

Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»	
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»	

Лабораторная работа №2

Дисциплина _		Операционные системы	
Тема	Файль	и каталоги	
Студент		Набиев Ф.М.	
Группа		ИУ7-63Б	
Оценка (баллы)		
Преподав	атель	Рязанова Н.Ю.	

1 Задание

1.1 Условие

Структурировать исходный код программы в листинге 4.7 из исходного условия к данной лабораторной работе. Изменить программу так, чтобы она выводила на экран дерево каталогов.

Изменить функцию myftw так, чтобы каждый раз, когда встречается каталог, функции lstat() передавался не полный путь к файлу, а только его имя. Для этого после обработки всех файлов в каталоге вызовите chdir("..").

1.2 Реализация

Листинг 1.1 - Функция таіп

```
#include <stdio.h>
2
   #include <string.h>
3
   #include "myftw.h"
4
5
6
   int main(int argc, char **argv)
7
8
       int ret;
9
10
       if (argc != 2)
11
12
            fprintf(stderr, "Usage:\n %s <input directory>\n", argv[0]);
13
           return 1;
14
15
16
       ret = myftw(argv[1], myfunc);
17
       ntot = nreg + ndir + nblk + nchr + nfifo + nslink + nsock;
18
19
       if (ntot == 0)
20
           ntot = 1;
21
22
       char hrule[256];
23
       memset(hrule, '-', sizeof(hrule));
24
25
       printf("%.*s\n", 35, hrule);
26
       printf("Regular files: %7ld, %5.2f%%\n", nreg,
                                                                 * 100.0/ntot);
                                                          nreg
```

```
27
       printf("Directories:
                           %7ld, %5.2f%%\n", ndir, ndir
                                                              * 100.0/ntot);
28
       printf("Block devices: %7ld, %5.2f%%\n", nblk, nblk * 100.0/ntot);
29
       printf("Char devices: %7ld, %5.2f%%\n", nchr, nchr * 100.0/ntot);
30
       printf("FIFOs:
                             %71d, %5.2f%%\n", nfifo, nfifo * 100.0/ntot);
31
       printf("Symlinks:
                            %7ld, %5.2f%%\n", nslink, nslink * 100.0/ntot);
32
       printf("Sockets:
                             %7ld, %5.2f%%\n", nsock, nsock * 100.0/ntot);
33
       printf("Total:
                             %71d\n",
                                               ntot);
34
35
       return ret;
36
```

Листинг 1.2 – Объявление функции myftw, определение функции вывода, константы

```
#ifndef OSLAB02_MYFTW_H_
2 #define OSLAB02_MYFTW_H_
3
  #include <stdio.h>
5 #include <string.h>
  #include <sys/stat.h>
6
  #define FTW_F
8
                   1 // файл, не являющийся каталогом
9
   #define FTW D
                    2 // каталог
   #define FTW_DNR 3 // каталог, который не доступен для чтения
11
  | #define FTW_NS 4 // файл, о котором нельзя получить информацию
12
13
   extern long nreg, ndir, nblk, nchr, nfifo, nslink, nsock, ntot;
14
   typedef int MyFunc(const char *, const struct stat *, int, int);
15
16
  static int myfunc(const char *pathname,
17
18
                      const struct stat *statptr,
19
                      int type, int len)
20
   {
21
       for (int i = 0; i < len; i++)</pre>
22
               printf(" | ");
23
       if (len >= 0) printf(" |--- ");
24
25
       switch (type)
26
27
           case FTW F:
28
               printf("%s \n", pathname);
29
30
                switch (statptr->st_mode & S_IFMT)
31
32
                    case S_IFREG:
33
                        ++nreg;
34
                        break;
```

```
35
36
                     case S_IFBLK:
37
                         ++nblk;
38
                         break;
39
40
                     case S_IFCHR:
41
                         ++nchr;
42
                         break;
43
44
                     case S_IFIFO:
45
                         ++nfifo;
46
                         break;
47
48
                     case S_IFLNK:
49
                         ++nslink;
50
                         break;
51
52
                     case S_IFSOCK:
53
                         ++nsock;
54
                         break;
55
56
                     case S_IFDIR:
57
                         perror("The directory is of type FTW_F");
58
                         return -1;
59
                }
60
61
                break;
62
63
            case FTW_D:
64
                if (pathname[strlen(pathname) - 1] == '/')
                     printf("%s\n", pathname);
65
66
67
                     printf("%s/\n", pathname);
68
                ++ndir;
69
                break;
70
71
            case FTW_DNR:
72
                perror("Blocked access to one of the directories");
73
                return -1;
74
75
            case FTW_NS:
76
                perror("Function error stat");
77
                return -1;
78
79
            default:
80
                perror("Unknown file type");
81
                return -1;
82
        }
```

```
83 | return 0;

85 | }

86 | int myftw(char *pathname, MyFunc *func);

88 | #endif // OSLABO2_MYFTW_H_
```

