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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Программирование»

ТЕМА: Строки. Рекурсия, циклы, обход дерева.

Студент гр. 6304 _____ Иванов В.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2017

Оглавление

Цель работы

Создание функции для обхода дерева из папок, реализованной с помощью рекурсии, а также освоение работы с файлами.

Задание

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.

Содержание (реализация функций)

1. Функция *text_from_file*

Это функция служит для работы с файлом. Она определяет его размер, а также читает его и записывает его содержимое в строку.

```
/*Функия для считывания информации из файла и определяющая его размер*/
char *text_from_file(const char *current_path, int *file_size){
    FILE *file=fopen(current_path,"rt");
    /*Определение размера файла*/
    fseek(file,0, SEEK_END);
    *file_size = ftell(file);
    fseek(file,0, SEEK_SET);

    char *str=(char*)malloc(sizeof(char)*(*file_size));
    int i=0;
    /*Посимвольное копирование из файла в строку*/
    char symb;
    while(fscanf(file,"%c",&symb)>0) str[i++]=symb;
    fclose(file);
    return str;
}
```

2. Функция *list_dir*

Функция используется для рекурсивного обхода директории в поиске файла, содержащего слово Minotaur, и возвращает длину пути (количество файлов, содержащих ссылку на искомый файл).

```
int list_dir(const char *newcur_path,const char *startdir, char* name_of_file, char **name_path)
{
    char current_path[10000];//Ввод для того, чтобы не изменять исходные данные
    int height_of_path=0; //Длина пути

    strcpy(current_path,newcur_path); //Запись пути для дальнейшего использования и изменения
    DIR *dir = opendir(current_path);//Открытие директории для получения информации
    struct  dirent *de = readdir(dir); //Рассматрение следующего объекта

    if(dir){
        while(de)//Если есть, что рассматривать
        {
            int path_len = strlen(current_path);
                /*Изменение пути на текущий путь*/
                strcat(current_path,"/");
                strcat(current_path,de->d_name);
             /*Поиск нужного файла*/
            if(de->d_type==DT_REG && !strcmp(de->d_name,name_of_file)){
                int i,j=0,file_size=0;
                char *str=text_from_file(current_path,&file_size);
                char token[199][file_size];//Массив строк для лексем
                /*Разбиение на лексемы*/
                char *tmp=strtok(str," \n");
                while(tmp){
                    strcpy(token[j],tmp);
                    j++;
                    tmp=strtok(NULL," \n");
                }
                /*Разбор и анализ лексем*/
                for(i=0;i<j;i++){
                    if(!strcmp(token[i],"@include"));
                    else if(!strcmp(token[i],"Deadlock")) return height_of_path;
                     /*Если файл Minotaur найден, запоминаем к нему путь и увеличиваем счётчик кол-ва путей
                        Иначе, продолжаем поиск*/
                    else if(!strcmp(token[i],"Minotaur")){
                        name_path[height_of_path]=(char*)malloc(sizeof(char)*1000);
                        strcpy(name_path[height_of_path],current_path);
                        height_of_path=1;
                        return height_of_path;
                    }
                    else{
                          height_of_path=list_dir(startdir,startdir,token[i],name_path);
                          if(height_of_path!=0){
                              name_path[height_of_path]=(char*)malloc(sizeof(char)*1000);
                              strcpy(name_path[height_of_path],current_path);
                              height_of_path++;
                              return height_of_path;
                          }
                      }
                  }
              }
              /*Если есть внутренние директории, кроме текущей и родительской, проверяем и их на наличие файла*/
              if( de->d_type == DT_DIR && 0!=strcmp(".",de->d_name) && 0!=strcmp("..",de->d_name))
              {
                  height_of_path=list_dir(current_path,startdir,name_of_file,name_path);
              }
              /*Предварительное завершение просмотра директории, если файл найден*/
              if (height_of_path!=0){
                  return height_of_path;
              }
              current_path[path_len] = '\0';
              de = readdir(dir);//Рассматриваем следующий объект в текущей директории
          }
           closedir(dir);
      }
      return height_of_path;
  }
```

3. Функция *main*

```c
int main()
{
    char **name_path=(char**)malloc(200*sizeof(char*));//Массив путей
    int height_of_path=0,i;
    char *name_of_file="file.txt";
    height_of_path=list_dir(".",".",name_of_file,name_path);
    for(i=height_of_path-1;i>=0;i--){//Вывод путей в обратном порядке
        printf("%s\n",name_path[i]);
        free(name_path[i]);}
    free(name_path);
    return 0;
}
```

Вывод

В результате выполнений данной лабораторной работы, была написана программа для рекурсивного обхода дерева из папок, которая использовалась для чтения и поиска файлов, находящихся в подпапках, что помогло изучить работу с файловой системой.

