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 Сербиновский Ю.М.

Преподаватель Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Изучение методов работы с файловой системой на языке Си. Написание программы для обхода директории и поиска файлов.

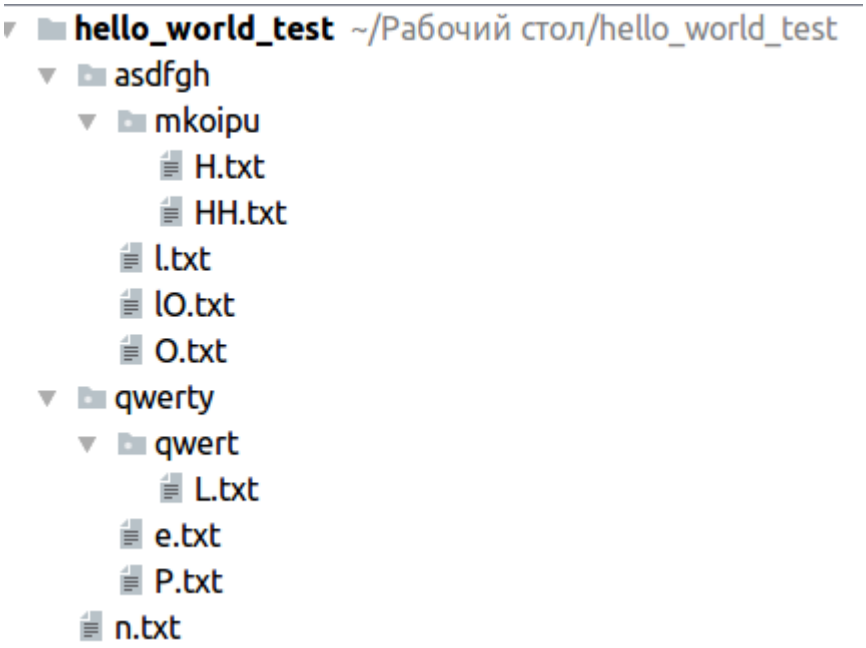
Задание

Вариант 4

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 качестве имени файла используется символ латинского алфавита.

На вход программе подается строка. Требуется найти и вывести последовательность полных путей файлов, имена которых образуют эту строку.

Пример



Входная строка:

HeLlO

Правильный ответ:

hello_world_test/asdfgh/mkoipu/H.txt

hello_world_test/qwerty/e.txt

hello_world_test/qwerty/qwert/L.txt

hello_world_test/asdfgh/l.txt
hello_world_test/asdfgh/O.txt

! Регистрозависимость
! Могут встречаться файлы, в имени которых есть несколько букв и эти файлы использовать нельзя.
! Одна буква может встречаться один раз.

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****. Ваша программа должна обрабатывать директорию, которая называется* ***tmp****.*

Выполнение работы

char* requiredFile()

Данная функция формирует шаблон следующего искомого файла за счет считывания символа с помощью getchar() и конструирования строки посредством spritf(). Если считан символ переноса строки, то программа завершает работу.

int recTravel(const char* dirname, const char* file, FILE* file)

Данная функция отвечает за рекурсивный обход директорий. Функция возвращает 1, если искомый файл был найден, или 0, если произошло обратное. В самом начале открывается файл и проверяется успешность действия, затем начинанается цикл while, который работает до того момента, пока программа ни дойдет до конца ветки директории, то есть не сможет открыть директорию. Если цикл завершился, то программа закрывает данную директорию и исследует предыдущую, и так до тех пор пока не найдется искомый файл или не будут исследованы все директории.

Создается ссылка на объект struct dirent* entity, хранящий информацию о текущей директории. В цикле while в первую очередь проверяется, что директория не является «ссылкой» на текущую или родительскую директорию («.» или «..»). Далее, если текущий объект — файл, то его путь записывается в result, функция последовательно возвращает 1 на всех уровнях рекурсии. Если текущий объект — директория, то название путь обновляется и происходит еще один вызов рекурсивной функции.

int main()

В начальный момент текущяя директория - «/.tmp», открывается файл result.txt, куда записывается результат работы программы. В цикле while результат requiredFile записывается в file, затем если посредством рекрсивного поиска исскомый файл был найден, то цикл продолжает работу, иначе выводится ошибка «Required file is missing».

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	HeLlO	hello_world_test/asdfgh/mkoipu/H.txt hello_world_test/qwerty/e.txt hello_world_test/qwerty/qwert/L.txt hello_world_test/asdfgh/l.txt hello_world_test/asdfgh/O.txt

Выводы

Были изучены методы работы с файловой системой на языке Си. Была написанна программа рекурсивного поиска файлов внутри файловой директории.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#define _GNU_SOURCE


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <dirent.h>


char* requiredFile(FILE* result);
int recTravel(const char* dirname, const char* file, FILE* result);


int main() {
    char dirname[256] = "/.tmp";
    char* file;
    FILE* result;
    result = fopen("result.txt", "w");
    if(result == NULL)
        fprintf(stderr, "Failed to open file.");

    while (1)
    {
        file = requiredFile(result);
        if (recTravel(dirname, file, result) == 0) {
            fprintf(stderr, "Required file is missing.");
            break;
        }
    }

    free(file);
    fclose(result);
    return 0;
}


char* requiredFile(FILE* result){
    char letter = getchar();
    char* file = malloc(sizeof(char)*256);
    if(file == NULL)
        fprintf(stderr, "Allocation failed.");
    if (letter != '\n')
        sprintf(file, "%c.txt", letter);
    else {
        free(file);
        fclose(result);
        exit(0);
    }
    return file;
```

```
    }


    int recTravel(const char* dirname, const char* file, FILE* result) {
        DIR* dir = opendir(dirname);
        if (dir == NULL) {
            return 0;
        }

        struct dirent* entity;
        while ((entity = readdir(dir)) != NULL) {
            if (strcmp(entity->d_name, ".") == 0 || strcmp(entity->d_name,
"..") == 0) {
                continue;
            }

            if (strcmp(file, entity->d_name) == 0) {
                fprintf(result, "%s/%s\n", dirname, entity->d_name);
                return 1;
            }

            if (entity->d_type == DT_DIR) {
                char path[512];
                sprintf(path, "%s/%s", dirname, entity->d_name);
                if(recTravel(path, file, result) == 1) {
                    closedir(dir);
                    return 1;
                }
            }
        }

        closedir(dir);
    }
```