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.

Студентка гр. 0382 _____ Кривенцова Л.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Научиться работать с файловой системой с помощью языка программирования Си.

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 <filename>.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.

Основные теоретические положения.

Рассмотрим основные функции для работы с деревом файловой системы, объявления которых находятся в заголовочном файле dirent.h (также, может понадобиться включить заголовочный файл sys/types.h)

Для того, чтобы получить доступ к содержимому некоторой директории можно использовать функцию

*DIR *opendir(const char *dirname);*

Которая возвращает указатель на объект типа DIR с помощью которого можно из программы работать с заданной директорией.

Тип DIR представляет собой поток содержимого директории. Для того, что бы получить очередной элемент этого потока, используется функция

*struct dirent *readdir(DIR *dirp);*

Она возвращает указатель на объект структуры dirent, в котором хранится информация о файле. Основной интерес представляют поля, хранящие имя и тип объекта в директории (это может быть не только "файл" и "папка").

После завершения работы с содержимым директории, необходимо вызвать функцию

*int closedir(DIR *dirp);*

Передав ей полученный функцией readdir() ранее дескриптор.

```
void list_dir(const char *dirPath)
{
   DIR *dir = opendir(dirPath);                    // "открываем" директорию
   if(dir) {                                       // если это удалось успешно
      struct  dirent *de = readdir(dir);  // получаем очередной элемент открытой директории
      while (de) {                                 // если это удалось
         printf("%s/%s\n", dirPath, de->d_name); // печатаем имя этого элемента
         de = readdir(dir);                       // снова получаем очередной элемент открытой директории
      }
   }
   closedir(dir);                                  // не забываем "закрыть" директорию
```

}

Выполнение работы.

Ход решения:

Используется стандартная библиотека языка си и её заголовочные файлы *stdlib.h, string.h*(для работы со строками) и dirent.h, sys/stat.h(для работы с файловой системой).

В главной функции main вызывается рекурсивная функция write_directory, создающая два массива – массив имён (names) и всех директорий имеющихся файлов(ways), помимо того считая их количество в переменную counter (рекурсия закручивается, когда находится не файл, а директория). Затем, обращаясь к функции find_file, проверяем все файлы (при встрече тупика функция возвращает 0, при встрече искомого файла возвращает 1). Если же файл содержит в себе дальнейший путь по файлам, запускается рекурсия. Функция завершается, а путь записывается в файл с результатом.

1. int find_minotaur(char* name, char names, char** ways, int counter, char** result, int* recount).**

Функция проверяет все файлы: при встрече тупика функция возвращает 0, при встрече искомого файла возвращает 1. Если же файл содержит в себе дальнейший путь по файлам, запускается рекурсия.

2. int gointolab(char *lab, char names, char** ways, int* counter)**

Функция для создания массивов – массив имён (names) и всех директорий имеющихся файлов(ways), помимо того для подсчёта их количества в переменную counter (рекурсия закручивается, когда находится не файл, а директория).

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:	./root/add/add/file.txt	Программа выводит верный ответ.

	@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/mul/add/ file4.txt ./root/add/mul/ file2.txt ./root/add/mul/ file3.txt	

Выводы.

Были освоены навыки 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: lb3.c

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

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

    FILE * xfile = fopen(needway, "r");
    char* str = malloc(260*sizeof(char));
    str[0]='\0';
    char* xname = malloc(260*sizeof(char));
    xname[0]='\0';

    while(fscanf(xfile, "%s", str) != EOF)
    {
        if(strcmp(str, "@include")==0)
        {
            fscanf(xfile, "%s\n", xname);
            if (find_minotaur(xname, names, ways, counter, result,
recount))
            {
                result[*recount] = malloc(256*sizeof(char));
                (result[*recount])[0] = '\0';
                strcat(result[*recount], needway);
                (*recount)++;
                fclose (xfile);
                return 1;
            }
        }
        else if(strcmp(str, "Deadlock")==0)
        {
            fclose (xfile);
            return 0;
        }
```

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

int gointolab( char *lab, char** names, char** ways, int* counter)
{
    DIR *dir = NULL;
    struct dirent *de = NULL;
    char wayname[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(wayname, lab, PATH_MAX);
        (void)strncat(wayname, "/", PATH_MAX);
        (void)strncat(wayname, de->d_name, PATH_MAX);
        if(lstat(wayname, &entryInfo)== 0)
        {  if(S_ISDIR(entryInfo.st_mode))
            {
                gointolab(wayname, 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 counter=0;
    char** names = malloc(4000*sizeof(char*));
    char** ways = malloc(4000*sizeof(char*));
    char** result = malloc(4000*sizeof(char*));
    int recount = 0;
    if (gointolab( "./labyrinth", names, ways, &counter) == 0)
    {
        find_minotaur("file.txt", names, ways, counter, result,
&recount);
        FILE * xfile = fopen("result.txt", "w");
        while(recount!=0)
        {
            fputs(result[recount-1], xfile);
            fputs("\n", xfile);

            printf("%s\n",result[recount-1]);
            recount--;
        }
        fclose (xfile);
        for(int i=0;i<counter;i++){
            free(names[i]);
            free(ways[i]);
        }
        for(int i=0;i<recount;i++){
            free(result[i]);
        }
        free(names);
        free(ways);
        free(result);
        return 0;
    }
}
```