|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
|  |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |

|  |
| --- |
| **ОТЧЕТ** |
|  |
| по учебной практике  УП.01.01. Учебная практика  по профессиональному модулю ПМ.01. Разработка программных модулей программного обеспечения для компьютерных систем |
|  |
|  |
| студента группы 090203-9о-20/1  специальности 09.02.03 Программирование в компьютерных системах |
| Смирнова Константина Вадимовича |

|  |  |  |
| --- | --- | --- |
| Студента |  | К.В. Смирнова |
| Руководитель практики от университета: |  |  |
| Преподаватель |  | Л.В. Салахова |

Москва

2023

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Введение……………………….......................................................................... | 3 |
| Основная часть………………………………………………………………... | 5 |
| Заключение……………………………………………………………............ | 10 |
| Приложение…………………………………………………………………… | 11 |

**ВВЕДЕНИЕ**

Учебная практика «Разработка программных модулей программного обеспечения для компьютерных систем» затронула профессиональные компетенции (ПК), такие как:

**−** осуществление разработки кода программного продукта на основе готовых спецификаций на уровне модуля,

**−** выполнение отладки программных модулей с использованием специализированных программных средств,

**−** выполнение тестирования программных модулей,

**−** осуществление оптимизации программного кода модуля.

В ходе практики были затронуты общие компетенции (ОК), такие как:

**−** понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес,

**−** организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество,

**−** принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность,

**−** осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития,

**−** использовать информационно-коммуникационные технологии в профессиональной деятельности,

**−** работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями,

**−** брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий,

**−** самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации,

**−** ориентироваться в условиях частой смены технологий в профессиональной деятельности.

По результатам освоения профессионального модуля были получены навыки:  
− осуществления разработки кода программного модуля на современных языках программирования,

− создания программы по разработанному алгоритму как отдельный модуль,  
− выполнение отладки и тестирования программы на уровне модуля

**ОСНОВНАЯ ЧАСТЬ**

В ходе практики осуществлялась разработка программ для решения поставленных задач. Уникальными заданиями являлись только задания 1, 4 и 7, остальные **−** являлись вариациями этих заданий, с дополнениями (например, нужно было выполнить задание с использованием потоков и pipe).

В ходе практики понадобилось повторение уже пройденных на других дисциплинах функций, а изучение новых, материал о которых находился в сети интернет.

Понадобилось повторить функции такие как CreateThread, WaitForSingleObject, CloseHandle, InitializeCriticalSection, EnterCriticalSection, LeaveCriticalSection, DeleteCriticalSection, CreatePipe, ReadFile, WriteFile, CreateNamedPipe, CreateFile. Понадобилось повторить работу со структурой WIN32\_FIND\_DATAW.

Потребовалось изучить такие функции как fork, waitpid, pipe, write, read. Потребовалось найти информацию о работе со структурами DIR и Stat. При работе со Stat понадобилось изучить все его особенности и информацию о файлах, которую можно извлечь из структуры.

Основная сложность данной практики состояла в том, что до неё не было опыта работы с языком C и С++ в среде Linux, соответственно, пришлось изучать самостоятельно.

Сложность была в том, что в каждой работе нужно было реализовать правильную передачу в функцию полного пути до следующей папки или до файла. Работа с получением полных путей к файлам была реализована при помощи функций strcat, strlen и strcmp, использование которых пришлось повторить, так как опыта работы с ними было недостаточно. Для этих функций в среде Windows приходилось учитывать и тип входящих данных (например для TCHAR используется функция \_tcscat), так как в Windows, в отличии от Linux, для строковых переменных использовались массивы символов типа TCHAR. В Linux использовался только тип char.

После ознакомления с разработкой кода в среде Linux, с первой работой не возникло особых проблем. Для работы с файлами использовалась структура DIR, которая является почти в точности такой же, как и структура WIN32\_FIND\_DATAW в Windows, опыт работы с которой был из дисциплины «Системное программирование».

В задании №1 Все обычные файлы, имеющие нулевую длину переименовать в имя, имеющее формат: zero<номер>, где <номер> – это числа по порядку, начиная с 1.. Задание требовалось выполнить в среде Linux.

Были повторены основы языка C и С++, закреплены навыки работы со средой Linux и с функциями, предназначенными для работы с каталогами. Например функции rename (stdio.h), opendir, closedir, readdir, scandir (dirent.h), unlink (unistd.h), stat (sys/stat.h); переменную errno, функцию strerror.

В задании №2 требовалось выполнить задание практической работы №1, но в среде Windows.

Основной целью работы являлось ознакомление с функциями, предназначенными для работы с каталогами (WinAPI).

