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
по операционным системам

Студент Колганов О.С.

Группа ИУ7 — 62Б

Преподаватель Рязанова Н.Ю.

Москва.
2020 г.

Текст программы:

```
#include <dirent.h>
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>


#define FTW_F 1 //файл, не являющийся каталогом
#define FTW_D 2 //каталог
#define FTW_DNR 3 //каталог, недоступный для чтения
#define FTW_NS 4 //файл, информацию о котором нельзя получить с помощью stat


// функция, которая будет вызываться для каждого встреченного файла
typedef int Handler(const char * ,const struct stat *, int);


static Handler counter;
static int my_ftw(char *, Handler * );
static int dopath(const char* filename, int depth, Handler * );


static long nreg, ndir, nblk, nchr, nfifo, nslink, nsock, ntot;


int main(int argc, char * argv[])
{
   int ret = -1;
   if (argc != 2)
   {
      printf("Использование: ftw <начальный каталог>\n");
```

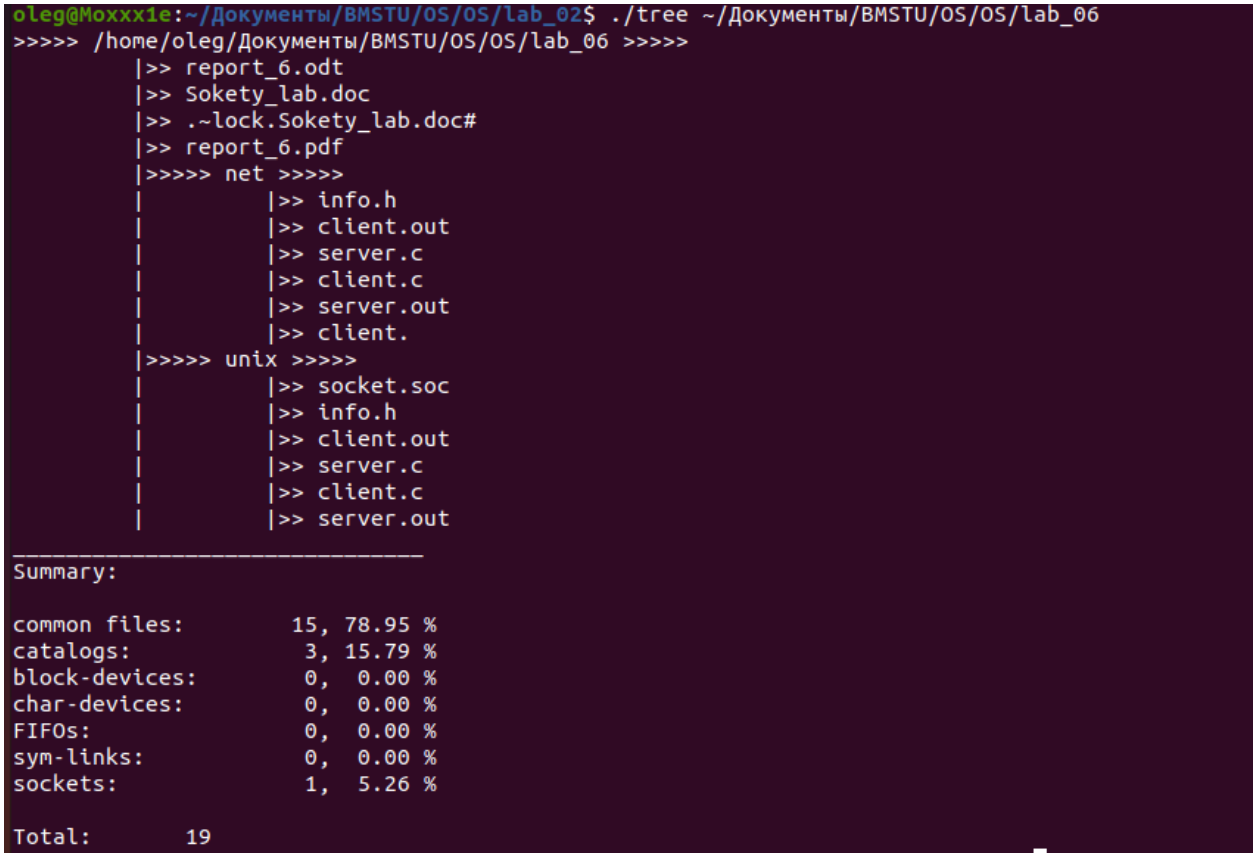
```
      exit(-1);
   }



   ret = my_ftw(argv[1], counter); //выполняет всю работу


   ntot = nreg + ndir +  nblk + nchr +  nfifo + nslink + nsock;


    if (ntot == 0)
           ntot = 1; //во избежание деления на 0; вывести 0 для всех
счетчиков


     printf("_____________________________________\nSummary:\n\n");
     printf("common files:\t%7ld, %5.2f %%\n", nreg,
nreg*100.0/ntot);
     printf("catalogs:\t%7ld, %5.2f %%\n", ndir, ndir*100.0/ntot);
     printf("block-devices:\t%7ld, %5.2f %%\n", nblk,
nblk*100.0/ntot);
     printf("char-devices:\t%7ld, %5.2f %%\n", nchr,
nchr*100.0/ntot);
     printf("FIFOs:\t\t%7ld, %5.2f %%\n", nfifo, nfifo*100.0/ntot);
     printf("sym-links:\t%7ld, %5.2f %%\n", nslink,
nslink*100.0/ntot);
     printf("sockets:\t%7ld, %5.2f %%\n\n", nsock, nsock*100.0/ntot);
    printf("Total:\t%7ld\n", ntot);


    exit(ret);
}


//обходит дерево каталогов, начиная с pathname и применяя к каждому
файлу func
static int my_ftw(char * pathname, Handler * func)
{
    return(dopath(pathname, 0, func));
```

```
}

//обход дерева каталогов, начиная с fullpath
static int dopath(const char* filename, int depth, Handler * func)
{
    struct stat statbuf;
    struct dirent * dirp;
    DIR * dp;
    int ret;

