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

Студентка гр. 3343 Лобова Е. И.

Преподаватель Государкин Я.С.

Санкт-Петербург
2024

Цель работы

Целью работы является освоение работы с рекурсивными функциями и файловой системой, а также ее рекурсивным обходом.

Для достижения поставленной цели требуется решить следующие задачи:

1) Ознакомиться с понятием рекурсии;

2) Освоить написание рекурсивных функций в языке Си;

3) Изучить работу с файловой системой в языке Си;

4) Написать программу для рекурсивного обхода всех файлов в папке в том числе во вложенных папках.

Задание

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

Пример:

root/file.txt: 4 Where am I?

root/Newfolder/Newfile.txt: 2 Simple text

root/Newfolder/Newfolder/Newfile.txt: 5 So much files!

root/Newfolder(1)/Newfile.txt: 3 Wow? Text?

root/Newfolder(1)/Newfile1.txt: 1 Small text

Решение:

1 Small text

2 Simple text

3 Wow? Text?

4 Where am I?

5 So much files!

Ваше решение должно находиться в директории /home/box, файл с решением должен называться solution.c. Результат работы программы должен быть записан в файл result.txt.

Выполнение работы

Для решения задания лабораторной работы была создана структура data с полями char text[100], int n, которые обозначают число и последующую часть строки.

Функции, реализованные в программе:

- Функция *data* read_file(const char *file_name)* открывает файл на чтение, имя которого передано в функцию, считывает данные из него в созданную структуру, закрывает его и возвращает указатель на структуру.
- Функция *void write_to_file(const char *file_name, int* count, data** strings)* открывает файл на запись, имя которого передано в функцию, записывает отсортированные строки в него и очищает память.
- Функция *char *pathcat(const char *path1, const char *path2)* принимает на вход две части пути к файлу или директории. Выделяется память под новую строку и с помощью функции sprintf() в строку соединяются две части пути, что и возвращается.
- Функция *data** list_dir(const char *dir_name, int* count, data** strings)* принимает на вход указатель на имя директории или файла, массив указателей на уже полученные структуры с информацией из файлов и их количество. Выполняется проверка - открыт файл или директория. Если файл, то получается его полное имя с помощью функции *char* pathcat()* и передается в функцию *data* read_file(),* в массив записывается полученный указатель на структуру. Если директория, то так же получается полный путь и вызывается эта же функция, но подается путь этой директории(рекурсия).
- Функция *int compar(const void *a, const void *b)* - компаратор для qsort, которая сравнивает структуры по значению поля int n.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	root/file.txt: 4 Where am I? root/Newfolder/Newfile.txt: 2 Simple text root/Newfolder/Newfolder/Newfile.txt: 5 So much files! root/Newfolder(1)/Newfile.txt: 3 Wow? Text? root/Newfolder(1)/Newfile1.txt: 1 Small text	1 Small text 2 Simple text 3 Wow? Text? 4 Where am I? 5 So much files!	Выходные данные корректны.

Выводы

Были изучены работа с рекурсивными функциями и их написание на языке Си, файловая система и основные функции библиотеки <*dirent.h*> для работы с файлами и директориями. Так же написана программа, реализующая рекурсивный обход по файловой системе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#define BLOCK 100

typedef struct data{
    char text[BLOCK];
    int n;
}data;

int compar(const void *a, const void *b) {
    const data* data_a = *(data**)a;
    const data* data_b = *(data**)b;
    if (data_a->n < data_b->n) return -1;
    if (data_a->n > data_b->n) return 1;
    return data_a->n == data_b->n;
}

data* read_file(const char *file_name){
    char str_file[BLOCK];
    FILE *fin = fopen(file_name,"r");
    data* f = NULL;
    if (fin){
        f = (data*)malloc(sizeof(data));
        fscanf(fin, "%d", &(f->n));
        fgets(str_file, BLOCK, fin);
        strncpy(f->text, str_file, BLOCK);
        fclose(fin);
    }else{
        printf("Failed to open %s file\n", file_name);
    }
    return f;
}
void write_to_file(const char *file_name, int* count, data**
strings){
    FILE *fout = fopen(file_name, "w");
    if (fout){
        for (int i = 0; i<(*count); i++){
            fprintf(fout, "%d%s\n", strings[i]->n, strings[i]->text);
            free(strings[i]);
        }
        free(strings);
        fclose(fout);
    } else{
        printf("Failed to open %s file\n", file_name);
    }
}

char *pathcat(const char *path1, const char *path2){
    int res_path_len = strlen(path1) + strlen(path2) + 2;
    char *res_path = malloc(res_path_len * sizeof(char));
```

```
        sprintf(res_path, "%s/%s", path1, path2);
        return res_path;
    }

    data** list_dir(const char *dir_name, int* count, data** strings){
        DIR *dir = opendir(dir_name);
        if(dir){
            struct dirent *de = readdir(dir);
            while (de){
                if(de->d_type == DT_REG && strstr(de->d_name, ".txt") !=
NULL && strstr(de->d_name, "result.txt") == NULL){
                    char *name_file = pathcat(dir_name, de->d_name);
                    data* buf = read_file(name_file);
                    if (buf){
                        strings[(*count)++] = buf;
                    }
                    free(name_file);
                }else if (de->d_type == DT_DIR && strcmp(de->d_name,
".") != 0 && strcmp(de->d_name, "..") != 0){
                    char *new_dir = pathcat(dir_name, de->d_name);
                    list_dir(new_dir, count, strings);
                    free(new_dir);
                }
                de = readdir(dir);
            }
            closedir(dir);
        }else
            printf("Failed to open %s directory\n", dir_name);

    }
    int main(){
        int count;
        count = 0;
        data** strings = (data**)malloc(50*BLOCK*sizeof(data*));
        list_dir(".", &count, strings);
        qsort(strings, count, sizeof(data*), compar);
        write_to_file("result.txt", &count, strings);
        return 0;
    }
```