**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 |
|  | 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), cmpFileInfo);

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;

}