    if (lstat(filename, &statbuf) < 0) //ошибка вызова функции lstat
        return(func(filename, &statbuf, FTW_NS));

    for (int i = 0; i < depth; ++i)
           printf("         |");

    if (S_ISDIR(statbuf.st_mode) == 0) // файл не является каталогом
        return(func(filename, &statbuf, FTW_F)); //отобразить в дереве

    if ((ret = func(filename, &statbuf, FTW_D)) != 0)
        return(ret);

    if ((dp = opendir(filename)) == NULL) //каталог недоступен
        return(func(filename, &statbuf, FTW_DNR));

    chdir(filename);
    while ((dirp = readdir(dp)) != NULL || ret != 0) {
        if (!(strcmp(dirp->d_name, ".") == 0 ||
            strcmp(dirp->d_name, "..") == 0)) {
            ret = dopath(dirp->d_name, depth + 1, func);
        }
    }
```

```
    chdir("..");

    if (closedir(dp) < 0)
        perror("Невозможно закрыть каталог");

    return(ret); // вернулись в родительский каталог
}

static int counter(const char* pathame, const struct stat * statptr,
int type)
{
    switch(type)
    {
        case FTW_F:
            printf( ">> %s\n", pathame);
            switch(statptr->st_mode & S_IFMT)
            {
                case S_IFREG: nreg++; break;
                case S_IFBLK: nblk++; break;
                case S_IFCHR: nchr++; break;
                case S_IFIFO: nfifo++; break;
                case S_IFLNK: nslink++; break;
                case S_IFSOCK: nsock++; break;
                case S_IFDIR:
                    perror("Католог имеет тип FTW_F"); return(-1);
            }
            break;
        case FTW_D:
            printf( ">>>>> %s >>>>>\n", pathame);
            ndir++; break;
        case FTW_DNR:
            perror("Закрыт доступ к одному из каталогов!"); return(-
```

```
1);
        case FTW_NS:
            perror("Ошибка функции stat!"); return(-1);
        default:
            perror("Неизвестый тип файла!"); return(-1);
    }

    return(0);
}
```

Демонстрация работы программы



Ответы на вопросы

1. Для чего нужно использовать chdir()?

- Для использования коротких имён.

2. Перечислить условия выхода из рекурсии.

а) ошибка вызова lstat

б) каталог недоступен для чтения

в) файл не является каталогом

г) обход каталога завершён.