**Тема 7: «Работа с файловой системой»**

**Цель работы:**

Составить алгоритм и разработать программу

Практическое задание:

**Задача 7.**

Напишите программу, которая создает поток. Используйте атрибуты по умолчанию. Родительский и вновь созданный поток должны распечатать десять строк текста из файла так, чтобы вывод родительского и дочернего потока был синхронизован: сначала родительский поток выводил бы 3 строки, затем дочерний одну, затем родительский 3 строки и т.д. Используйте критические секции.

**Математическая формулировка:**

**Задача 7.**

**Программа работает уже с существующем текстовым файлом файлом. Последовательно выводятся строчки текстового файла в разных патоках и с разным временным интервалом для каждого патока.**

**Алгоритм:**

Задача 7.





**Текст программного модуля:**

**Задача 7.**

**Main.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <ctime>

#include <fcntl.h>

#include <io.h>

#include <iostream>

#include <mutex>

#include <queue>

#include <string>

#include <thread>

#include <Windows.h>

using namespace std;

mutex mtx;

mutex mtx2;

// проверка на существование

bool TestDirForPath(HANDLE hf, wstring path) {

if (hf == nullptr || hf == INVALID\_HANDLE\_VALUE) {

mtx.lock();

wcout << L"Не удалось открыть файл по пути: " << path << endl;

mtx.unlock();

return true;

}

return false;

}

//сканирование директории

void SearchForDir(wstring path, wstring name, queue<wstring>& pathRezuld) {

WIN32\_FIND\_DATA findData;

queue<thread\*> qTreadsLocal;

HANDLE hf = FindFirstFile((path + L"\*").c\_str(), &findData);

if (TestDirForPath(hf, path)) return;

do {

try {

if (TestDirForPath(hf, path)) continue;

// пропускаем директории "." и ".."

if (!wcsncmp(findData.cFileName, L".", 1) ||

!wcsncmp(findData.cFileName, L"..", 2))

continue;

wstring filePath = path + findData.cFileName;

mtx.lock();

wcout << L"ID thread: " << this\_thread::get\_id()

<< L" path: " << filePath << endl;

mtx.unlock();

if (findData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY) {

// если это директория

filePath += L"\\";

//thread\* th = new thread(SearchForDir, filePath, name, ref(pathRezuld));

mtx2.lock();

qTreadsLocal.push(new thread(SearchForDir, filePath, name, ref(pathRezuld)));

// сканируем директорию в новом потоке

mtx2.unlock();

} else if (!wcsncmp(findData.cFileName, name.c\_str(), name.size())) {

// если есть совпадение с искомым именем файла

mtx2.lock();

pathRezuld.push(filePath);

mtx2.unlock();

mtx.lock();

wcout << L"HIT!" << endl;

mtx.unlock();

}

} catch (exception e) {

mtx.lock();

wcout << L"Ошибка: " << e.what() << L" Path: " << path << endl;

mtx.unlock();

}

} while (FindNextFile(hf, &findData) != 0);

//закрываем потоки

FindClose(hf);

while (!qTreadsLocal.empty()){

qTreadsLocal.front()->join();

qTreadsLocal.pop();

}

}

// конвертация массива char в wchar

wchar\_t\* ConvertChartoWchar(const char\* ch) {

const size\_t cSize = strlen(ch) + 1;

auto wch = new wchar\_t[cSize];

mbstowcs(wch, ch, cSize);

return wch;

}

int main() {

\_setmode(\_fileno(stdout), \_O\_U8TEXT);

\_setmode(\_fileno(stdin), \_O\_U8TEXT);

wstring fname;

queue<wstring> Paths; // очередь из имен путей к файлу

queue<thread\*> qTreads; // очередь потоков

wcout << L"Введите название файла\n: ";

wcin >> fname;

char buf[26];

GetLogicalDriveStringsA(sizeof(buf), buf); // получам список дисков установленных в стему

for (char\* s = buf; \*s; s += strlen(s) + 1) {

wstring disk(ConvertChartoWchar(s));

//начинам сканировать диски

qTreads.push(new thread(SearchForDir, disk, fname, ref(Paths)));

}

//закрываем потоки

while (!qTreads.empty()) {

qTreads.front()->join();

qTreads.pop();

}

wcout << L"\nFILE PATH:\n";

if (Paths.empty()) {

wcout << L"Файл не найден\n";

cin.get();

return 0;

}

queue<wstring> TestPaths = Paths;

for (unsigned int i = 1; !TestPaths.empty(); ++i) {

wcout << i << L") " << TestPaths.front() << endl;

TestPaths.pop();