Приложение

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


/*Функия для считывания информации из файла и определяющая его размер*/
char *text_from_file(const char *current_path, int *file_size){
    FILE *file=fopen(current_path,"rt");
    /*Определение размера файла*/
    fseek(file,0, SEEK_END);
    *file_size = ftell(file);
    fseek(file,0, SEEK_SET);

    char *str=(char*)malloc(sizeof(char)*(*file_size));
    int i=0;
    /*Посимвольное копирование из файла в строку*/
    char symb;
    while(fscanf(file,"%c",&symb)>0) str[i++]=symb;
    fclose(file);
    return str;
}
/*Функция обходит всю директорию в поиске файла, содержащего слово Minotaur, и
возвращает длину пути*/
int list_dir(const char *newcur_path,const char *startdir, char* name_of_file, char
**name_path)
{
    char current_path[10000];//Ввод для того, чтобы не изменять исходные данные
    int height_of_path=0; //Длина пути

    strcpy(current_path,newcur_path); //Запись пути для дальнейшего использования и
изменения
    DIR *dir = opendir(current_path);//Открытие директории для получения информации
    struct  dirent *de = readdir(dir); //Рассматрение следующего объекта


    if(dir){
        while(de)//Если есть, что рассматривать
        {
            int path_len = strlen(current_path);
                /*Изменение пути на текущий путь*/
                strcat(current_path,"/");
                strcat(current_path,de->d_name);
             /*Поиск нужного файла*/
            if(de->d_type==DT_REG && !strcmp(de->d_name,name_of_file)){
                int i,j=0,file_size=0;
                char *str=text_from_file(current_path,&file_size);
                char token[199][file_size];//Массив строк для лексем
                /*Разбиение на лексемы*/
                char *tmp=strtok(str," \n");
                while(tmp){
                    strcpy(token[j],tmp);
                    j++;
                    tmp=strtok(NULL," \n");
                }
                /*Разбор и анализ лексем*/
                for(i=0;i<j;i++){
                    if(!strcmp(token[i],"@include"));
                    else if(!strcmp(token[i],"Deadlock")) return height_of_path;
                     /*Если файл Minotaur найден, запоминаем к нему путь и увеличиваем
счётчик кол-ва путей
                        Иначе, продолжаем поиск*/
                    else if(!strcmp(token[i],"Minotaur")){
                        name_path[height_of_path]=(char*)malloc(sizeof(char)*1000);
```

```
                        strcpy(name_path[height_of_path],current_path);
                        height_of_path=1;
                        return height_of_path;
                    }
                    else{
                        height_of_path=list_dir(startdir,startdir,token[i],name_path);
                        if(height_of_path!=0){

name_path[height_of_path]=(char*)malloc(sizeof(char)*1000);
                            strcpy(name_path[height_of_path],current_path);
                            height_of_path++;
                            return height_of_path;
                        }
                    }
                }
            }
            /*Если есть внутренние директории, кроме текущей и родительской, проверяем
и их на наличие файла*/
            if( de->d_type == DT_DIR && 0!=strcmp(".",de->d_name) &&
0!=strcmp("..",de->d_name))
            {
                height_of_path=list_dir(current_path,startdir,name_of_file,name_path);
            }
            /*Предварительное завершение просмотра директории, если файл найден*/
            if (height_of_path!=0){
                return height_of_path;
            }
            current_path[path_len] = '\0';
            de = readdir(dir);//Рассматриваем следующий объект в текущей директории
        }
         closedir(dir);
    }
    return height_of_path;
}


int main()
{
    char **name_path=(char**)malloc(200*sizeof(char*));//Массив путей
    int height_of_path=0,i;
    char *name_of_file="file.txt";
    height_of_path=list_dir(".",".",name_of_file,name_path);
    for(i=height_of_path-1;i>=0;i--){//Вывод путей в обратном порядке
        printf("%s\n",name_path[i]);
        free(name_path[i]);}
    free(name_path);
    return 0;
}
```