Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации Сибирский Государственный Университет Телекоммуникаций и Информатики СибГУТИ

Кафедра вычислительных систем

**Лабораторная работа №1**

по дисциплине «Операционные системы»

Выполнил:

Студент группы ИВ-921

Ярошев Р.А.

Работу проверил:

ассистент кафедры ВС

Петрук Е. А.

Новосибирск 2021

## Задание

Реализовать простой файловый менеджер для ОС GNU/Linux, поддерживающий следующий (минимальный) набор операций:

• Создание файла;

• Просмотр содержимого файла;

• Перемещение, копирование;

• Удаление;

• Создание ссылки на файл;

• Просмотр списка файлов в директории;

Список не исчерпывающий, можно по желанию добавить любые другие операции.

Файловый менеджер может быть реализован как простая программа для терминала, принимающая от пользователя команду и список файлов, завершающаяся после выполнения:

$ tfm - command [ filename1 filename2 ...]

Так и в виде консольной программы с интерактивным взаимодействием:

tfm >

tfm > create

Enter filename ...

tfm > test . txt

Succesfull !

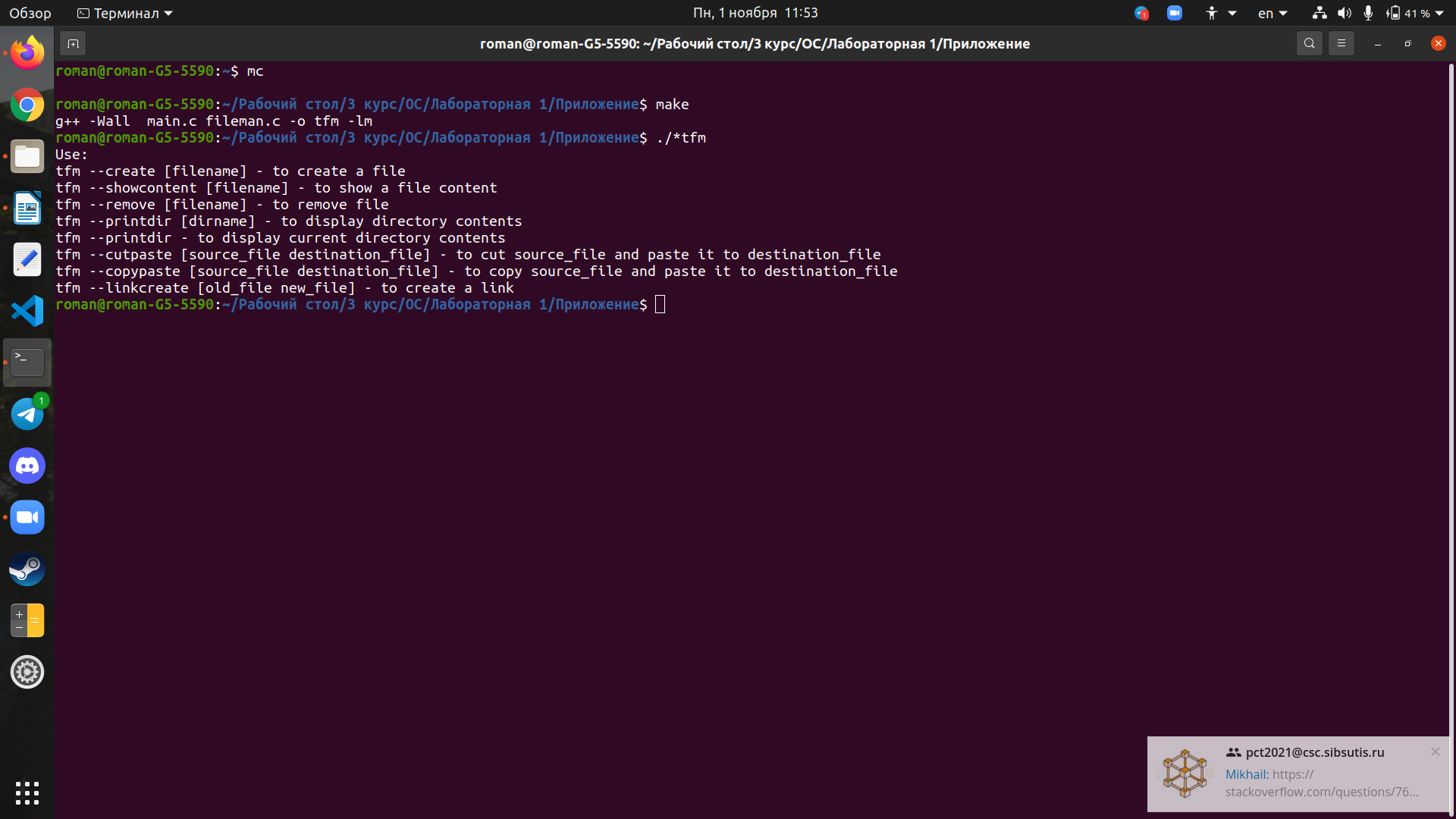
tfm >

Список опять же неполный, по желанию можно реализовать псевдографический интерфейс (библиотека ncurses) или полноценную графику. Выбранный вариант на оценку не влияет.

