**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе 4**

**по дисциплине «Программирование»**

Тема: «**Строки. Рекурсия, циклы, обход дерева**»

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Попов С.С. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2017

# Цель работы

Научиться работать с директориями и поддиректориями. Искать файлы и выполнять различные действия в зависимости от их содержимого.

# Задание

**Лабиринт**

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

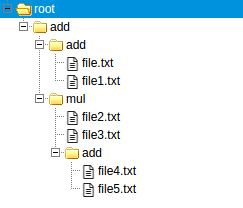
А также файл может содержать тупик:

**Содержимое файла a2.txt**

Deadlock

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

**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 200.*

*! Циклических зависимостей быть не может.*

*! Файлы не могут иметь одинаковые имена.*

# Содержание

**Lab3.c**

#include <stdio.h>

#include <dirent.h>

#include <string.h>

#include <sys/stat.h>

#include <stdlib.h>

// ---------------------------------------------

// List of strings to store the path to Minotaur

// ---------------------------------------------

// Represents one item of the list

typedef struct string

{

char value[1024]; // Lenght of one path

} string;

// Represents list of strings

typedef struct list

{

string item[200]; // Maximum amount of items in the list according to the requirments of the task

int count;

} list;

// Add new item to the end of the list

add(list\* mylist, char\* string)

{

strcpy(mylist->item[mylist->count].value, string);

mylist->count++;

}

// Remove last item from the list

remove\_last(list\* mylist)

{

mylist->count--;

}

// ---------------------------------------------

// Looking for file with given filename in specific

// directory including subdirectories and return path to the file

// Return value: 1 if found, 0 otherwise

int find\_file(char\* dir, char\* file, char\* path)

{

int result = 0;

DIR \*dp;

struct dirent \*entry;

struct stat statbuf;

// Open given directory

if((dp = opendir(dir)) == NULL)

{

printf("Cannot open directory: %s\n", dir);

return result;

}

// Change current working directory to the given one

chdir(dir);

// Read items from the directory

while((entry = readdir(dp)) != NULL)

{

// Get additional information about current item

lstat(entry->d\_name, &statbuf);

// Check if the found item is a directory

if( S\_ISDIR(statbuf.st\_mode) )

{

// Found a directory, but ignore current(.) and parent(..) directories

if(strcmp(".",entry->d\_name) == 0 || strcmp("..",entry->d\_name) == 0)

{

continue;

}

// Recurse at the level of found directory

result = find\_file(entry->d\_name, file, path);

if (result == 1)

{

// If found, run away

break;

}

}

else // Assume if it's not directory, it is a file

{

// Check if the found file is the required one

if (strcmp(entry->d\_name, file) == 0)

{

// Build the path to the file

char cwd[1024];

getcwd(cwd, sizeof(cwd)); // Get current working directory

strcat(path, cwd);

strcat(path, "/"); // Add '/'

strcat(path, entry->d\_name); // Add filename

result = 1;

// If found, run away

break;

}

}

}

chdir(".."); // Change current working directory back to parent one

closedir(dp); // Close directory

return result;

}

// Looking for the Minotaur and return steps to him

// Return value: 1 if found, 0 otherwise

int find\_minotaur(char\* dir, char\* file, list\* steps)

{

int result = 0;

// Try to find file in the given directory and subdirectories

char path[1024] = "";

if (find\_file(dir, file, path) == 0)

{

// If not found, report the error and stop the program

printf("Cannot find file: %s\n", file);

return result;

}

// If found, add current file to the list of steps

add(steps, path);

// Open found file for reading

FILE\* fp = NULL;

fp = fopen(path,"r");

// Read from file line by line

char line[256];

while (fgets(line, sizeof(line), fp))

{

// We found the Minotaur!

if (strcmp(line, "Minotaur\n") == 0) // fgets reads lines including trailing '\n'

{

result = 1;

break;

}

// Deadlock found

if (strcmp(line, "Deadlock\n") == 0)

{

// Since Deadlock is found, remove last steps from the list

remove\_last(steps);

break;

}

// If not 'Minotaur' nor 'Deadlock', assume it is 'include', try to parse

char next\_file[256] = "";

int n = sscanf(line,"@include %s", next\_file);

if (n == 0)

{

printf("Cannot read file: %s\n", path);

result = 0;

return result;

}

// Managed to parse included file, try to find Minotaur there...

result = find\_minotaur(dir, next\_file, steps);

if (result == 1)

{

// If found, run away

break;

}

}

// Close the file

fclose(fp);

return result;

}

int main(int argc, char\*\* argv)

{

// Use input argument as starting directory if provided, otherwise use 'root'

char dir[256] = "root";

if (argc >= 2)

{

strcpy(dir, argv[1]);

}

// Allocate list of steps to store path to Minotaur

list steps;

steps.count = 0;

// Find path to Minotaur

int result = find\_minotaur(dir, "file.txt", &steps);

if (result == 0)

{

printf("Cannot find the Minotaur\n");

return 0;

}

// If path is found, print it out

int i = 0;

for (i=0;i<steps.count;i++)

{

printf("%s\n",steps.item[i].value);

}

return 0;

}

# Вывод

В ходе данной лабораторной работы были получены навыки перемещения по дереву папок. Закреплено на практике умение читать текст из файлов и выполнять различные действия в зависимости от содержимого