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. 7383 _____ Зуев Д. В.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург
2018

Содержание

1. ЦЕЛЬ РАБОТЫ.....3
2. РЕАЛИЗАЦИЯ.....4
3. ТЕСТИРОВАНИЕ.....5
4. ВЫВОД.....6
ПРИЛОЖЕНИЕ А. ТЕСТОВЫЕ СЛУЧАИ.....7
ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД ПРОГРАММЫ.....8

1. ЦЕЛЬ РАБОТЫ

Цель работы: ознакомиться с рекурсивными функциями, научиться использовать их для работы с деревом файловой системы.

Формулировка задачи:

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

2. РЕАЛИЗАЦИЯ

В файле readfile.c реализованы функции

- comp — функция — компаратор для сортировки строк.
- readfiles — рекурсивная функция, проходящая по файловой системе.

  На вход получает название папки, в которой она будет считывать файлы, указатель на счетчик — количество строк и адрес указателя текущей структуры. Функция считывает текущий объект в директории и в зависимости от типа объекта выполняет действия. Если объект — папка, то функция вызывает себя для этой папки. Если объект — регулярный файл то функция считывает содержимое этого файла и записывает его в структуру.

В файле readfiles.h объявлены вышеперечисленные функции и структура, в которую записывается строка и число.

В функции main.c вызывается readfiles для папки «root», сортируются считанные строки и записываютс в файл «result.txt».

Makefile – make-файл, отвечающий за сборку проекта.

Коды функций представлены в приложении Б.

3. ТЕСТИРОВАНИЕ

Использовался компилятор gcc. Программа запускалась на Ubuntu 16.04 LTS.

Тестовые случаи представлены в приложении А

По результатам тестирования выявлено, что поставленная задача успешно реализована.

4. ВЫВОД

В ходе данной работы были получены навыки работы с рекурсивными функциями, с помощью которых можно совершить обход файловой системы. Обход файловой системы с помощью рекурсивных функций удобен, так как директории имеют одинаковую структуру.

ПРИЛОЖЕНИЕ А. ТЕСТОВЫЕ СЛУЧАИ.

Файловая система	result.txt
./root/Безымянная папка 2/c.txt: 30 zxcv ./root/Безымянная папка 2/a.txt: 34 qwer ./root/Безымянная папка 2/b.txt: 2 adsf ./root/Безымянная папка/a.txt: 12 qwer ./root/Безымянная папка/b.txt: 23 asdf ./root/a.txt: 1 qwer ./root/b.txt: 1 asfd	1 qwer 1 asfd 2 adsf 12 qwer 23 asdf 30 zxcv 34 qwer
./root/file.txt: 4 Where am I? ./root/Newfolder/Newfile.txt: 2 Simple text ./root/Newfolder/Newfolder/Newfile.txt: 5 So much files! ./root/Newfolder(1)/Newfile.txt: 3 Wow? Text? ./root/Newfolder(1)/Newfile1.txt: 1 Small text	1 Small text 2 Simple text 3 Wow? Text? 4 Where am I? 5 So much files!

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД ПРОГРАММЫ

main.c:

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

int main(){
    int count = 0;
                                lines**       array       =
(lines**)malloc(ARR_SIZE*sizeof(lines*));
    readfiles("./root", &count, array);
    qsort(array, count, sizeof(lines*), comp);
    FILE* f = fopen("result.txt", "w");
    for(int i = 0; i<count; i++)
    {
        fprintf(f, "%s",(array[i]->line));
    }
    fclose(f);
    return 0;
}
```

readfiles.c:

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

int comp(const void* a, const void* b) {
    return (*(lines**)a)->num > (*(lines**)b)->num;
}
```

```
lines** readfiles(char *startDir, int* count, lines**
array){
    char nextDir[200]={0};
    strcpy(nextDir, startDir);
    DIR *dir = opendir(startDir);
    struct dirent *de = readdir(dir);
    while(de){
          if(de->d_type == DT_DIR && strcmp(de->d_name,
".") != 0 && strcmp(de->d_name, "..") != 0 && dir){
            int len = strlen(nextDir);
            strcat(nextDir, "/");
            strcat(nextDir,de->d_name);
            array = readfiles(nextDir, count, array);
            nextDir[len] = '\0';
        }
        if(de->d_type == DT_REG ){
            int len = strlen(nextDir);
            strcat(nextDir, "/");
            strcat(nextDir,de->d_name);
            FILE* f = fopen(nextDir, "r");
            *array = (lines*)malloc(sizeof(lines));
                                    (*array)->line   =
(char*)malloc(STR_LEN*sizeof(char));
            fgets((*array)->line, STR_LEN, f);
            (*array)->num = atoi((*array)->line);
            array++;
            (*count)++;
            fclose(f);
            nextDir[len] = '\0';
        }
        de = readdir(dir);
    }
    closedir(dir);
    return array;

}
```

readfiles.h:

```
#pragma once
#define STR_LEN 100
#define ARR_SIZE 3000
```

```
typedef struct lines
{
    int num;
    char* line;
}lines;

int comp(const void* a, const void* b);

lines** readfiles(char *startDir, int* count, lines**
array);
```

Makefile:

```
OBJ = readfiles.o main.o

all: $(OBJ)
    gcc $(OBJ) -o main
%.o: %.c func.h
    gcc -c $<
clear:
    rm *.o main
```