\\ файл “main.c”

#define \_DEFAULT\_SOURCE

#include <getopt.h>

#include "utils.c"

int main(int argc, char\* argv[])

{

    char c;

    int option\_index = 0;

    struct option option\_types[] = {

            {"create", required\_argument, 0, 'c'},

            {"remove", required\_argument, 0, 'r'},

            {"dirwalk", required\_argument, 0, 'd'},

            {"listdir", required\_argument, 0, 'l'},

            {"move", required\_argument, 0, 'm'},

            {"chmod", required\_argument, 0, 'h'}

    };

    while((c = getopt\_long(argc, argv, "cr:d:l:mh",

                           option\_types, &option\_index)) != EOF)

    {

        switch (c)

        {

            case 'c':

                if (argc < 3) {

                   printf("Usage: %s <filename> <directory>\n", argv[0]);

                   return 1;

                }

                create(argv[argc - 2], argv[argc - 1]);

                break;

            case 'r':

                remove\_file(optarg);

                break;

            case 'd':

                dirwalk(optarg);

                break;

            case 'l':

                listdir(optarg);

                break;

            case 'm':

                if (argc < 3) { // Check if both arguments are provided

                    printf("Usage: sudo %s <file> <directory>\n", argv[1]);

                    return 1;

                }

                move(argv[argc - 2], argv[argc - 1]);

                break;

            case 'h':

                if (argc < 3) {

                    fprintf(stderr, "Usage: sudo %s <filepath> <mode>\n", argv[1]);

                    return EXIT\_FAILURE;

                }

                change\_mode(argv[argc - 2], argv[argc - 1]);

                break;

            default:

                perror("Incorrect option");

                exit(1);

                break;

        }

    }

}

\\ файл “utils.c”

#define \_DEFAULT\_SOURCE

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <dirent.h>

#include <string.h>

#include <getopt.h>

#include <stdbool.h>

#include <unistd.h>

#include <time.h>

/\* change\_mode function sets file's permission mode given in argv[1] to mode given argv[2] \*/

int change\_mode(char\* filepath, char\* mode\_str) {

    if (!is\_path( filepath))

    {

        perror(filepath);

        exit(2);

    }

    \_\_mode\_t mode = strtol(mode\_str, NULL, 8);

    if (chmod(filepath, mode) == -1) {

        perror("chmod");

        return EXIT\_FAILURE;

    }

    printf("Changed mode of file '%s' to %04o\n", filepath, mode);

    return EXIT\_SUCCESS;

}

/\* create function creates file with name given in argv[1] including file extension \*/

int create(char\* filename, char\* directory) {

    if (!is\_path(directory))

    {

        perror(directory);

        exit(2);

    }

    char filepath[strlen(directory) + strlen(filename) + 2];

    sprintf(filepath, "%s/%s", directory, filename);

    FILE\* fp = fopen(filepath, "w");

    if(fp == NULL) {

        printf("Error creating %s.\n", filename);

        return 1;

    }

    printf("%s created successfully in %s.\n", filename, directory);

    fclose(fp);

    return 0;

}

/\* remove\_file function deletes file given in argv[1] \*/

int remove\_file(char\* filepath) {

    if (!is\_path( filepath))

    {

        perror(filepath);

        exit(2);

    }

    if(remove(filepath) == 0) {

        printf("%s deleted successfully.\n", filepath);

    } else {

        printf("Error deleting %s.\n", filepath);

    }

    return 0;

}

/\* function is\_path checks whether given char\* is correct path or not. Return 1 if it is, 0 if it isn't , otherwise it's error\*/

int is\_path(char\* path) {

    struct stat st;

    if (stat(path, &st) == 0) return 1;

    return 0;

}

/\* listfile lists given file like ls -la \*/

void listfile(char\* name)

{

    struct stat sb;

    char \*modtime;

    char \*filetype[] = {"?", "p", "c", "?", "d", "?", "b", "?", ".", "?", "l", "?", "s"};

    if(stat(name, &sb) < 0)