В коде работы использовалась структура WIN32\_FIND\_DATAW и функции FindFirstFile, FindNextFile, опыт работы с которыми был получен при освоении дисциплины «Системное программирование».

Так как все используемые структуры и методы написания кода были уже изучены на других дисциплинах и был опыт решения подобных задач, трудностей с работой №2 не возникло.

В задании №3 нужно было выполнить задания практических работ №1 и №2 с учетом того, что файлы ищутся не только в текущем каталоге, но и во всех его подкаталогах (любой степени вложенности).

Основной целью работы являлся дополнительный практический опыт использования рекурсии.

Так как нужно было выполнить задания №1 и №2, разработка кода задания №3 проходила как в среде Windows, так и в среде Linux.

Во время выполнения работы, прибавился опыт работы с рекурсивными функциями. Сложность состояла в передаче нужных путей к папкам в функцию, в остальном задание было простым.

В среде Windows для перебора файлов в папке использовалась структура WIN32\_FIND\_DATAW и функции FindFirstFile, FindNextFile, для работы в среде Linux – структура DIR и её функции.

В задании №4 требовалось написать функцию sort, которая бы выполняла роль sort − встроенной функции Linux.

Эта функция сортирует строки внутри файла. 4-ое задание было самым простым за практику, так как код был не объёмным и не сложным в написании и понимании.

Для задания требовалось передавать пути к файлам из консоли, что было реализовано с помощью argc и argv – параметров, которые передаются main из консоли при запуске программы.

Основной целью работы являлось изучение способов обработки аргументов из командной строки. Цель была достигнута.

В задании №5требовалось реализовать задание 3-ей практической работы таким образом, чтобы каждый каталог обрабатывался отдельным процессом.

Задание необходимо было выполнить как в среде Linux, так и в среде Windows, только в Windows нужно было использовать потоки.

Основная цель работы состояла в изучении способа создания новых процессов и потоков.

Для написания кода в среде Linux потребовалось изучить новые функции, такие как fork и waitpid. До практики опыта работы с ними не было.

В коде в среде Windows использовались такие функции как CreateThread, WaitForSingleObject, CloseHandle. Опыт работы с ними был получен во время освоения дисциплины «Системное программирование», поэтому написание кода с использованием этих функций не вызвало трудностей.

Сложности возникли при разработке кода в среде Linux, потому что ранее не было опыта работы с fork. Функцию fork пришлось изучать самостоятельно. С основным построением алгоритма не возникло проблем.

В задании №7 требовалось реализовать задание в соответствии со своим вариантом в среде Linux (с обязательным использованием рекурсии и возвратом из функции результата, соответствующему каталогу, имя которого было передано в нее).

Задание варианта состояло: найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий наименьшее число файлов, доступных для чтения (см. S\_IRUSR). Если таких каталогов несколько, то вывести любой из них.

Основная цель работы состояла в изучении методики решения задачи динамического программирования на простейшем примере.

Алгоритм решения данного задания такой же, как и в задании №3, но вместо удаления нужно передавать имя папки в функцию.

Трудность задания состояла в том, что нужно было передавать в функцию не только имя искомой папки, но и путь до неё и количество файлов, подошедших по условию. Было сложно реализовать эту задачу потому, что часто возникало зацикливание или потому, что вывод результата происходил неверно.

В задании №8требовалось реализовать задание №7, но так, чтобы передача результата осуществлялась через pipe. Задание необходимо было выполнить в среде Linux.

Основной целью работы являлось изучение каналов в Linux.

Опыт работы с pipe в среде Windows уже имелся, но в среде Linux опыта работы с ними не было, поэтому информацию по работе с pipe в Linux пришлось изучать самостоятельно.

Для реализации передачи информации через pipe использовались функции write и read. При работе с pipe в среде Linux, запись производится в pipe[1], а чтение происходит из pipe[0], что отличается от pipe в среде Windows.

Основная трудность работы заключалась в отсутствии опыта использования pipe в среде Linux, что часто происходило зацикливание из-за того, что создаваемые процессы не были синхронизированы между собой.

В ходе выполнения всех заданий был получен ценный опыт работы в среде Linux, опыт использования процессов и потоков, опыт работы с pipe.

По прохождению практики был заполнен дневник, в котором кратко описано, в какие даты были выполнены задания и как проходила работа над ними. В дневнике указано количество часов, затраченных на выполнение всех практических работ.

В качестве компилятора для кодов в среде Linux использовался g++ Visual Studio Code, а для Windows – Microsoft Visual Studio. использовались встроенные функции g++.

Для отладки кодов в среде Windows использовались встроенные функции Microsoft Visual Studio, такие как расстановка точек остановки и др.

