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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Программирование»
Тема: Обход файловой системы.

Студентка гр. 0382 _____ Здобнова К.Д.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Изучить способы обхода файловой системы. Научиться работать с данными, полученными при обходе.

Задание.

Вариант 1.

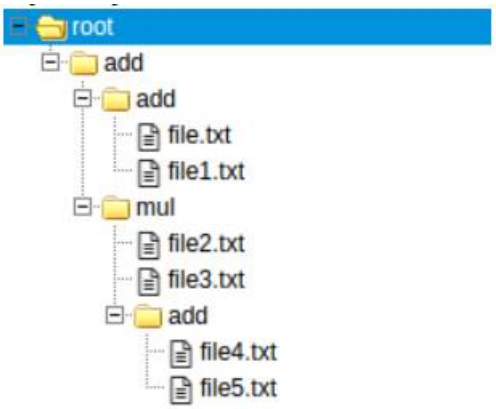
Дана некоторая корневая директория, в которой может находиться некоторое количество папок, в том числе вложенных. В этих папках хранятся некоторые текстовые файлы, имеющие имя вида .txt.

Требуется найти файл, который содержит строку "Minotaur" (файл-минотавр).

Файл, с которого следует начинать поиск, всегда называется file.txt (но полный путь к нему неизвестен).

Каждый текстовый файл, кроме искомого, может содержать в себе ссылку на название другого файла (эта ссылка не содержит пути к файлу). Таких ссылок может быть несколько.

Пример:



Содержимое файла a1.txt

@include a2.txt

@include b5.txt

@include a7.txt

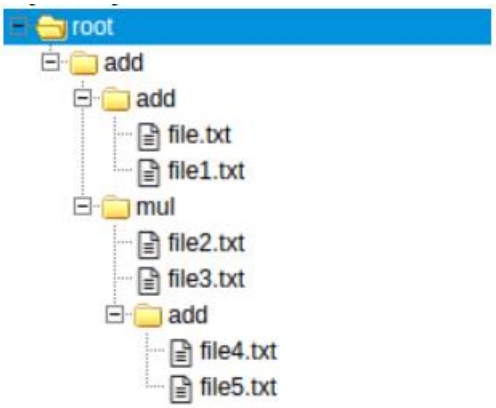
А также файл может содержать тупик:

Содержимое файла a2.txt

Deadlock

Программа должна вывести правильную цепочку файлов (с путями), которая привела к поимке файла-минотавра.

Пример



file.txt:

@include file1.txt

@include file4.txt

@include file5.txt

file1.txt:

Deadlock

file2.txt:

@include file3.txt

file3.txt:

Minotaur

file4.txt:

@include file2.txt

@include file1.txt

file5.txt:

Deadlock

Правильный ответ:

./root/add/add/file.txt

./root/add/mul/add/file4.txt

./root/add/mul/file2.txt

./root/add/mul/file3.txt

Цепочка, приводящая к файлу-минотавру может быть только одна.

Общее количество файлов в каталоге не может быть больше 3000.

Циклических зависимостей быть не может.

Файлы не могут иметь одинаковые имена.

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*. Ваша программа должна обрабатывать директорию, которая называется *labyrinth*.

Выполнение работы.

Поиск исходного файла осуществляется с помощь функции *int find_minotaur(char* name, char** names, char** paths, int cnt, char** ans, int* recount)*, которая возвращает 1 при нахождении файла, 0 – в противном случае.

Для создания массивов данных используется функция *int gointolab(char *lab, char** names, char** ways, int* counter)*.

В основной функции *main()* происходит вывоз рекурсивной функции *write_directory*, создающая два массива – массив имен и всех директорий имеющихся файлов, и считающая их количество в переменную *counter*. Проверка всех файлов происходит в *find_file* (которая возвращает 1 при нахождении файла, 0 – в противном случае). Если файл содержит в себе дальнейший путь по файлам, запускается рекурсия, при завершении функции путь записывается в *result.txt*. Освобождаем память с помощью функции *free()*.

