



1. Посчитать число каталогов
2. Обработка

#include <Windows.h>

#include <malloc.h>

#include <string.h>

#include <tchar.h>

#include <cstdlib>

#include <iostream>

#include <string>

#include <string.h>

#include <stdio.h>

#include <vector>

#include <array>

using namespace std;

#if defined(\_UNICODE)

#define \_T(x) L##x

#else

#define \_T(x) x

#endif

#define \_CRT\_SECURE\_NO\_WARNINGS

#define BUFFER\_SIZE 260

#define DIRECTORY\_NAME \_T("C:\\Users\\Constantine\\Desktop\\Education\\Practice1\\2\\Test\_2\\")

#define REC\_COUNT 0

#define REC\_DIRS 1

int empty\_files\_counter = 1; //итератор zero<индекс>

bool is\_pseudodirectory(WIN32\_FIND\_DATA\* wfd) {

return wcscmp(wfd->cFileName, \_T(".")) == 0 || wcscmp(wfd->cFileName, \_T("..")) == 0;

}

bool is\_directory(WIN32\_FIND\_DATA\* wfd) {

return (wfd->dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY) && !(wfd->dwFileAttributes & FILE\_ATTRIBUTE\_REPARSE\_POINT);

}

void get\_new\_path(LPTSTR path, WIN32\_FIND\_DATA\* wfd, LPTSTR new\_path) {

wcscpy\_s(new\_path, BUFFER\_SIZE, path);

wcscat\_s(new\_path, BUFFER\_SIZE, wfd->cFileName);

wcscat\_s(new\_path, BUFFER\_SIZE, L"\\");

}

void get\_old\_name(LPTSTR path, WIN32\_FIND\_DATA\* wfd, LPTSTR old\_file\_name)

{

wcscpy\_s(old\_file\_name, BUFFER\_SIZE, path);

wcscat\_s(old\_file\_name, BUFFER\_SIZE, L"\\");

wcscat\_s(old\_file\_name, BUFFER\_SIZE, wfd->cFileName);

}

void get\_new\_name(LPTSTR path, WIN32\_FIND\_DATA\* wfd, LPTSTR new\_file\_name)

{

TCHAR empty\_file\_number[BUFFER\_SIZE];

//новое имя

wcscat\_s(new\_file\_name, BUFFER\_SIZE, path);

wcscat\_s(new\_file\_name, BUFFER\_SIZE, \_T("zero"));

\_itot\_s(empty\_files\_counter, empty\_file\_number, 10); //в символы в 10-ой сс

wcscat\_s(new\_file\_name, BUFFER\_SIZE, (LPTSTR)empty\_file\_number); //значение

wcscat\_s(new\_file\_name, BUFFER\_SIZE, \_T(".txt")); //расширение

}

HANDLE get\_directory\_iterator(LPTSTR path\_, WIN32\_FIND\_DATA\* wfd) {

TCHAR loc\_path[BUFFER\_SIZE];

wcscpy\_s(loc\_path, BUFFER\_SIZE, path\_);

wcscat\_s(loc\_path, BUFFER\_SIZE, L"\*");

HANDLE hdl = FindFirstFile(loc\_path, wfd);

if (hdl == INVALID\_HANDLE\_VALUE)

{

FindClose(hdl);

exit(0);

}

return hdl;

}

int get\_file\_size(WIN32\_FIND\_DATA\* wfd, LPTSTR path) {

TCHAR loc\_path[BUFFER\_SIZE] = \_T("");

get\_old\_name(path, wfd, loc\_path);

HANDLE file = CreateFile(loc\_path, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

int size = GetFileSize(file, NULL);

CloseHandle(file);

return size;

}

HANDLE mut = CreateMutex(NULL, FALSE, NULL);

bool rename\_file(WIN32\_FIND\_DATA\* wfd, LPTSTR path)

{

//переименовывание файла

//старое имя

TCHAR old\_file\_name[BUFFER\_SIZE] = \_T("");

get\_old\_name(path, wfd, old\_file\_name);

//новое имя

TCHAR new\_file\_name[BUFFER\_SIZE] = \_T("");

get\_new\_name(path, wfd, new\_file\_name);

return MoveFile(old\_file\_name, new\_file\_name); //типа переименовывает

}

//перебор файлов

void search\_file(LPTSTR path) {

WIN32\_FIND\_DATA wfd = { 0 };

HANDLE hdl = get\_directory\_iterator(path, &wfd);

//поиск всех файлов в каталоге

do {

if (is\_directory(&wfd))

{

continue;

} // файл

if (is\_pseudodirectory(&wfd))

{

continue;

} // не псевдодиректория

if (!(get\_file\_size(&wfd, path) == 0)) {

continue;

} // нулевой размер

if (!(rename\_file(&wfd, path)))

{

cout << "ошибка" << endl;

continue;

}

rename\_file(&wfd, path);

WaitForSingleObject(mut, INFINITE);

std::wcout << path << L"//" << wfd.cFileName << endl;

empty\_files\_counter++;

ReleaseMutex(mut);

} while (FindNextFile(hdl, &wfd) != NULL);

FindClose(hdl); //поиск окончен

}

//перебор каталогов

void rec\_dir(LPTSTR path) //DIRECTORY\_NAME

{

wcout << path << endl;

search\_file(path); //изначально перебираю файлы в папке -> потом рассматриваю каталоги

WIN32\_FIND\_DATA wfd = { 0 };

HANDLE hdl = get\_directory\_iterator(path, &wfd);

std::vector<HANDLE> threads; //потоки

std::vector<DWORD> thrId; //идентификаторы потоков

std::vector<LPTSTR> paths;

do

{

//обработка каталогов -> потоки

//WaitForSingleObject(mut, INFINITE);

//Начало критической секции

if (!(is\_directory(&wfd))) //если директория

{

continue;

} //директория

if (is\_pseudodirectory(&wfd)) //не точки

{

continue;

} //не псевдодиректория

paths.push\_back(new TCHAR[BUFFER\_SIZE]()); //мусор в вектор

get\_new\_path(path, &wfd, paths.back());

//thrId.emplace\_back();

threads.push\_back(CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)rec\_dir,

(LPVOID)paths.back(), 0, NULL));

if (threads.back() == NULL)

{

cout << "ошибка создания потока" << endl;

}

//WaitForSingleObject(threads.back(), INFINITE);

// конец критической секции

//ReleaseMutex(mut);

} while (FindNextFile(hdl, &wfd));

FindClose(hdl); // поиск окончен

if (threads.size() != 0)

{

WaitForMultipleObjects(threads.size(), threads.data(), TRUE, INFINITE);

}

for (int a = 0; a < threads.size(); a++)

{

CloseHandle(threads[a]);

}

}

int wmain(int argc, wchar\_t\* argv[])

{

setlocale(LC\_ALL, "Russian");

rec\_dir((LPTSTR)DIRECTORY\_NAME);

return 0;

}