В ходе прохождения практики был получен опыт работы с процессами, потоками и структурами сред в средах Linux и Windows. Был получен ценный опыт работы с языком C и C++д в среде Linux.

**ЗАКЛЮЧЕНИЕ**

Результатом освоения программы практики стало овладение разработкой программных модулей программного обеспечения для компьютерных систем.

За период практики были получены навыки разработки алгоритма поставленной задачи и его реализации, разработки кода программного продукта, использования инструментальных средства на этапе отладки программы.

Приобретены знания создания отдельных программных модулей, выполнения их отладки и тестирования.

После прохождения практики стала понятна сущность и социальная значимость будущей профессии, был улучшен навык организации собственной деятельности, были освоены навыки выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество, принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность, осуществлять поиск и использование информации, необходимой для эффективного выполнения задач профессионального и личностного развития, работать в коллективе и в команде.

ПРИЛОЖЕНИЕ

№3

#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)

{

//старое имя

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)

{

//новое имя

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;

}

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); // 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);

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

empty\_files\_counter++;

} 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<LPTSTR> paths;

do

{

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());

rec\_dir(paths.back());

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

FindClose(hdl); // поиск окончен1

}

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

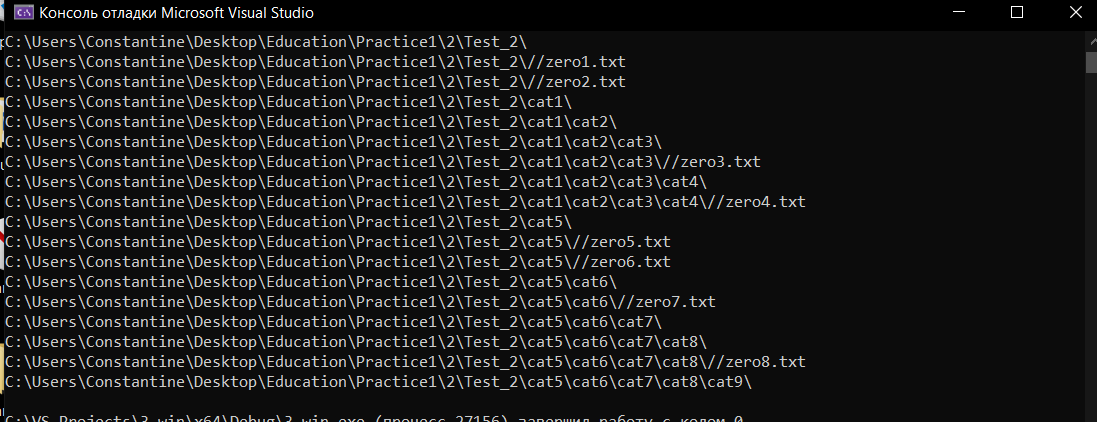
{

setlocale(LC\_ALL, "Russian");

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

return 0;

}



№5

#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[])

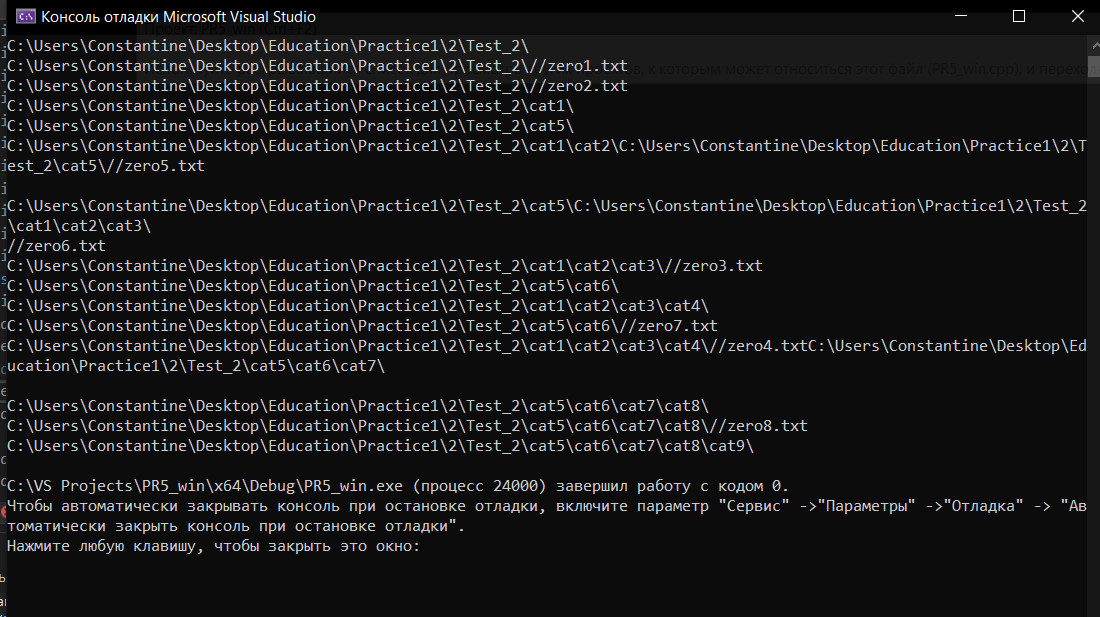
{

setlocale(LC\_ALL, "Russian");

