append) %d bytes to %s
successfully.\n"), dwBytesWritten, sFileName);
    }

    dwAttrs = GetFileAttributes(sFileName);
    if (!(dwAttrs & FILE_ATTRIBUTE_READONLY))
    {
        SetFileAttributes(sFileName,
            dwAttrs | FILE_ATTRIBUTE_READONLY);
    }

    CloseHandle(hFile);

    // Async count words < 4 letters in TempFile.txt

    StringCchCopy(sFileName, MAX_PATH, sTargetFileDirectory);
    StringCchCat(sFileName, MAX_PATH, TEXT("\\TempFile.txt"));

    hFile = CreateFile(sFileName,
        GENERIC_READ,
        FILE_SHARE_READ,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED,
        NULL);

    if (hFile == INVALID_HANDLE_VALUE)
    {
        _tprintf(TEXT("Unable to open file \"%s\" for read.\n"),
sFileName);
        return 1;
    }

    if (FALSE == ReadFileEx(hFile, ReadBuffer2, BUFFERSIZE - 1, &ol2,
FileIOCompletionRoutine))
    {
        printf("Unable to read from file.\n GetLastError=%08x\n",
GetLastError());
```

```c
            CloseHandle(hFile);
            return 1;
        }
    SleepEx(5000, TRUE);
    dwBytesRead2 = g_BytesTransferred;

    if (dwBytesRead2 > 0 && dwBytesRead2 <= BUFFERSIZE - 1)
    {
        ReadBuffer2[dwBytesRead2] = '\0'; // NULL character

        _tprintf(TEXT("\nData read from %s (%d bytes): \n"), sFileName,
dwBytesRead2);
        printf("%s", ReadBuffer2);

        int i = 0, word_length = 0, words_amount = 0;
        while (TRUE)
        {
            if ((ReadBuffer2[i] == ' ') | (ReadBuffer2[i] == '\0'))
            {
                if (word_length < 4)
                    words_amount++;
                word_length = 0;
                if (ReadBuffer2[i] == '\0')
                    break;
            }
            else
                word_length++;
            i++;
        }

        _tprintf(TEXT("Number of words < 4 letters: %d\n"),
words_amount);
    }
    else if (dwBytesRead2 == 0)
        _tprintf(TEXT("No data read from file %s\n"), sFileName);
    else
        printf("\n ** Unexpected value for dwBytesRead ** \n");

    CloseHandle(hFile);

    // Lock last 1kb of TempFile.txt

    StringCchCopy(sFileName, MAX_PATH, sTargetFileDirectory);
    StringCchCat(sFileName, MAX_PATH, TEXT("\\TempFile.txt"));

    hFile = CreateFile(sFileName,
```

```c
        GENERIC_READ,
        FILE_SHARE_READ,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED,
        NULL);

    if (hFile == INVALID_HANDLE_VALUE)
    {
        _tprintf(TEXT("Unable to open file \"%s\" for read.\n"),
sFileName);
        return 1;
    }

    OVERLAPPED sOverlapped;
    sOverlapped.Offset = TESTSTRLEN * 3;
    sOverlapped.OffsetHigh = 0;

    fLockSuccess = LockFileEx(hFile,
        NULL,
        0,
        TESTSTRLEN,
        0,
        &sOverlapped);

    if (!fLockSuccess)
    {
        printf("LockFileEx failed (%d)\n", GetLastError());
        return 1;
    }
    else _tprintf(TEXT("\nLockFileEx last 1kb of %s succeeded\n"),
sFileName);

    // Unlock the file

    fLockSuccess = UnlockFileEx(hFile,
        0,
        TESTSTRLEN,
        0,
        &sOverlapped);

    if (!fLockSuccess)
    {
        printf("UnlockFileEx failed (%d)\n", GetLastError());
        return 1;
    }
```

```
    else _tprintf(TEXT("UnlockFileEx last 1kb of %s succeeded\n"),
sFileName);

    CloseHandle(hFile);

    return 0;
}
```

Скріншоти:

```
D:\Microsoft Visual Studio\Workspace\Lab22\Debug>Lab22.exe . La*
Directories created

Search for files in FILE21
Search for files in FILE21\FILE22
Search for files in FILE21\FILE22\FILE23
All files checked in FILE21\FILE22\FILE23
All files checked in FILE21\FILE22
All files checked in FILE21

File "FILE21\FILE22\FILE23\TempFile.txt" succesfully created.
Wrote 13 bytes to FILE21\FILE22\FILE23\TempFile.txt successfully.

Read-Only and LastWriteFileTime attrs for file "FILE21\FILE22\FILE23\TempFile.txt" changed.

Copied .\*.txt to FILE21\FILE22\FILE23
Copied .\*.exe to FILE21\FILE22\FILE23

Searching files...
Found: FILE21\FILE22\FILE23\Lab22.exe
Found: FILE21\FILE22\FILE23\Lab22.vcxproj.FileListAbsolute.txt
All La* files checked in FILE21\FILE22\FILE23

Data read from FILE21\FILE22\FILE23\TempFile.txt (13 bytes):
Lorem ipsum.
Wrote (append) 40 bytes to FILE21\FILE22\FILE23\TempFile.txt successfully.

Data read from FILE21\FILE22\FILE23\TempFile.txt (53 bytes):
Lorem ipsum.
ROBOTA Z FAYLAMY dlia WIN32. Variant 10
Number of words < 4 letters: 2

LockFileEx last 1kb of FILE21\FILE22\FILE23\TempFile.txt succeeded
UnlockFileEx last 1kb of FILE21\FILE22\FILE23\TempFile.txt succeeded

D:\Microsoft Visual Studio\Workspace\Lab22\Debug>
```

Повний код можна знайти у GitHub-репозиторії:
https://github.com/NazarPonochevnyi/Programming-Labs/blob/master/System%20Progra
mming/Lab2/lab2.cpp