}

int mod;

while (true) {

wcout << L"С каким файлом работать?\n:";

wcin >> mod;

if (1 <= mod && mod <= Paths.size() && cin.good()) break;

wcout << L"Ошибка ввода\n";

}

for (unsigned int i = 1; i < mod && !Paths.empty(); ++i) Paths.pop();

wstring path = Paths.front();

WIN32\_FIND\_DATA findData;

HANDLE fData = FindFirstFileW(path.c\_str(), &findData);

LONGLONG nFileLen = (findData.nFileSizeHigh \* (MAXDWORD + 1)) + findData.nFileSizeLow;

wcout << L"Размер файла: " << nFileLen << L" байт" << endl;

SYSTEMTIME sm;

FileTimeToSystemTime(&findData.ftCreationTime, &sm);

wprintf(L"Время создания: %d.%d.%d %d:%d:%d:%d", sm.wDay, sm.wMonth, sm.wYear, sm.wHour, sm.wMinute, sm.wSecond, sm.wMilliseconds);

// проверяем не является ли файл системным

if (findData.dwFileAttributes & FILE\_ATTRIBUTE\_DEVICE || //Это значение зарезервировано для использования системы.

findData.dwFileAttributes & FILE\_ATTRIBUTE\_SYSTEM ||

//Файл или каталог, в который операционная система использует часть или используется исключительно.

findData.dwFileAttributes & FILE\_ATTRIBUTE\_TEMPORARY || // Файл, используемый для временного хранилища.

findData.dwFileAttributes & FILE\_ATTRIBUTE\_VIRTUAL || // Это значение зарезервировано для использования системы.

findData.dwFileAttributes & FILE\_ATTRIBUTE\_HIDDEN) { //Файл или каталог скрыт.Он не включен в обычный список каталогов.

wcout << L"Файл системный\n";

} else {

while (true) {

wcout << L"Файл не системный, атрибут: "<< findData.dwFileAttributes << L"\nПеременовать его? Y/N\n:";

wstring input;

wcin >> input;

if (input == L"y" || input == L"Y" || input == L"yes" ||

input == L"YES" || input == L"Yes" && cin.good()){

wcout << L"Введите новое название файла\n:";

wcin >> input;

if (!CopyFileW(path.c\_str(), (path.substr(0, path.size() - wcslen(findData.cFileName)) + input).c\_str(), 0)||

!DeleteFileW(path.c\_str())) {

wcout << L"Не удалось переминовть файл";

}

} else if (input == L"n" || input == L"N" || input == L"no" ||

input == L"NO" || input == L"No" && cin.good()){

break;

} else wcout << L"Не понятный ввод\n";

}

}

FindClose(fData);

cin.get();

return 0;

}

План тестирования на основе потока данных:

Задача 7.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Классы эквивалентности** | **Тестовый набор** | **Ожидаемые результаты** |
| 1 | Класс допустимых значений  X является именем файла существующий в системе  Y является элементом списка | X=”testFile.txt”  Y- номер в списке  Z = “y”  N = “test” | Если имя файла находится в системе и в нескольких экземплярах, то на выбор выводится список из различных путей, если y указывает на существующий элемент списка, то выводится информация о файле и, если он не системный предоставляется возможность переименовать файл. В n по решению z. Если пользователь согласился и ввел новое имя файл должен пересоздаться с новым именем. |
| 2 | Класс допустимых значений  X является именем файла существующий в системе  Y является элементом списка | X=”testFile.txt”  Y- номер в списке  Z = “n” | Если имя файла находится в системе и в нескольких экземплярах, то на выбор выводится список из различных путей, если y указывает на существующий элемент списка, то выводится информация о файле и, если он не системный предоставляется возможность переименовать файл. В n по решению z. Если пользователь отказался, то программа должна завершиться. |
| 3 | Класс недопустимых значений  X является именем файла не существующего в системе | X=”testtesttest.test” | Если имя x не разу не встретиться в системе, программа должна сообщить об этом и прекратить свою работу |
| 4 | Класс допустимых значений  X является именем файла существующий в системе  Y не является элементом списка | X=”testFile.txt”  Y- не являтся элементом списка | Если имя файла находится в системе и в нескольких экземплярах, то на выбор выводится список из различных путей, если y указывает на не существующий элемент списка, то выводится ошибка “Ошибка ввода” и просят ввести номер еще раз. |

Результаты испытаний в виде скриншотов:

Задача 7.

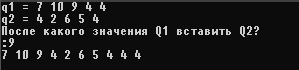


Рис. По № 1

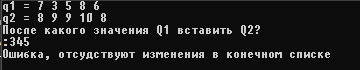


Рис. По № 2