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. 1304 _____ Крупин Н. С.

Преподаватель _____ Чайка К. В.

Санкт-Петербург
2022

Цель работы

Освоение работы с рекурсией, реализация обхода дерева с её помощью, знакомство с библиотекой dirent.h.

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 .txt. В качестве имени файла используется символ латинского алфавита.

На вход программе подается строка. Требуется найти и вывести последовательность полных путей файлов, имена которых образуют эту строку.

! Регистрозависимость.

! Могут встречаться файлы, в имени которых есть несколько букв и эти файлы использовать нельзя.

! Одна буква может встречаться один раз.

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 tmp».

Выполнение работы

Для решения задачи рекурсивно создана функция *find_letter()*, просматривающая все файлы в конкретной директории на соответствие названия заданной букве и печатающая путь к нужному в заданный файл. Если очередной файл является директорией, функция вызывает сама себя, но для новой директории.

Функция имеет параметры *dirPath* – указатель на строку с адресом директории, *letter* – искомая буква и *file* – указатель на файл, в который требуется записать результат. Так как пути для каждого рекурсивного вызова будут отличаться, было принято решение поберечь стековую память и хранить их динамически.

Ввиду того что проверить, является ли файл директорией, зная лишь его название, невозможно, проверка происходит в самом начале функции – с помощью встроенной функции *opendir()* пробуем открыть переданный адрес как директорию, и если не получилось, возвращаем ненулевое значение и поднимаемся на уровень выше. Если получилось, сразу выделяется динамическая память для хранения нового адреса, куда сначала дублируется прежний адрес, оканчивающийся символом `'/'`. Память выделяется с учётом длины имеющегося адреса и длины максимально возможного в Linux названия файла (255 байт), а если выделения не происходит, молча пропускает данную директорию. Таким образом программа имеет ограничение по вложенности директорий и длине названий, но ввиду использования динамической памяти это ограничение менее жёсткое.

Далее с помощью встроенной функции *readdir()* в цикле для каждого файла директории дописывается свой адрес и вызывается *find_letter()*. Скрытые файлы, начинающиеся на точку, сразу исключаются из рассмотрения ввиду опасности зацикливания. Если файл не был директорией и функция вернула ненулевое значение, название файла сравнивается с эталонным, начинающимся на *letter*. В случае соответствия его адрес записывается в файл на отдельной строке и работа функции корректно завершается. Нахождение директории с названием, соответствующим шаблону, не считается удовлетворительным результатом.

Если ни один файл данной директории и вложенных в неё не соответствует шаблону, работа функции также завершается – очищается

динамическая память и директория закрывается от рассмотрения с помощью встроенной функции *closedir()*.

В функции *main()* происходит открытие файла result.txt на запись и посимвольное считывание с консоли в цикле, пока соблюдается принадлежность символов к латинскому алфавиту. Для каждой буквы вызывается функция *find_letter()*. После цикла файл закрывается и работа программы завершается. Также программа завершается при возникновении ошибки при создании файла, в этом случае считывания входных данных не происходит.

Разработанный программный код см. в приложении А.

Тестирование

Схема файловой системы:

tmp (директория)
- **A.txt** (директория)
  - **a.txt** (директория)
    - A.txt
  - B.txt
  - C.txt
- **DirNextDoor** (директория)
  - **D.txt** (директория)
  - a.txt
  - b.txt
  - cc.txt
- **.Slave** (директория)
  - d.txt
- a.txttxt
- c.txt
- 2.txt

solution.c

Данные, введённые в консоль:

AaBbCcDdAa2Bb

Содержимое файла result.txt после выполнения программы:

./tmp/A.txt/a.txt/A.txt
./tmp/DirNextDoor/a.txt
./tmp/A.txt/B.txt
./tmp/DirNextDoor/b.txt
./tmp/A.txt/C.txt
./tmp/c.txt
./tmp/A.txt/a.txt/A.txt
./tmp/DirNextDoor/a.txt

Выводы

Изучены особенности применения рекурсивных алгоритмов на языке Си, а также возможности языка для работы с содержимым директорий, подключаемые с помощью заголовочного файла dirent.h стандартной библиотеки.

Разработана программа, реализующая обход файлового дерева и выполняющая поиск пути к файлу по шаблону его названия. Проведена примитивная работа над исключением возможности аварийного завершения программы. Программа использует рекурсивный, а не циклический метод, но с уменьшенной нагрузкой на стек за счёт использования динамической памяти.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: solution.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <dirent.h>
#define MAX_NAME 255 // max size of file name in Linux (bytes)

int find_letter(const char* dirPath, const char letter, FILE* file){
	DIR* dir; struct dirent* de; int len;
	char *newDirPath, wanted_name[] = " .txt";
	if (!(dir = opendir(dirPath))) return 1;
	// open directory or say "it's just a file"

	wanted_name[0] = letter;
	len = strlen(dirPath);
	newDirPath = (char*)malloc((len+2)*sizeof(char) + MAX_NAME);
	if (!newDirPath){
		closedir(dir);
		return 0; // error: memory can't be allocated
	}
	strcpy(newDirPath, dirPath);
	*(newDirPath + len++) = '/';
	*(newDirPath + len + MAX_NAME) = '\0';

	while (de = readdir(dir)){
		if (de->d_name[0] == '.') continue;
		strncpy(newDirPath+len, de->d_name, MAX_NAME);
		if (find_letter(newDirPath, letter, file))
			if (!strcmp(de->d_name, wanted_name)){
				fputs(newDirPath, file);
				fputs("\n", file);
				break;
			}
	}

	free(newDirPath);
	closedir(dir);
	return 0;
}

int main(){
	FILE* file = fopen("result.txt", "w");
	if (!file) return 0; // error: file can't be created
	char ch = fgetc(stdin);
	while (isalpha(ch)){
		find_letter("./tmp", ch, file);
		ch = fgetc(stdin);
	}
	fclose(file);
	return 0;
}
```