    {

        perror(name);

        exit(2);

    }

    printf("%s", filetype[(sb.st\_mode >> 12) & 017]);

    printf("%c%c%c%c%c%c%c%c%c",

            (sb.st\_mode & S\_IRUSR) ? 'r' : '-',

            (sb.st\_mode & S\_IWUSR) ? 'w' : '-',

            (sb.st\_mode & S\_IXUSR) ? 'x' : '-',

            (sb.st\_mode & S\_IRGRP) ? 'r' : '-',

            (sb.st\_mode & S\_IWGRP) ? 'w' : '-',

            (sb.st\_mode & S\_IXGRP) ? 'x' : '-',

            (sb.st\_mode & S\_IROTH) ? 'r' : '-',

            (sb.st\_mode & S\_IWOTH) ? 'w' : '-',

            (sb.st\_mode & S\_IXOTH) ? 'x' : '-');

    printf("%8ld", sb.st\_size);

    modtime = ctime(&sb.st\_mtime);

    modtime[strlen(modtime) - 1] = '\0';

    printf(" %s ", modtime);

    printf("%s\n", name);

}

/\* lists all files in given directory like ls -la \*/

void listdir(char \* directory)

{

    DIR \*d;

    struct dirent \* info;

    // if(argc != 2){

    //     fprintf(stderr, "usage: listdir dirname\n");

    //     exit(1);

    // }

    if (!is\_path( directory))

    {

        perror(directory);

        exit(2);

    }

    chdir(directory);

    d = opendir(directory);

    while((info = readdir(d)) != NULL)

    {

        if (info->d\_type == DT\_LNK)

        {

            continue;

        }

        listfile(info->d\_name);

    }

    closedir(d);

}

/\* move functions moves file given in argv[1] to dest in argv[2] \*/

int move(char\* filepath, char\* destination) {

    if (!is\_path( filepath))

    {

        perror(filepath);

        exit(2);

    }

    if (!is\_path( destination))

    {

        perror(destination);

        exit(2);

    }

    // Construct the full path of the destination file

    char dest[256];

    snprintf(dest, sizeof(dest), "%s/%s", destination, filepath);

    // Move the file to the destination directory

    if (rename(filepath, dest) < 0) {

        perror("Error moving file");

        return 1;

    }

    printf("%s successfully moved to %s\n",filepath, destination);

    return 0;

}

void recWalk(const char\* dir\_name)

{

    DIR \*directory;

    /\* namelist изначально NULL, тк scandir сам выделяет память, \*/

    struct dirent \*\*namelist = NULL;

    int numOfFiles = 0;

    if ((directory = opendir(dir\_name)) == NULL)

    {

        perror(dir\_name);

        printf("opendir error\n");

        exit(1);

    }

    /\* смена рабочей директории на ту, которую проходим, чтобы в дальнейшем можно было

    открывать директорию с помощью только ее имени\*/

    chdir(dir\_name);

    numOfFiles = scandir(".", &namelist, NULL, alphasort);

    if(numOfFiles == -1)

    {

        perror("Scandir error");

        exit(1);

    }

    while (numOfFiles--)

    {

        printf("%s\n", namelist[numOfFiles]->d\_name);

        if (namelist[numOfFiles]->d\_type == DT\_DIR

        && namelist[numOfFiles]->d\_name[0] != '.')

        {

            recWalk((const char\*)namelist[numOfFiles]->d\_name);

        }

    }

    free(namelist);

    closedir(directory);

    chdir("..");

}

/\* dirwalk function recursively traverses files in given directory\*/

int dirwalk(char\* directory)

{

    if (!is\_path( directory))

    {

        perror(directory);

        exit(2);

    }

    recWalk(directory);

}

\\ файл “Makefile”

CC = gcc

CFLAGS = -c -std=c11 -pedantic -W -Wall -Wextra

.PHONY:all

all: start

start:

    sudo $(CC) main.c utils.c  $(CFLAGS)

    $(CC) main.o -o ~bin/ufs\_si

clean:

    rm -rf main.o utils.o