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

return 0;

}



№8

#include <dirent.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <string>

#include <memory>

#include <vector>

#include <stdlib.h>

#include <malloc.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <limits.h>

#include <errno.h>

#include <pthread.h>

#include <sys/mman.h>

#define PATH "/home/consta/practice1/1/Test1/"

#define BUFFER\_SIZE 256

#include <stdio.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

bool is\_pseudodirectory(struct dirent \*ent)

{

return strcmp(ent->d\_name, (".")) == 0 || strcmp(ent->d\_name, ("..")) == 0;

}

void get\_old\_name(char \*path, struct dirent \*ent, char \*old\_file\_name)

{

sprintf(old\_file\_name, "%s%s", path, ent->d\_name);

}

void get\_new\_path(char \*path, struct dirent \*ent, char \*new\_path)

{

sprintf(new\_path, "%s%s%s", path, ent->d\_name, "/");

}

bool is\_directory(char \*path\_, struct stat \*st)

{

stat(path\_, st);

return S\_ISDIR(st->st\_mode);

}

struct DirInfo

{

int file\_count;

char path[BUFFER\_SIZE];

};

void UpdateMin(DirInfo \*old\_min, DirInfo \*new\_info)

{

if (old\_min->file\_count <= new\_info->file\_count)

return;

old\_min->file\_count = new\_info->file\_count;

strcpy(old\_min->path, new\_info->path);

}

int fd[2];

void count\_files(char \*path)

{

DIR \*dir = opendir(path);

DirInfo dir\_info;

strcpy(dir\_info.path, path);

dir\_info.file\_count = 0;

while (struct dirent \*ent = readdir(dir))

{

struct stat sb;

char file\_name[BUFFER\_SIZE] = "";

get\_old\_name(path, ent, file\_name);

if (stat(file\_name, &sb) == -1)

{

continue;

}

if (is\_pseudodirectory(ent))

{

continue;

}

if (!(sb.st\_mode & S\_IFREG))

{

continue;

}

if (!(sb.st\_mode & S\_IRUSR))

{

continue;

}

// if ((sb.st\_mode & S\_IRWXU))

// {

// continue;

// }

printf("файл на чтение");

dir\_info.file\_count += 1;

}

closedir(dir);

write(fd[1], &dir\_info, sizeof(DirInfo));

}

void \*rec\_dir\_helper(void \*path);

void rec\_dir(char \*path)

{

count\_files(path);

DIR \*dir = opendir(path);

std::vector<std::unique\_ptr<char[]>> paths;

std::vector<pthread\_t> threads;

while (struct dirent \*ent = readdir(dir))

{

struct stat sb;

char dir\_path[BUFFER\_SIZE] = "";

get\_old\_name(path, ent, dir\_path);

if (!(is\_directory(dir\_path, &sb)))

{

continue;

}

if (is\_pseudodirectory(ent))

{

continue;

}

paths.emplace\_back(new char[BUFFER\_SIZE]());

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

threads.emplace\_back();

pthread\_create(&threads.back(), NULL, rec\_dir\_helper, (void \*)paths.back().get());

}

for (int i = 0; i < threads.size(); i += 1)

{

pthread\_join(threads[i], NULL);

}

closedir(dir);

}

void \*rec\_dir\_helper(void \*path)

{

rec\_dir((char \*)path);

return NULL;

}

void \*main\_rec\_dir(void \*path)

{

rec\_dir((char \*)path);

close(fd[1]);

return NULL;

}

int main(int argc, char \*argv[])

{

pipe(fd);

DirInfo answer;

answer.file\_count = INT\_MAX;

strcpy(answer.path, "default dir");

pthread\_t rec\_dir\_thread;

pthread\_create(&rec\_dir\_thread, NULL, main\_rec\_dir, (void \*)PATH);

DirInfo buffer;

while (read(fd[0], &buffer, sizeof(DirInfo)))

{

UpdateMin(&answer, &buffer);

}

printf("%s\n", answer.path);

}

