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

Студент гр. 8383 _____ Дейнега В.Е.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург
2019

Цель работы.

Научиться делать обход файловой системы с помощью рекурсивного алгоритма. Рассмотреть основные функции заголовочного файла dirent.h.

Основные теоретические положения.

Самое простое определение рекурсии можно дать как вызов функции из неё же самой, непосредственно или через другие функции. (например, функция foo() вызывает функцию bar(), а функция bar() — функцию foo(). Количество вложенных вызовов функции называется глубиной рекурсии. Важно понимать, что на каждый вызов функции приходятся как временные затраты, так и затраты памяти в области стека, основная часть которых приходится на хранение аргументов функции и ее локальных переменных.

Очень хорошо использование рекурсии ложится на задачи, которые можно разделять на себе подобные. Иногда это не так просто увидеть (например, утверждение, что "сумма элементов массива - это сумма первого элемента плюс сумма остальных элементов" может прийти в голову не сразу).

При разработке рекурсивного алгоритма, в первую очередь следует продумать условие завершения рекурсии. Часто, это выглядит как обработка частных конечных случаев. Продолжая пример из предыдущего абзаца: сумма элементов массива, состоящего из одного элемента - сам этот элемент.

Рекурсия хороша только там, где она действительно уместна, но что бы научиться ее использовать, начинать стоит с простых задач (хоть и многие из которых имеют более эффективное итеративное решение). Следующие задачи имеют рекурсивное решение, которое Вам предстоит реализовать.

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.

Постановка задачи:

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

Реализация:

В функции main вызывается рекурсивная функция list_dir, которая перебирает все файлы и папки из корневой директории “root”. Встречая файл, функция записывает его содержимое в динамический массив строк, выделяя, когда нужно, память. Если функция встречает директорию, она вызывает саму себя для новой директории. Затем вызывается функция qsort, которая сортирует массив строк по возрастанию первых чисел в строке. Функция cmp выделяет из строк первые числа, с помощью функции findNumbers, использующей функцию стандартной библиотеки sscanf, и сравнивает эти 2 числа, возвращая соответствующее сравнению число. После, с помощью цикла for происходит запись отсортированного массива строк в файл.

Вывод.

В ходе лабораторной работы был изучен рекурсивный обход файловой системы, рассмотрены функции для работы с файлами в языке си.

ПРИЛОЖЕНИЕ

КОД ПРОГРАММЫ:

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include<string.h>
#include <ctype.h>

#define vtos(v) ( *((char**)v) )
#define STRLEN 100
#define STRBLOCK 10



long long int findNumbers(char* strs){
    long long int k = 0;
    int tmp = sscanf(strs, "%lld", &k);
    return k;
}

int cmp(const void* a, const void* b){
    long long int c;c = findNumbers(vtos(a));
    long long int h;h = findNumbers(vtos(b));
    return c>h?1:c==h?0:-1;
}


void list_dir(const char *dirPath, char*** strs, int* i){
    DIR *dir = opendir(dirPath);
    char* c = 0;
    char* str = NULL;
    char buf[STRLEN];
    if(dir) {
        struct  dirent *de = readdir(dir);
        while (de) {
            str = malloc(strlen(dirPath) + strlen(de->d_name) +
2);
            strcpy(str,dirPath);
            strcat(str, "/");
            strcat(str, de->d_name);
            if (!strchr(de->d_name, '.'))
                    list_dir(str, strs,i);

            FILE* tempFile = fopen(str, "r+");
            if (tempFile){
                while(c = fgets(buf,STRLEN,tempFile)){
                    char* tmp = strchr(buf,'\n');
                    if(tmp)*tmp = '\0';
                    if(!((*i)%STRBLOCK))(*strs) =
realloc(*strs,((*i)+STRBLOCK)*sizeof(char*));
                    (*strs)[(*i)]=malloc(STRLEN * sizeof(char));
```

```
                    strcpy((*strs)[(*i)],buf);
                    (*i)++;

                }
                fclose(tempFile);
            }

            de = readdir(dir);
            free(str);
        }
    }
    closedir(dir);
}


int main(){
    int i = 0;
    char path[] = "./root";
    char** strs = NULL;
    FILE* file = fopen("vivod.txt","w");
    list_dir(path, &strs, &i);
    qsort(strs, i,sizeof(char*),cmp);
    for(int k=0; k<i; k++){
        fputs(strs[k],file);
        fputs("\n",file);
    }
    return 0;
}
```