

**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. 0382

Преподаватель

\_\_\_\_\_  
\_\_\_\_\_

Литягин С.М.

Берленко Т.А.

Санкт-Петербург

2021

## **Цель работы.**

Изучение функций для работы с файловой системой.

## **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

## **Основные теоретические положения.**

Для работы с деревом файловой системы используется библиотека *dirent.h*;

Для получения доступа к содержимому некоторой директории используется функция: *DIR\* opendir(const char\* dirname)*; (может вернуть пустой указатель). Тип *DIR* представляет собой поток содержимого директории.

Чтобы получить очередной элемент этого потока используется функция: *struct dirent\* readdir(dirname)*;

Поля структуры *struct dirent*:

- *char d\_name[256]*
- *ino\_t d\_ino*
- *off\_t d\_off*
- *unsigned short d\_reclen*
- *unsigned char d\_type*

После завершения работы с директорией ее нужно закрыть. Функция: *closedir(DIR\* dirname)*;

Для работы с файловой системой используется библиотека *stdio.h*;

Функция открытия файла: *FILE\* fopen(const char\* fname, const char\* modeopen)*; (*fname* – имя или путь к файлу, *modeopen* – режим доступа к файлу(“r”, “w”, “a” и др.));

Функция закрытия файла: *int fclose(FILE\* filestream)*;

Функция чтения символов из потока: *char\* fgets(char\* string, intnum, FILE\* filestream)*; (*string* – указатель на массив типа *char*, куда сохраняются записанные символы, *intnum* – максимальное количество считываемых символов, *filestream* – поток, из которого считываются символы);

Функция записи строки в поток: *int fputs(const char\* string, FILE\* filestream)*; (*string* – указатель на символьную константу, *filestream* – поток, куда записываются символы);

Функции записи и считывания из файла: *int fprintf(...), int fscanf(...)*; (аналогично обычным *printf* и *scanf*, но первый аргумент – *FILE\* filestream*);

И др. функции.

### **Выполнение работы.**

Функция *char\*\* listDir(const char\* path, char\*\* text, int\* index)*:

Аргументы функции: *path* – название директории, *text* – массив, куда будут сохраняться строки из файлов в директории, *index* – счетчик для *text*.

Объявляем вспомогательный символьный массив *new\_path*, в который с помощью функции *strcpy* копируем данные переменной *path*, а затем добавляем с помощью функции *strcat* символ “/”. В дальнейшем *new\_path* нам нужно для определения пути к файлу.

Получаем доступ к содержимому директории *path*: *DIR\* dir = opendir(path)*. Если указатель *dir == NULL*, то функция возвращает *text*, иначе получаем элемент потока: *struct dirent\* de = readdir(dir)*.

Пока указатель *de != NULL*, проверяем, какой тип данных у элемента.

Если поле *d\_type* имеет тип *DT\_REG*, то с помощью функции *strcat* к массиву *new\_path* добавляем данные поля *d\_name*. Затем вызываем функцию

*get\_text(new\_path, text, index)* (описана ниже). Удаляем добавление имя файла из массива *new\_path*.

Если поле *d\_type* имеет тип *DT\_DIR*, а имя файла не “.” или “..”, то к массиву *new\_path* дописывается данные поля *d\_name*. Затем вызывается *listDir(new\_path, text, index)* (таким образом, функция рекурсивно обходит всё внутри директории). Удаляем имя файла из *new\_path*.

В конце итерации цикла получаем новый элемент директории с помощью функции *readdir(dir)*.

Когда *de == NULL*, то все файлы директории будут проверены. Закрываем ее функцией *closedir(dir)*. Возвращаем *text*.

Функция *char\*\* get\_text(const char\* path, char\*\* text, int\* index)*:

Аргументы функции: *path* – путь к файлу, *text* – массив, куда будут сохраняться строки из файлов в директории, *index* – индекс свободного элемента массива *text*.

