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

Студентк гр. 3343 Отмахов Д. В.

Преподаватель Государкин Я.С.

Санкт-Петербург
2024

Цель работы

Изучить принцип работы рекурсивных алгоритмов. На основе полученных знаний, реализовать программу на языке Си, выполняющую обход файловой системы.

Задание

Вариант 3.

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

Пример:

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

Решение:

1 Small text

2 Simple text

3 Wow? Text?

4 Where am I?

5 So much files!

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.

Выполнение работы

Описание функций:

- *int main()* – вызывает рекурсивную функцию *listDir*, записывает в файл результат;
- *FileInfo getFileInfo(char *dir_name, char *file_name)* – считывает информацию из данного файла;
- *void listDir(char *dir_name, Array *arr)* – рекурсивно обходит файловую систему, сохраняя информацию из файлов в массив *arr*;
- *int cmpFileInfo(const void *obj_a, const void *obj_b)* – сравнивает информацию файлов по числу указанному в начале файла;
- *char *pathcat(const char *path1, const char *path2)* – возвращает путь к файлу.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	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!	Выходные данные соответствуют ожидаемым.

Выводы

В ходе выполнения лабораторной работы были изучены основные принципы работы рекурсивных алгоритмов, реализована программа на языке Си, выполняющая обход файловой системы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define STEP 10
#define ROOT_DIR "root"
#define RESULT_FILE "result.txt"


typedef struct fileinfo_t{
    char *text;
    int number;
} FileInfo;

typedef struct array_t{
    size_t cur_index;
    size_t max_count;
    FileInfo *data;
} Array;


char *pathcat(const char *path1, const char *path2){
    int res_path_len = strlen(path1) + strlen(path2) + 2;
    char *res_path = malloc(res_path_len * sizeof(char));
    sprintf(res_path, "%s/%s", path1, path2);

    return res_path;
}


void check_and_resize(Array *arr){
    if(arr->cur_index >= arr->max_count){
        arr->max_count += STEP;
        FileInfo *tmp = realloc(arr->data, arr->max_count *
sizeof(FileInfo));
        arr->data = tmp;
    }
}


int cmpFileInfo(const void *obj_a, const void *obj_b){
    FileInfo *info_a = (FileInfo *)obj_a;
    FileInfo *info_b = (FileInfo *)obj_b;
    if (info_a->number < info_b->number)
        return -1;
    if (info_a->number > info_b->number)
        return 1;
    return info_a->number == info_b->number;
}
```

```
FileInfo getFileInfo(char *dir_name, char *file_name){
    char* file_path = pathcat(dir_name, file_name);

    FileInfo info;
    info.text = malloc(256);

    FILE *file = fopen(file_path, "r");
    char str[256];
    fgets(str, 256, file);

    snprintf(info.text, sizeof(str), "%s", str);

    char *num = strtok (str, " ");
    info.number = atoi(num);

    free(file_path);
    fclose(file);

    return info;
}


void listDir(char *dir_name, Array *arr){
    DIR *root_dir = opendir(dir_name);

    if (!root_dir)
        return;

    struct dirent *dir = readdir(root_dir);
    while (dir){
        char* new_dir = pathcat(dir_name, dir->d_name);

        if(dir->d_type == DT_REG){
            arr->data[arr->cur_index++] = getFileInfo(dir_name, dir-
>d_name);
            check_and_resize(arr);
            free(new_dir);
        }

        else if (dir->d_type == DT_DIR && strcmp(dir->d_name, ".") != 0
&& strcmp(dir->d_name, "..") != 0){
            listDir(new_dir, arr);
            free(new_dir);
        }

        dir = readdir(root_dir);
    }

    closedir(root_dir);
}


int main(){
    Array info_arr;
    info_arr.cur_index = 0;
    info_arr.max_count = STEP;
    info_arr.data = malloc(info_arr.max_count * sizeof(FileInfo));
```

```
    listDir(ROOT_DIR, &info_arr);
    qsort(info_arr.data, info_arr.cur_index, sizeof(FileInfo), cmp-
FileInfo);

    FILE *res_file = fopen(RESULT_FILE, "w");

    for (size_t i = 0; i < info_arr.cur_index; i++){
        fprintf(res_file, "%s\n", info_arr.data[i].text);
        free(info_arr.data[i].text);
    }

    free(info_arr.data);
    fclose(res_file);

    return 0;
}
```