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.	file.txt: @include file1.txt @include file4.txt @include file5.txt file1.txt: Deadlock file2.txt: @include file3.txt file3.txt: Minotaur file4.txt: @include file2.txt @include file1.txt file5.txt: Deadlock	./root/add/add/file.txt ./root/add/mul/add/file4.txt ./root/add/mul/file2.txt ./root/add/mul/file3.txt	программа работает корректно

Выводы.

Были освоены основные принципы работы с файловой системой на языке программирования Си. Разработана программа, совершающая обход файловой системы и поиск файла с определенным содержимым.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: *main.c*

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

int find_minotaur(char* name, char** names, char** paths, int cnt, char**
ans, int* recount)
{
    int i;
    for(i = 0; i < cnt; i++)
    {
        if (strcmp(names[i], name) == 0)
            break;
    }
    char* way = malloc(260 * sizeof(char));
    way[0]='\0';
    strcat(way, paths[i]);
    strcat(way, "/");
    strcat(way, names[i]);

    FILE * open_file = fopen(way, "r");
    char* str = malloc(260 * sizeof(char));
    str[0]='\0';
    char* obj = malloc(260 * sizeof(char));
    obj[0]='\0';
    while(fscanf(open_file, "%s", str) != EOF) {
        if(strcmp(str, "@include")==0) {
            fscanf(open_file, "%s\n", obj);
            if (find_minotaur(obj, names, paths, cnt, ans, recount)) {
                ans[*recount] = malloc(256*sizeof(char));
                (ans[*recount])[0] = '\0';
                strcat(ans[*recount], way);
                (*recount)++;
                fclose (open_file);
                return 1;
            }
        }
        else if(strcmp(str, "Deadlock")==0) {
            fclose (open_file);
```

```
            return 0;
        }
        else if(strcmp(str, "Minotaur")==0) {
            ans[*recount] = malloc(260*sizeof(char));
            (ans[*recount])[0] = '\0';
            strcat(ans[*recount], way);
            (*recount)++;
            fclose (open_file);
            return 1;
        }
    }
    fclose (open_file);
    return 0;
}

int gointolab( char *lab, char** names, char** ways, int* counter)
{
    DIR *dir = NULL;
    struct dirent *de = NULL;
    char pathname[PATH_MAX + 1];
    dir = opendir(lab);
    if(dir == NULL)
        return -1;
    de = readdir(dir);
    while(de != NULL){
        struct stat entryInfo;
        if((strncmp( de->d_name, ".", PATH_MAX) == 0) || (strncmp( de-
>d_name, "..", PATH_MAX) == 0)){
            de = readdir(dir);
            continue;
        }
        (void)strncpy(pathname, lab, PATH_MAX);
        (void)strncat(pathname, "/", PATH_MAX);
        (void)strncat(pathname, de->d_name, PATH_MAX);
        if(lstat(pathname, &entryInfo)== 0){
            if(S_ISDIR(entryInfo.st_mode))
                gointolab(pathname, names, ways, counter);
            else if(S_ISREG(entryInfo.st_mode)) {
                names[*counter] = malloc(260*sizeof(char));
                ways[*counter] = malloc(260*sizeof(char));
                (names[*counter])[0]='\0';
                (ways[*counter])[0]='\0';

                strcat(names[*counter], de->d_name);
                strcat(ways[*counter], lab);
```

```
                (*counter)++;
            }
        }
        de = readdir(dir);
    }
    (void)closedir(dir);
    return 0;
}

int main()
{   int  cnt = 0;
    char** names = malloc(4000 * sizeof(char*));
    char** paths = malloc(4000 * sizeof(char*));
    char** ans = malloc(4000 * sizeof(char*));
    int recount = 0;
    if (gointolab( "./labyrinth", names, paths, &cnt) == 0)
    {
        find_minotaur("file.txt", names, paths, cnt, ans, &recount);
        FILE * open_file = fopen("result.txt", "w");
        while(recount!=0)
        {
            fputs(ans[recount-1], open_file);
            fputs("\n", open_file);

            printf("%s\n",ans[recount - 1]);
            recount--;
        }
        fclose (open_file);
        for(int i = 0; i < cnt; i++){
            free(names[i]);
            free(paths[i]);
        }
        for(int i = 0; i < recount; i++)
            free(ans[i]);
        free(names);
        free(paths);
        free(ans);
        return 0;
    }
}
```