МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Математического Обеспечения и Применения ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Программирование»
Тема: Обход файловой системы
Вариант 4

Студент гр. 0382 _____ Кондратов Ю.А.
Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Изучение основных принципов работы с файловыми директориями операционной системы Linux на языке программирования Си. Реализация рекурсивного обхода файлового дерева.

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 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.

Основные теоретические положения.

Для работы с файлами и директориями в языке Си используются соответственно файловые дескрипторы и дескрипторы директорий.

Функции, использующиеся для получения дескрипторов:

1) FILE * fopen (const char * fname, const char * mode) - функция открывает файл, имя которого указано в параметре fname и связывает его с потоком, который может быть идентифицирован для выполнения различных операций с файлом. Операции с потоком, выполнение которых разрешено определяются параметром modeopen.

Возможные операции:

- «r» - режим открытия файла для чтения. Файл должен существовать.
- «w» - режим создания пустого файла для записи. Если файл с таким именем уже существует его содержимое стирается, и файл рассматривается как новый пустой файл.

- «a» - дописать в файл. Операция добавления данных в конец файла. Файл создается, если он не существует.

Помимо основных трёх существуют также комбинированные режимы.

Для использования функции fopen необходимо подключение заголовочного файла stdio.h.

2) DIR *opendir (const char *name) — функция открывает поток каталога, соответствующий каталогу name, и возвращает указатель на этот поток. Поток устанавливается на первой записи в каталоге.

Для использования этой функции необходимо подключение заголовочных файлов sys/types.h и dirent.h.

В случае ошибок обе функции возвращают NULL.

По завершении работы с файлами и директориями необходимо их «закрыть» с помощью функций соответственно fclose (char *name) и closedir (char *name).

Функция предназначенная для чтения каталога - struct dirent *readdir (DIR *dir). Возвращает указатель на следующую запись каталога в виде структуры dirent. На вход принимает дескриптор директории. Возвращает NULL по достижении последней записи, или если была обнаружена ошибка.

Выполнение работы.

Общая структура программы такова: сначала считывается входная строка, далее для каждого символа этой строки в директории tmp ищется файл с соответствующим названием, с помощью функции get_file_path (её реализация описана далее) получается путь к файлу относительно текущей директории, этот путь записывается в файл result.txt.

Входная строка считывается в массив символов input размера MAX_I_SIZE (именованная константа, её значение — 128) следующим образом:

```
char input[MAX_I_SIZE];
fgets(input, MAX_I_SIZE, stdin);
input[strlen(input) - 1] = '\0';
```

Функция void get_file_path(char *file_name, char *dir_name, char *path) реализована следующим образом:

1. Сначала производится присваивание переменной DIR *dir дескриптора директории, названии которой передано в функцию при помощи аргумента dir_name.
2. Далее при помощи цикла while по переменной struct dirent *cur_elem, которой присвоен первый указатель на запись каталога, производится обработка всех элементов в каталоге.
3. Если очередной элемент является файлом, то его имя при помощи функции strcmp проверяется на соответствие значению аргумента file_name. Если имя файла и значение file_name идентичны, то в переменную char *path (является аргументом функции) записывается путь к файлу:

```
strcpy(path, "./");
strcat(path, dir_name);
strcat(path, "/");
strcat(path, file_name);
return;
```

4. Если очередной элемент является директорией, но при этом его название не «.» и не «..», то сначала производится создание строки, содержащей путь к этой директории относительно той, из которой запускалась программа, а затем рекурсивный вызов функции get_file_path:

```
char new_dir[strlen(dir_name) + strlen(cur_elem->d_name) + 2];
new_dir[0] = '\0';
```

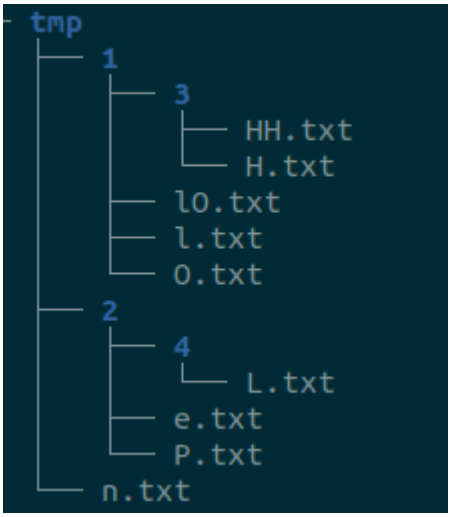
```
strcat(new_dir, dir_name);
strcat(new_dir, "/");
strcat(new_dir, cur_elem->d_name);
get_file_path(file_name, new_dir, path);
```

В качестве аргумента path в функцию get_file_path передаётся массив символов path из функции main. Именно в этот массив и происходит запись пути к файлу. После того как путь получен, он записывается в файл result.txt:

```
char path[MAX_PATH_SIZE];
get_file_path(file_name, dir_name, path);
strcat(path, "\n");
fputs(path, result);
```

Тестирование.

Директория tmp созданная для тестирования имеет следующий вид:



Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

Входные данные	Содержимое файла result.txt	Комментарии
HeLlO	./tmp/1/3/H.txt ./tmp/2/e.txt ./tmp/2/4/L.txt ./tmp/1/l.txt ./tmp/1/O.txt	Программа работает правильно

Выводы.

В ходе работа были изучены основные принципы работы с файловыми директориями операционной системы Linux на языке Си. Реализован поиск определённых файлов в директории при помощи рекурсивного обхода каталога. Для получения пути к файлам была реализована функция get_file_path, работа с директориями и файлами осуществлялась при помощи функция заголовочных файлов stdio.h и dirent.h.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ФАЙЛОВ ПРОЕКТА

1. Название файла: solution.c

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

#define MAX_I_SIZE 128
#define MAX_PATH_SIZE 512


void get_file_path(char *file_name, char *dir_name, char *path){

    DIR *dir = opendir(dir_name);
    if (!dir) return;

    struct dirent *cur_elem = readdir(dir);
    while (cur_elem){
        if (cur_elem->d_type == DT_REG){
            if (!strcmp(cur_elem->d_name, file_name)){
                strcpy(path, "./");
                strcat(path, dir_name);
                strcat(path, "/");
                strcat(path, file_name);
                return;
            }
        }
        if (cur_elem->d_type == DT_DIR &&
             strcmp(cur_elem->d_name, ".") &&
             strcmp(cur_elem->d_name, "..")) {
             char   new_dir[strlen(dir_name)   +   strlen(cur_elem-
>d_name) + 2];
            new_dir[0] = '\0';
            strcat(new_dir, dir_name);
            strcat(new_dir, "/");
            strcat(new_dir, cur_elem->d_name);
            get_file_path(file_name, new_dir, path);
        }
        cur_elem = readdir(dir);
    }
    closedir(dir);
}

int main() {
    char *dir_name = "tmp";
    char input[MAX_I_SIZE];
    fgets(input, MAX_I_SIZE, stdin);
    input[strlen(input) - 1] = '\0';

    FILE *result = fopen("result.txt", "w");
    char file_name[] = {' ', '.', 't', 'x', 't', '\0'};
```

```
    for (int i = 0; i < strlen(input); i++){
        file_name[0] = input[i];
        char path[MAX_PATH_SIZE];
        get_file_path(file_name, dir_name, path);
        strcat(path, "\n");
        fputs(path, result);
    }
    fclose(result);
    return 0;
}
```