**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. 3384 _____ Рудаков А.Л.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

**Цель работы.**

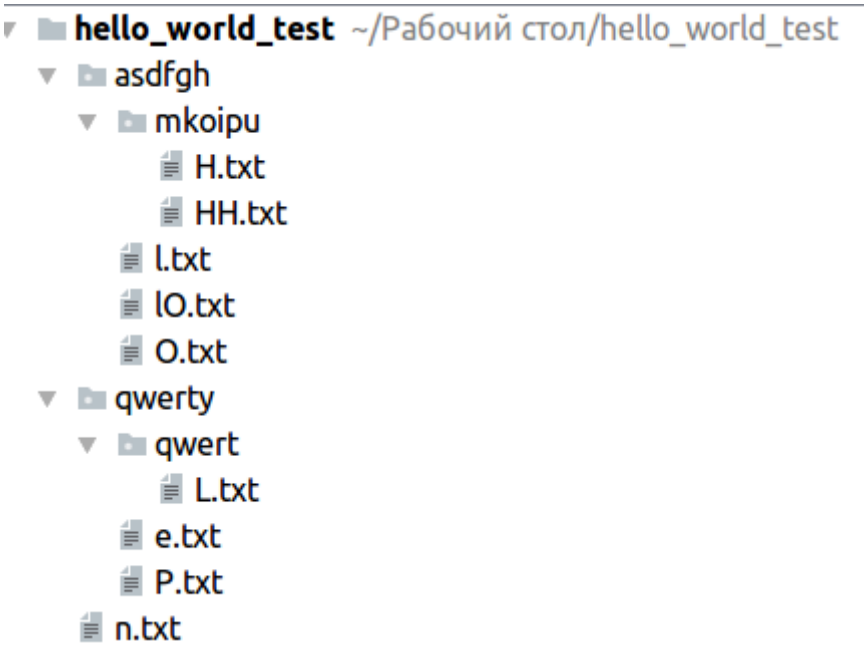
Изучить теорию работы с файловой системой посредством кода на языке С. Получить практические навыки путем написания программного кода.

**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.

**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.**

В начале работы была написана функция *int main()*. Внутри нее был создана строка *directory* с названием обрабатываемой директории *"./tmp"* и создан указатель на объект типа *DIR *dp*, благодаря которому будет осуществляться проход по директории *directory*. Открытие потока директории происходит при помощи функции *dp=opendir(directory)*. Далее создается указатель на структуру *nameList* (которая будет описана ниже), которая создается при помощи функции *createPath()* и будет использоваться как линейный список. Далее считывается входная строка в *input*. После чего

открывается файл для записи *result.txt*, указатель на который записан в *output_file*. Далее происходит проход в цикле по строке *input*, где для каждой буквы ищется нужный путь к файлу при помощи функции *find_file(input[i],dp,file_path,flag,output_file)*. Далее поток директории переоткрывается.

В программе использована структура *struct nameLis*t, которая при помощи ключевого слова *typedef* переименовывает её в *nameList*. Полями структуры являются строка *name* — название директории или файла, указатель на следующую структуру *next* и указатель на предыдущую структуру *previous*.

Далее идет функция *nameList *createPath(char *name, nameList *next, nameList *previous)*, которая принимает название текущего элемента, указатель на следующий элемент и указатель на предыдущий. Функция создает и возвращает указатель на структуру *nameList*, с поданными параметрами.

Далее идет функция *nameList *appendPath(char *name, nameList *next, nameList *previous)*, которой подается название текущего элемента, указатель на следующий и указатель на предыдущий. Внутри создается новая структура, с поданными параметрами, а предыдущему элементу в поле next записывается указатель на созданную структуру. После чего возвращается созданная структура.

Далее идет функция *nameList *removeDir(nameList *directory)*, которой подается указатель на текущий элемент списка. Он удаляет его и очищает память, а также в поле *next* предыдущего элемента записывает *NULL*, после чего возвращает указатель на предыдущий элемент.

После этого написана функция *char *create_path(nameList *file_path)*, в которую подается указатель на элемент линейного списка. Внутри функции указатель меняется на указатель первого элемента списка, после чего происходит проход про линейному списку с целью создания пути к файлу в строчном представлении. В конце возвращается строка.

