

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

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Информатика»
Тема: Обход файловой системы

Студент гр. 0382

Ильин Д.А.

Преподаватель

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.

Ход работы:

Для сохранения и последующей сортировки содержания файлов выделяем память под двумерный массив символов `text`.

После чего вызываем функцию `print()` для определенной корневой директории, в данной функции будут рекурсивно перебираться файлы и папки, а также генерироваться пути для соответствующих папок и файлов с помощью функции `strcpy()` и `srtcat()`.

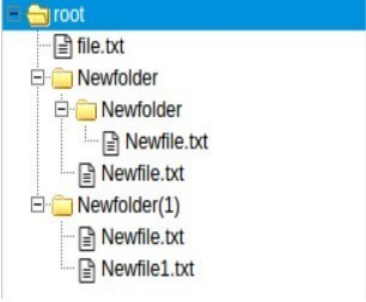
В ходе работы функции, если программа находит папку, то генерируется путь к данной директории и вызов функции `print()` с новым путем. Если программа нашла файл, то происходит также генерация пути, но помимо этого с помощью функции `check()` записываем содержимое файла в массив `text`.

Далее сортируем массив `text` относительно первых чисел в каждом элементе, при помощи функции `qsort()`, сравнивать будем строки относительно чисел, которые идут первыми, при помощи функции `atoi()`.

В конце записываем отсортированный массив `text` в файл `result.txt`, каждый элемент с новой строки, а так же очищаем память.

Тестирование.

Таблица 1 – Результаты тестирования.

№ п/п	Входные данные	Выходные данные	Результат
1.		1 Small text 2 Simple text 3 Wow? Text? 4 4 Where am I? 5 5 So much files!	Программа работает верно

Выводы.

В ходе работы были изучены функции для работы с файловой системой.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.c

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

#define BUF 10000

int count = 0;

int cmp(const void* a, const void* b){
    const char* aa = *((const char**)a);
    const char* bb = *((const char**)b);
    long int num1 = atol(aa);
    long int num2 = atol(bb);
    if(num1 > num2){
        return 1;
    }
    else if(num1 < num2){
        return -1;
    }
    else return 0;
}

void check(const char *path, char** text){
    FILE *fp = fopen(path, "r");

    if(NULL != fgets(text[count++], BUF, fp))

        fclose(fp);
}

void print(const char *dirPath, char** text){
    char next[BUF] = "";
    strcpy(next, dirPath);
    strcat(next, "/");

    DIR *dir = opendir(dirPath);
    struct dirent* de = readdir(dir);
```

```

    if(dir){
        while(de){
            if (de->d_type == DT_REG){
                char file_path[BUF] = "";
                strcat(file_path, next);
                strcat(file_path, de->d_name);
                check(file_path, text);
            }

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

int main(){
    char** text = malloc(sizeof(char*) * BUF);
    int i;

    for (i = 0; i < BUF; i++){
        text[i] = malloc(sizeof(char) * BUF);
        text[i][0] = '\0';
    }

    print("root", text);
    qsort(text, count, sizeof(char*), cmp);
    FILE *result = fopen("result.txt", "w");

    for (i = 0; i < count; i++){
        fprintf(result, "%s\n", text[i]);
    }

    fclose(result);

    for (i = 0; i < BUF; i++){
        free(text[i]);
    }
    free(text);
}

```



```
    return 0;  
}
```