Открываем файл для чтения с помощью строки *FILE\* f = fopen(path, “r”)*. Если *f == NULL* (т.е. поток пустой), то возвращаем *text*. Иначе считываем символы из потока функцией *fgets()*, и добавляем их к *text[index]* с помощью функции *strcat*. Когда считывать будет нечего – закрываем файл функцией *fclose(f)*. Возвращаем *text*.

Главная функция *int main()*:

Создаем символьный динамический двумерный массив *text* с помощью функции *calloc*. Получаем данные из файлов в *text* с помощью функции *listDir(“root”, text, &index)*. Сортируем данный массив функцией быстрой сортировки *qsort* (не буду описывать функцию компаратора, поскольку единственная интересная в нем вещь – функция *atol* библиотеки *stdlib*; она возвращает целое число, которое содержится в начале строки).

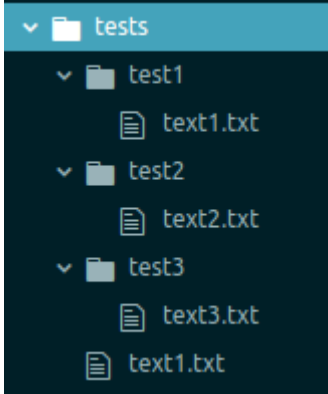
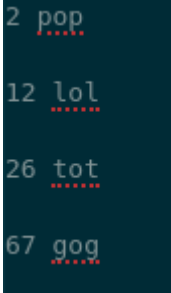
Осталось записать результат в файл “*result.txt*”. Создаем файл с помощью строки *FILE\* f = fopen(“result.txt”, “w”)*. Записываем в файл все непустые элементы массива *text* функцией *fprintf()*. Закрываем файл функцией *fclose()*. Освобождаем выделенную память для *text*.

Разработанный программный код см. в приложении А.

### Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№п/п	Входные данные	Выходные данные	Результат
1			Программа работает верно

### Выводы.

В ходе работы были изучены функции для работы с файловой системой. Также они были применены в ходе выполнения задания, суть которого – создать файл, в котором будут записаны строки из текстовых файлов в порядке возрастания цифр в начале записи.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: solution.c

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

#define buf 4000

int compare(const void * a, const void * b)
{
    const char* aa = *(char**)a;
    const char* bb = *(char**)b;
    long int a1 = atol(aa);
    long int b1 = atol(bb);
    if(a1 > b1) return 1;
    if(a1 < b1) return -1;
    return 0;
}

char** get_text(const char* path, char** text, int* index){
    char s[50];

    FILE* f = fopen(path, "r");
    if(!f){
        return text;
    }

    while(fgets(s, 50, f)){
        strcat(text[*index], s);
    }

    fclose(f);
    return text;
}

char** listDir(const char* path, char** text, int* index){
    char new_path[256] = {0};
    strcpy(new_path, path);
    strcat(new_path, "/");

    DIR* dir = opendir(path);
    if(!dir){
        return text;
    }
    struct dirent* de = readdir(dir);

    while(de){
        if(de->d_type == DT_REG){
            int len = strlen(new_path);
            strcat(new_path, de->d_name);
            get_text(new_path, text, index);
            *index = *index + 1;
        }
    }
}
```

```

        new_path[len] = '\0';
    }

    if((de->d_type == DT_DIR) && strcmp(de->d_name, ".") &&
    strcmp(de->d_name, "..")){
        int len = strlen(new_path);
        strcat(new_path, de->d_name);
        listDir(new_path, text, index);
        new_path[len] = '\0';
    }
    de = readdir(dir);
}

closedir(dir);
return text;
}

int main(){
    int index = 0;
    char** text = calloc(buf, sizeof(char*));
    for(int i = 0; i < buf; i++){
        text[i] = calloc(buf, sizeof(char));
    }

    text = listDir("root", text, &index);
    qsort(text, index, sizeof(char*), compare);

    FILE* f = fopen("result.txt", "w");
    for(int i = 0; i < index; i++){
        if(text[i][0] != '\0') fprintf(f, "%s\n", text[i]);
    }
    fclose(f);

    for(int i = 0; i < buf; i++){
        free(text[i]);
    }
    free(text);
    return 0;
}

```