**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 |  | Мурдасов М.К. |
| 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.

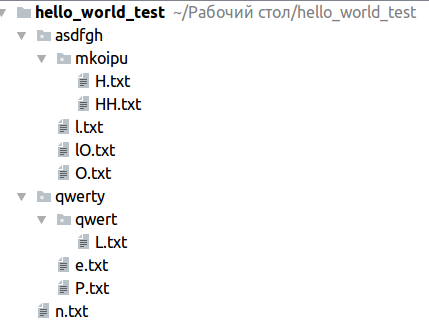
## Задание

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

*void scan\_dir(char\* base\_dir, FILE\* file, char target)*.

В первую очередь был подключен заголовочный файл *<dirent.h>* для работы с директориями.

Была написана функция *scan\_dir()*, принимающая в качестве аргументов путь к исследуемой директории – *base\_dir*, указатель на объект файла для записи результатов - *file* и искомое имя файла – *target*. Внутри функции объявлены следующие переменные: *path* – строка для записи пути к текущему подкаталогу/файлу для дальнейшего обхода или записи результата, *current\_dir* – указатель на структуру *dirent*, *dir* и *is\_dir* – указатели на структуру *DIR*.

Вначале с помощью функции *opendir()* открывается поток исследуемого каталога и в переменную *dir* сохраняется указатель на структуру *DIR*, содержащую информацию о каталоге.

Запускается цикл, который перебирает каждую следующую запись в каталоге с помощью *readdir()*, пока записи не закончатся. *readdir()* в качестве аргумента принимает указатель на структуру *DIR* открытого ранее каталога и возвращает указатель на структуру *dirent*, хранящей информацию о рассматриваемой записи. Если текущая запись является родительской либо корневой директорией, то они пропускаются, чтобы избежать бесконечной рекурсии. Далее в переменную *path* сохраняется полный путь до текущей записи. С помощью ранее созданного указателя на *DIR* – *is\_dir* выполняется проверка на файл. Функция пытается открыть текущую запись по получившемуся в *path* пути и записать указатель в *is\_dir*. Если не получилось, значит запись является файлом и в случае совпадения имени этой записи с *target* ее полный путь записывается в *file*. Если запись получилось открыть как директорию, то проверка не пройдена, функция просто закрывает ее и передает в следующий вызов *scan\_dir()*. Это реализовано для того, чтобы в результат вместо пути к файлу ошибочно не попал путь к директории с таким же именем.

В конце цикла производится вызов *scan\_dir()* c получившейся переменной *path* в качестве первого аргумента. Также стоит упомянуть, что функция прекращает работу в случае, если директорию по переданному пути невозможно открыть. Поэтому если передан путь к файлу, то функция не будет выполнять никаких действий.

В конце работы функция закрывает директорию с помощью *closedir()*.

Таким образом функция *scan\_dir()* рекурсивно производит обход директории и всех ее подкаталогов. При этом избегается обход корневой и родительской директорий, а также учитывается вероятность совпадения имен директорий и файлов.

*int main()*. С помощью функции *fopen()* открывается файл по указанному пути, в переменную *result* записывается указатель на структуру *FILE*, который используется для идентификации потока и выполнений операций с файлом. В качестве второго аргумента используется параметр *«w»* для открытия файла для записи, при этом перезаписав содержимое. Если файл по заданному пути отсутствует, то *fopen()* автоматически создает его. Далее считывается входная строка и для каждого ее символа выполняется вызов *scan\_dir()* с соответствующим аргументом *target*. После записи результатов в файл, он закрывается с помощью *fclose()*.

**Тестирование**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 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 | Корректно |

## Выводы

Были изучены способы работы с файловой системой на языке Си. С использованием полученных навыков была написана программа, проводящая обход директории и сохраняющая в текстовый файл пути искомых файлов.

# Приложение А Исходный код программы

Название файла: Murdasov\_Mikhail\_lb3.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <dirent.h>

void scan\_dir(char\* base\_dir, FILE\* file, char target){

    char path[3000];

    struct dirent\* current\_dir;

    DIR\* dir = opendir(base\_dir);

    DIR\* is\_dir;

    if(dir == NULL) return;

    while((current\_dir = readdir(dir)) != NULL){

        if(strcmp(current\_dir->d\_name, ".") == 0 || strcmp(current\_dir->d\_name, "..") == 0){

            continue;

        }

        strcpy(path, base\_dir);

        strcat(path, "/");

        strcat(path, current\_dir->d\_name);

        char\* file\_name = (char\*)strdup(current\_dir->d\_name);

        if(file\_name == NULL){

            fprintf(stderr, "[ ERROR ]: Memory allocation error.");

        }

        if((is\_dir = opendir(path)) == NULL && strcmp(strtok(file\_name, "."), &target) == 0){

            fprintf(file, "%s\n", path);

        }else{

            closedir(is\_dir);

        }

        free(file\_name);

        scan\_dir(path, file, target);

    }

    closedir(dir);

}

int main(){

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

    if(result == NULL){

        fprintf(stderr, "[ ERROR ]: Failed to open file.");

    }

    char\* input = (char\*)malloc(sizeof(char)\*1000);

    if(input == NULL){

        fprintf(stderr, "[ ERROR ]: Memory allocation error.");

    }

    scanf("%s", input);

    for(int i = 0; i<strlen(input); i++){

        scan\_dir("./tmp", result, input[i]);

    }

    fclose(result);

    free(input);

    return 0;

}