Листинг 1.3 – Рекурсивная реализация myftw

```
#include "myftw.h"
2
3 #include <sys/types.h>
4 #include <sys/stat.h>
5 #include <dirent.h>
   #include <unistd.h>
7 #include <string.h>
8 #include <stdio.h>
9
   #include <errno.h>
10
int dopath(MyFunc *func, char *filename, int len);
12
13 | int myftw(char *pathname, MyFunc *func)
14
15
       return dopath(func, pathname, 0);
16
   }
17
18
   int dopath(MyFunc *func, char *filename, int len)
19
20
       struct stat
                        statbuf;
21
       struct dirent
                        *dirp;
22
       DIR
                        *dp;
23
       int
                        ret;
24
25
       if (lstat(filename, &statbuf) < 0)</pre>
26
            return func(filename, &statbuf, FTW_NS, len);
27
28
       if (S_ISDIR(statbuf.st_mode) == 0)
29
            return func(filename, &statbuf, FTW_F, len);
30
31
       ret = func(filename, &statbuf, FTW_D, len);
32
33
       if (ret != 0)
34
           return ret;
35
36
       dp = opendir(filename);
37
       if (dp == NULL)
38
```

```
39
            printf("DEBUG2");
40
            return func(filename, &statbuf, FTW_DNR, len);
41
        }
42
        if (chdir(filename) < 0)</pre>
43
44
        {
45
            printf("Cannot chdir into %s\n", filename);
46
            return func(filename, &statbuf, FTW_DNR, len);
47
        }
48
49
       ++len;
50
       while ((dirp = readdir(dp)) != NULL)
51
52
53
            if (strcmp(dirp->d_name, ".") != 0 &&
54
                strcmp(dirp->d_name, "..") != 0)
55
                dopath(func, dirp->d_name, len);
56
        }
57
58
        if (chdir("..") < 0)
59
        {
60
            printf("Cannot return into .. from %s", filename);
61
            return -1;
62
        }
63
64
        if (closedir(dp) < 0)</pre>
65
        {
66
            printf("Can't close directory %s\n", filename);
67
            return -1;
68
        }
69
70
        return ret;
71
```

Листинг 1.4 – Нерекурсивная реализация myftw

```
#include "myftw.h"

#include <sys/types.h>
#include <sys/stat.h>

#include <dirent.h>
#include <unistd.h>

#include <string.h>
#include <stdio.h>
#include <errno.h>

#include "stack.h"

#include "stack.h"
```

```
13
   int dopath(MyFunc *func, char *filename);
14
15
   int myftw(char *pathname, MyFunc *func)
16
17
       return dopath(func, pathname);
18
19
20
   int dopath(MyFunc *func, char *filename)
21
22
        struct stat
                         statbuf;
23
       struct dirent
                         *dirp;
24
       DIR
                         *dp;
25
       int
                         len = 0;
26
27
       struct stack *stack = stack_create();
28
       stack_push(stack, filename);
29
30
       while (!stack_is_empty(stack))
31
32
            const char *filename = stack_pop(stack);
33
34
            if (strcmp(filename, "..") == 0)
35
36
                --len;
37
38
                if (chdir(filename) < 0)</pre>
39
                {
40
                    printf("Cannot return into .. from");
41
                    return -1;
42
                }
43
44
            else if (lstat(filename, &statbuf) < 0)</pre>
45
                func(filename, &statbuf, FTW_NS, len);
            else if (S_ISDIR(statbuf.st_mode) == 0)
46
47
                func(filename, &statbuf, FTW_F, len);
48
            else
49
            {
50
                func(filename, &statbuf, FTW_D, len);
51
52
                dp = opendir(filename);
53
54
                if (dp == NULL)
55
                     func(filename, &statbuf, FTW DNR, len);
56
57
                if (chdir(filename) < 0)</pre>
58
59
                    printf("Cannot chdir to %s\n", filename);
60
                    func(filename, &statbuf, FTW_DNR, len);
```

```
61
62
                else
63
64
                     ++len;
65
66
                     stack_push(stack, "..");
67
                     while ((dirp = readdir(dp)) != NULL)
68
69
70
                         if (strcmp(dirp->d_name, ".") != 0 &&
                             strcmp(dirp->d_name, "..") != 0)
71
72
                             stack_push(stack, dirp->d_name);
73
                     }
74
                }
75
            }
76
77
78
        return 0;
79
```

Листинг 1.5 – Объявление структуры стека и функций взаимодействия с ним

```
1 #ifndef OSLAB02_STACK_H_
2
   #define OSLAB02_STACK_H_
3
  #define STACKLIM 256
4
6
  struct stack;
7
   struct stack *stack_create();
9
   void stack_free(struct stack **stack);
10
11
   int stack_is_empty(struct stack *stack);
12
13 | int stack_push(struct stack *stack, char *item);
14
   const char *stack_pop(struct stack *stack);
15
16
   #endif // OSLAB02 STACK H
```