Последней написана функция *void find_file(char letter, DIR *dp, nameList *file_path, int *flag, FILE * output_file)*, которой подается буква, файл с названием которой нужно найти, указатель на директорию, указатель на текущий элемент списка, флаг и указатель на файл в который производится запись. Внутри создается указатель на структуру *dirent struct dirent *dirp*. Далее инициализирована переменная *reegex* типа *regex_t*. После этого инициализирована переменная *value* типа *int*, которой присваивается значение выполнения функции *regcomp* — 0, если успешно, значение макроса *REG_EXTENDED*, если не успешно. В функции *regcomp(&reegex, "(.+)\\.txt$",REG_EXTENDED)* в *reegex* записывается регулярное выражение. Далее создается массив структур *regmatch_t matches[2],* и инициализируется дополнительный флаг *flag1*. Далее идет цикл *while*, выполняющийся, пока оба флага равны 0. *flag* может стать 1, если будет найден файл, *flag1* может стать 1, если будет достигнут конец считываемой директории. При каждой итерации цикла происходит считывание следующего элемента директории в *dirp*. Далее идет проверка на конец директории. Если проверка пройдена, проверяется на несоответствие с *"."* и *".."* - текущей директорией и директорией прародителем соответственно. Если проверка пройдена, то при помощи функции *value=regexec(&reegex,dirp→d_name,2,matches,0)* проверяется схожесть названия файла с регулярным выражением. Если оно подходит, то значение *value == 0*, после чего проверяется на то, что длина группы равна 1, чтобы в названии файла была один символ. Если это так, то если символ названия файла равен искомому символу, то название файла добавляется в линейный список при помощи функции *appendPath()*, После чего происходит вывод в файл пути при помощи f*printf(output_file,"%s\n",create_path(file_path))*, значение *flag* меняется на 1. Если значение *value* не равно 0, значит это не файл *.txt* а директория, в таком случае имя директории добавляется в линейный список благодаря *appendPath()*, после чего создается путь в строчном виде через функцию *create_path()*, чтобы открыть директорию по этому пути через

функцию *opendir()* в указатель на элемент типа *DIR *dp1*. После чего вызывается функция *find_file(letter,dp1,file_path,flag,output_file)*. после ее исполнения директория закрывается при помощи *closedir()* и удаляется из линейного списка благодаря *removeDir()*.

Разработанный программный код см. в приложении А.

**Выводы.**

Была изучена теория работы с файловой системой посредством кода на языке С. В работе были использованы функции для работы с директориями, такие как opendir(), readdir(), closedir().

## ПРИЛОЖЕНИЕ А

## ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: solution.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <regex.h>

#define BLOCK 20

typedef struct nameList{
    char *name;
    struct nameList *next;
    struct nameList *previous;
} nameList;

nameList *createPath(char *name, nameList *next, nameList *previous){
    nameList *new_dir=(nameList *)malloc(sizeof(nameList));
    new_dir->name=name;
    new_dir->next=next;
    new_dir->previous=previous;
    return new_dir;
}

nameList *appendPath(char *name, nameList *next, nameList *previous){
    nameList *new_dir=(nameList *)malloc(sizeof(nameList));
    new_dir->name=name;
    new_dir->next=next;
    new_dir->previous=previous;
    new_dir->previous->next=new_dir;
    return new_dir;
}

nameList *removeDir(nameList *directory){
    directory=directory->previous;
    free(directory->next);
    directory->next=NULL;
    return directory;
}

char *create_path(nameList *file_path){
    while (file_path->previous!=NULL){
        file_path=file_path->previous;
    }

    char *output=NULL;
    int size=0;

    while (file_path!=NULL){
        if (output==NULL){
            size=strlen(file_path->name)+2;
            output=(char *)malloc(sizeof(char)*size);
```

```
            strcpy(output,file_path->name);
        } else {
            size=strlen(output)+strlen(file_path->name)+2;
            output=(char *)realloc(output,sizeof(char)*size);
            strcat(output,file_path->name);
        }
        output[size-2]='/';
        output[size-1]='\0';
        file_path=file_path->next;
    }

    output[size-2]='\0';
    return output;
}

void find_file(char letter, DIR *dp, nameList *file_path, int *flag,
FILE * output_file){
    struct dirent *dirp;

    regex_t reegex;
    int value = regcomp(&reegex, "(.+)\\.txt$",REG_EXTENDED);
    regmatch_t matches[2];

    int flag1=0;

    while(flag1==0 && *flag==0){
        dirp=readdir(dp);
        if (dirp==NULL){
            flag1=1;
            break;
        }

        if ((strcmp(dirp->d_name, ".") != 0 && strcmp(dirp->d_name,
"..") != 0)){
            value=regexec(&reegex,dirp->d_name,2,matches,0);
            if (value==0){
                if ((matches[1].rm_eo-matches[1].rm_so)==1){
                    if (letter==dirp->d_name[0]){

file_path=appendPath(dirp->d_name,NULL,file_path);

fprintf(output_file,"%s\n",create_path(file_path));
                        *flag=1;
                    }
                }
            } else{
                file_path=appendPath(dirp->d_name,NULL,file_path);

                char *directory=create_path(file_path);
                DIR *dp1;

                dp1=opendir(directory);
                find_file(letter,dp1,file_path,flag,output_file);

                closedir(dp1);
                file_path=removeDir(file_path);
```

```
            }
        }
    }
}

int main(){
    DIR *dp;
    char *directory=(char *)malloc(sizeof(char)*5);
    directory="./tmp";
    dp=opendir(directory);

    nameList *file_path=createPath(directory, NULL, NULL);

    char *input=malloc(sizeof(char)*BLOCK);
    char c='0';
    int count=0;

    while ((c!='\n') && (c!=EOF)){
        c=getchar();
        if (count%BLOCK==BLOCK-1){
            input=(char *)realloc(input,sizeof(char)*(count+1+BLOCK));
        }
        input[count]=c;
        count++;
    }
    count--;
    input[count]='\0';

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

    int f=0;
    int *flag=&f;

    for (int i=0; i<count; i++){
        find_file(input[i],dp,file_path,flag,output_file);
        closedir(dp);
        *flag=0;
        dp=opendir(directory);
    }
}
```