

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Преподаватель Рязанова Н.Ю.

Москва, 2020 г.

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 `main`

```
1 #include <stdio.h>
2 #include <string.h>
3
4 #include "myftw.h"
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
18    ntot = nreg + ndir +  nblk + nchr +  nfifo + nslink + nsock;
19    if (ntot == 0)
20        ntot = 1;
21
22    char hrule[256];
23    memset(hrule, '-', sizeof(hrule));
24
25    printf("%.5s\n", 35, hrule);
26    printf("Regular files: %7ld, %5.2f%%\n", nreg,    nreg    * 100.0/ntot);
```

```

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:         %7ld, %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:         %7ld\n",          ntot);
34
35     return ret;
36 }

```

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

```

1  #ifndef OSLAB02_MYFTW_H_
2  #define OSLAB02_MYFTW_H_
3
4  #include <stdio.h>
5  #include <string.h>
6  #include <sys/stat.h>
7
8  #define FTW_F    1 // файл, не являющийся каталогом
9  #define FTW_D    2 // каталог
10 #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
15 typedef int MyFunc(const char *, const struct stat *, int, int);
16
17 static int myfunc(const char *pathname,
18                  const struct stat *statptr,
19                  int type, int len)
20 {
21     for (int i = 0; i < len; i++)
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] == '/')
65         printf("%s\n", pathname);
66     else
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
84     return 0;
85 }
86
87 int myftw(char *pathname, MyFunc *func);
88
89 #endif // OSLAB02_MYFTW_H_

```

Листинг 1.3 – Рекурсивная реализация myftw

```

1  #include "myftw.h"
2
3  #include <sys/types.h>
4  #include <sys/stat.h>
5  #include <dirent.h>
6  #include <unistd.h>
7  #include <string.h>
8  #include <stdio.h>
9  #include <errno.h>
10
11 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)
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
43 if (chdir(filename) < 0)
44 {
45     printf("Cannot chdir into %s\n", filename);
46     return func(filename, &statbuf, FTW_DNR, len);
47 }
48
49 ++len;
50
51 while ((dirp = readdir(dp)) != NULL)
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)
65 {
66     printf("Can't close directory %s\n", filename);
67     return -1;
68 }
69
70 return ret;
71 }

```

1.3 Результаты работы

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

```
~/Documents/Repositories/bmstu/OperatingSystems/lab_12
# gcc main.c myftw_vars.c myftw_rec.c -o file_tree_walk

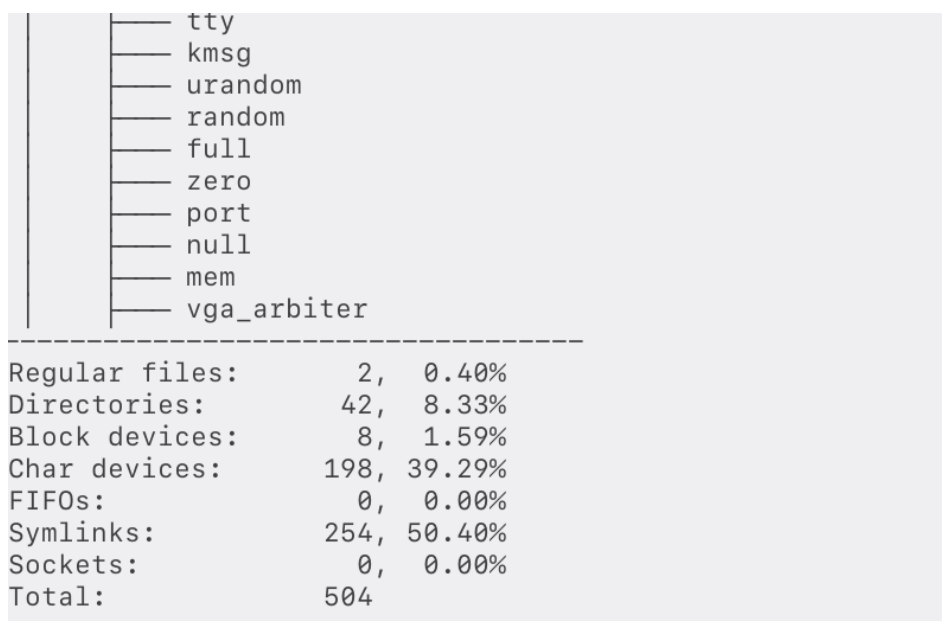
~/Documents/Repositories/bmstu/OperatingSystems/lab_12
# ./file_tree_walk
Usage:
  ./file_tree_walk <input directory>

~/Documents/Repositories/bmstu/OperatingSystems/lab_12
# ./file_tree_walk nonexistent_path
Function error stat: No such file or directory
```

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

```
~/Documents/Repositories/bmstu/OperatingSystems/lab_12
# ./file_tree_walk /dev
├── /dev/
│   ├── v4l/
│   │   ├── by-id/
│   │   │   ├── usb-CNFGH19022441044CFE2_Integrated_V
│   │   │   └── usb-CNFGH19022441044CFE2_Integrated_V
│   │   └── by-path/
│   │       ├── pci-0000:00:14.0-usb-0:11:1.0-video-:
│   │       └── pci-0000:00:14.0-usb-0:11:1.0-video-:
│   ├── media0
│   ├── video1
│   ├── video0
│   ├── kvm
│   ├── hidraw0
│   └── cpu/
│       ├── 7/
│       │   └── msr
│       ├── 6/
│       │   └── msr
│       ├── 5/
│       │   └── msr
│       ├── 4/
│       │   └── msr
│       └── 3/
```

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



```

tty
kmsg
urandom
random
full
zero
port
null
mem
vga_arbiter
-----
Regular files:      2,   0.40%
Directories:       42,   8.33%
Block devices:      8,   1.59%
Char devices:     198,  39.29%
FIFOs:              0,   0.00%
Symlinks:          254,  50.40%
Sockets:           0,   0.00%
Total:             504
```

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 `name` 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 символичные 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.