**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе № 3**

**по дисциплине «Программирование»**

Тема: **Обход файловой системы**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3344 |  | Волков А.А. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Освоить работу с файловой системой при помощи языка Си. Изучить рекурсивный подход в алгоритмах. Обучиться обходить дерево файловой системы. Реализовать алгоритм сортировки информации в заданных файлах и записать полученный результат в результирующий файл.

## Задание

Вариант 3.

Дана некоторая корневая директория, в которой может находиться некоторое количество папок, в том числе вложенных. В этих папках хранятся некоторые текстовые файлы, имеющие имя вида <filename>.txt

В каждом текстовом файле хранится одна строка, начинающаяся с числа вида:

<число><пробел><латинские буквы, цифры, знаки препинания> ("124 string example!")

Требуется написать программу, которая, будучи запущенной в корневой директории, выведет строки из файлов всех поддиректорий в порядке возрастания числа, с которого строки начинаются.

## Выполнение работы

Подключаются требуемые заголовочные файлы <stdio.h>, <stdlib.h> (для работы с динамической памятью и функциями открытия, закрытия файлов), <string.h> (для работы со строками), <dirent.h> (для работы с директориями и их содержимым). Вводится макроопределение BUF, которое определяет размер считываемых символов за один раз из файла.

В функции main() задается указатель str\_list, в который нужно записать массив строк, а также целочисленная переменная str\_count, хранящая текущее количество элементов в этом массиве. Вызывается рекурсивная функция check\_dir(), в которую передается путь на директорию, с которой нужно начать обход вглубь файловой системы, указатель на список строк и указатель на str\_count.

Внутри функции check\_dir() с помощью функций из <dirent.h> открывается директория, к которой ведет текущий путь и начинается считывание содержимого этой директории. Если же обнаружена еще одна директория, то check\_dir() вызывается и для нее с обновленным путем. В случае обнаружения файла подходящего формата, расширяется массив строк и вызывается функция check\_file(), в которую передается путь файлу, список строк и указатель на str\_count. Полностью пройденная директория закрывается в конце обхода.

Внутри функции check\_file() с помощью функций из <stdlib.h> открывается файл и при помощи временной строки и функции fgets() из <string.h> считывается записанная в нем строка, после чего она помещается в массив str\_list.

В конце функции main() полученный str\_list сортируется при помощи qsort() и записывается в файл result.txt при помощи fprintf().

**Тестирование**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1. | tmp1.txt: -543 first  tmp2.txt: 1500 thi!rd  tmp3.txt: 754 second | -543 first  754 second:  1500 thi!rd | Файлы расположены внутри корневой директории, внутри которой есть другие директории. Поиск файлов и сортировка происходят успешно. |
| 2. | tmp1.txt: -543 first  tmp2.txt: 1500 thi!rd  tmp3.txt: 754 second  tmp4.txt: 1786 qweruiqweyfkjbasdbfhsdggfsdsdfuiogudosifugoisdufgoisdufgpoisdufpgoiusdfpoigudsfpoigudpsoifgu | -543 first  754 second:  1500 thi!rd  1786 qweruiqweyfkjbasdbfhsdggfsdsdfuiogudosifugoisdufgoisdufgpoisdufpgoiusdfpoigudsfpoigudpsoifgu | Длинные строки корректно обрабатываются. |

## Выводы

Был изучен рекурсивный подход при написании программы, освоена работа с файловой системой при помощи языка Си.

Была разработана программа, которая обходит корневую директории и выполняет поставленную задачу, результат работы программы записывается в файл в нужном формате.

# Приложение А Исходный код программы

Название файла: lb\_3.c

#include <stdio.h>

#include <string.h>

#include <dirent.h>

#include <stdlib.h>

#define BUF 64

int cmp\_by\_number(const void \*a, const void \*b)

{

const char \*f\_line = \*(const char \*\*)a;

const char \*s\_line = \*(const char \*\*)b;

long f\_num = atol(f\_line);

long s\_num = atol(s\_line);

if (f\_num - s\_num > 0)

return 1;

if (f\_num - s\_num < 0)

return -1;

return 0;

}

void check\_file(char \*filename, char \*\*\*str\_list, int \*str\_count)

{

FILE \*file = fopen(filename, "r");

char \*line = (char \*)malloc(BUF);

char \*tmp\_line = (char \*)malloc(BUF);

fgets(line, BUF, file);

while (fgets(tmp\_line, BUF, file)) {

line = (char \*)realloc(line, strlen(line) + strlen(tmp\_line) + 1);

strcat(line, tmp\_line);

}

(\*str\_list)[\*str\_count-1] = line;

free(tmp\_line);

free(filename);

fclose(file);

}

void check\_dir(char \*path, char \*\*\*str\_list, int \*str\_count)

{

DIR \*dir = opendir(path);

if (dir) {

struct dirent \*obj;

while (obj = readdir(dir)) {

char \*obj\_name = obj->d\_name;

if (!strcmp(obj\_name, "..") || !strcmp(obj\_name, ".") || !strcmp(obj\_name, "result.txt")) {

continue;

}

if (obj->d\_type == DT\_DIR) {

char \*new\_path = (char \*)malloc(strlen(path) + strlen(obj->d\_name) + 2);

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

check\_dir(new\_path, str\_list, str\_count);

free(new\_path);

}

if (obj->d\_type == DT\_REG && strstr(obj\_name, ".txt")) {

if (\*str\_count == 0) {

\*str\_list = (char \*\*)malloc((++(\*str\_count)) \* sizeof(char \*));

} else {

\*str\_list = (char \*\*)realloc(\*str\_list, (++(\*str\_count)) \* sizeof(char \*));

}

char \*filename = (char \*)malloc(strlen(path) + strlen(obj\_name) + 2);

sprintf(filename, "%s/%s", path, obj\_name);

check\_file(filename, str\_list, str\_count);

}

}

}

closedir(dir);

}

int main()

{

char \*\*str\_list = NULL;

int str\_count = 0;

check\_dir(".", &str\_list, &str\_count);

qsort(str\_list, str\_count, sizeof(char\*), cmp\_by\_number);

FILE \*res = fopen("result.txt", "w");

for (int i = 0; i < str\_count; i++) {

fprintf(res, "%s\n", str\_list[i]);

free(str\_list[i]);

}

free(str\_list);

fclose(res);

return 0;

}