Листинг 1.6 - Определение структуры стека и функций взаимодействия с ним

```
#include "stack.h"

#include <stdlib.h>

struct stack
```

```
6 | {
 7
       char *data[STACKLIM];
 8
       int size;
9 | };
10
11 | struct stack *stack_create()
12 | {
13
       struct stack *stack = malloc(sizeof(struct stack));
14
       stack->size = 0;
15
16
       return stack;
17 }
18
19 | int stack_is_empty(struct stack *stack)
20 | {
21
       return (stack->size == 0);
22 }
23
24 | int stack_push(struct stack *stack, char *element)
25 | {
26
       if (stack->size >= STACKLIM)
27
           return -1;
28
29
       stack->data[stack->size++] = element;
30
31
       return 0;
32 | }
33
34 | const char *stack_pop(struct stack *stack)
35
36
       return stack->data[--stack->size];
37 | }
38
39 void stack_free(struct stack **stack)
40
41
       if (stack != NULL && *stack != NULL)
42
           free(*stack);
43
44
       *stack = NULL;
45 }
```

1.3 Результаты работы

Рис. 1.1 – Сборка рекурсивной реализации и демонстрация запуска программы с неверными аргументами командной строки

Рис. 1.2 – Демонстрация работы рекурсивной реализации на примере каталога /dev, часть 1

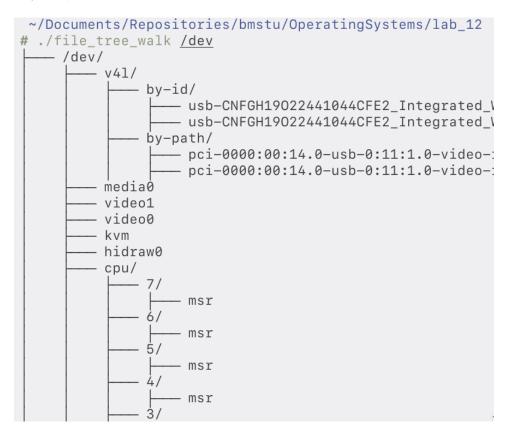


Рис. 1.3 – Демонстрация работы рекурсивной реализации на примере каталога /dev, часть 2

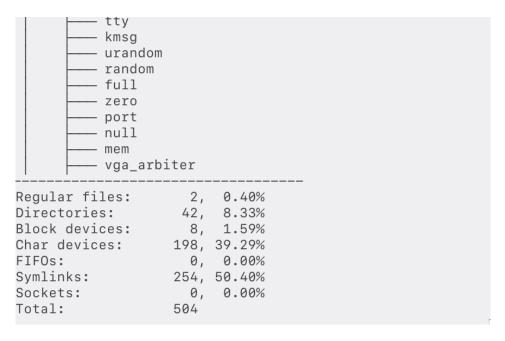


Рис. 1.4 – Сборка нерекурсивной реализации и её демонстрация на примере каталога /dev, часть 1

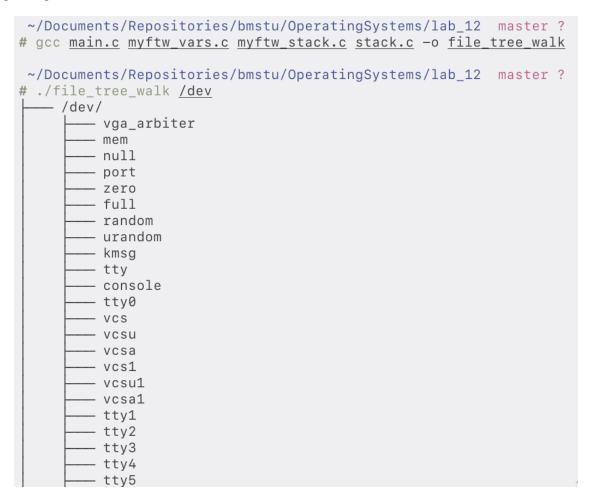
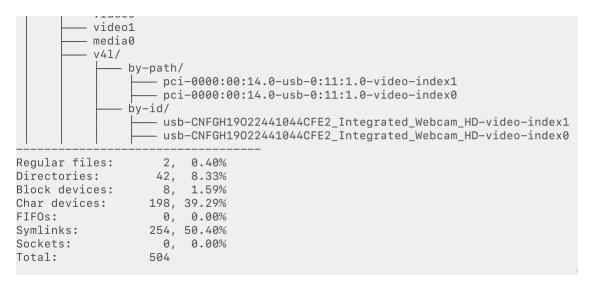


Рис. 1.5 — Демонстрация работы нерекурсивной реализации на примере каталога /dev, часть 2



1.4 Комментарии к программе

- chdir(const char *path) функция смены рабочего каталога. Она устанавливает рабочий каталог, указанный в аргументе path.
- opendir(const char *name) функция, которая открывает директорию с именем пате для чтения и возвращает указатель на структуру типа DIR.
- closedir(DIR *dirp) функция, которая закрывает поток каталога, на который указывает dirp.
- После обработки всех файлов в каталоге вызывается функция chdir(".."), чтобы использовать короткие имена.
- Функция stat заполняет структуру struct stat информацией о определённом файле. Если необходимо обнаружить символьные ссылки, следует использовать функцию lstat.
- lstat(const char *filename, struct stat *buf) возвращает информацию о файле с именем filename и заполняет буфер buf.
- S_ISDIR макрос для поля st_mode, определяющий, является ли тип файла каталогом.