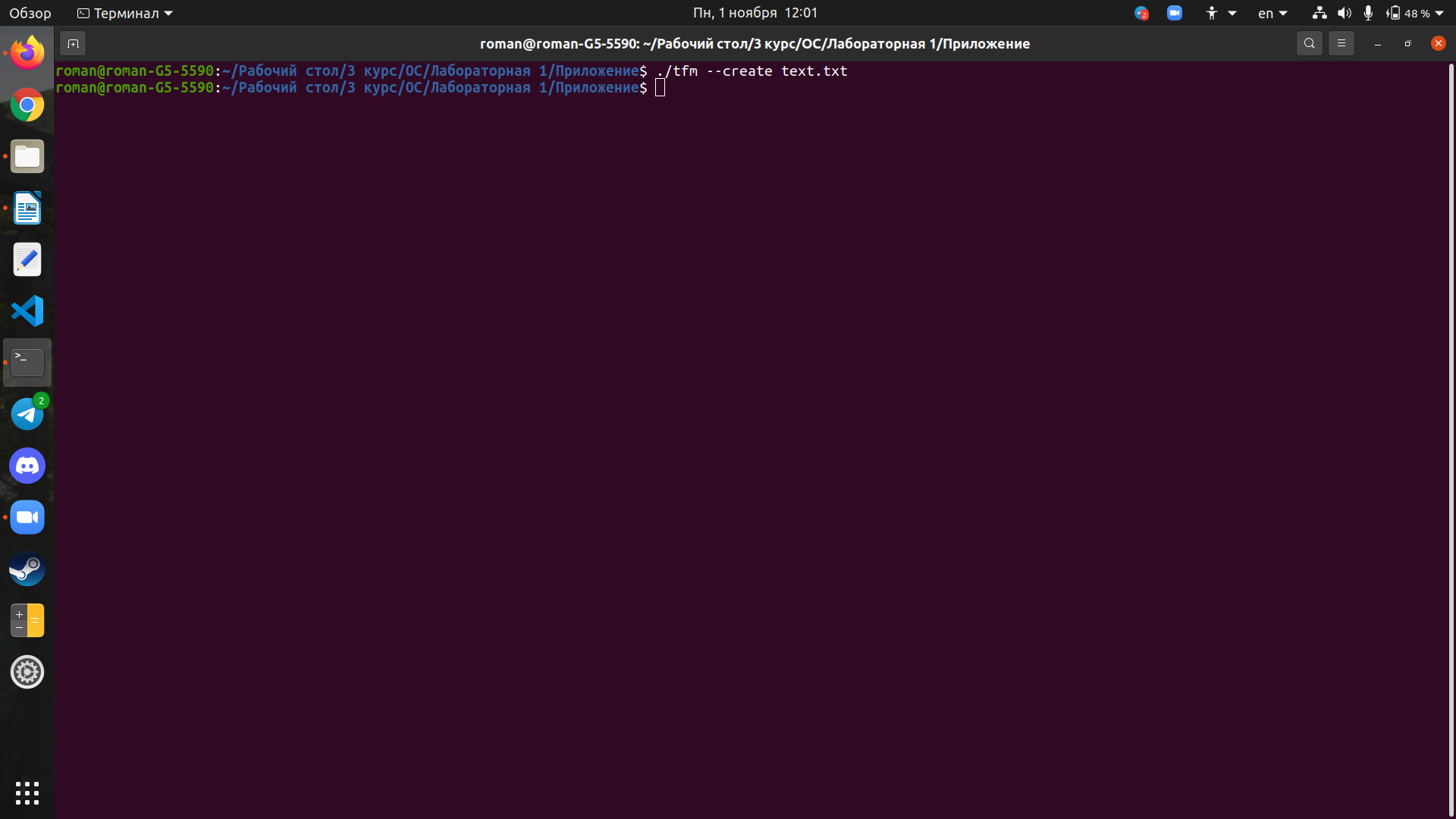
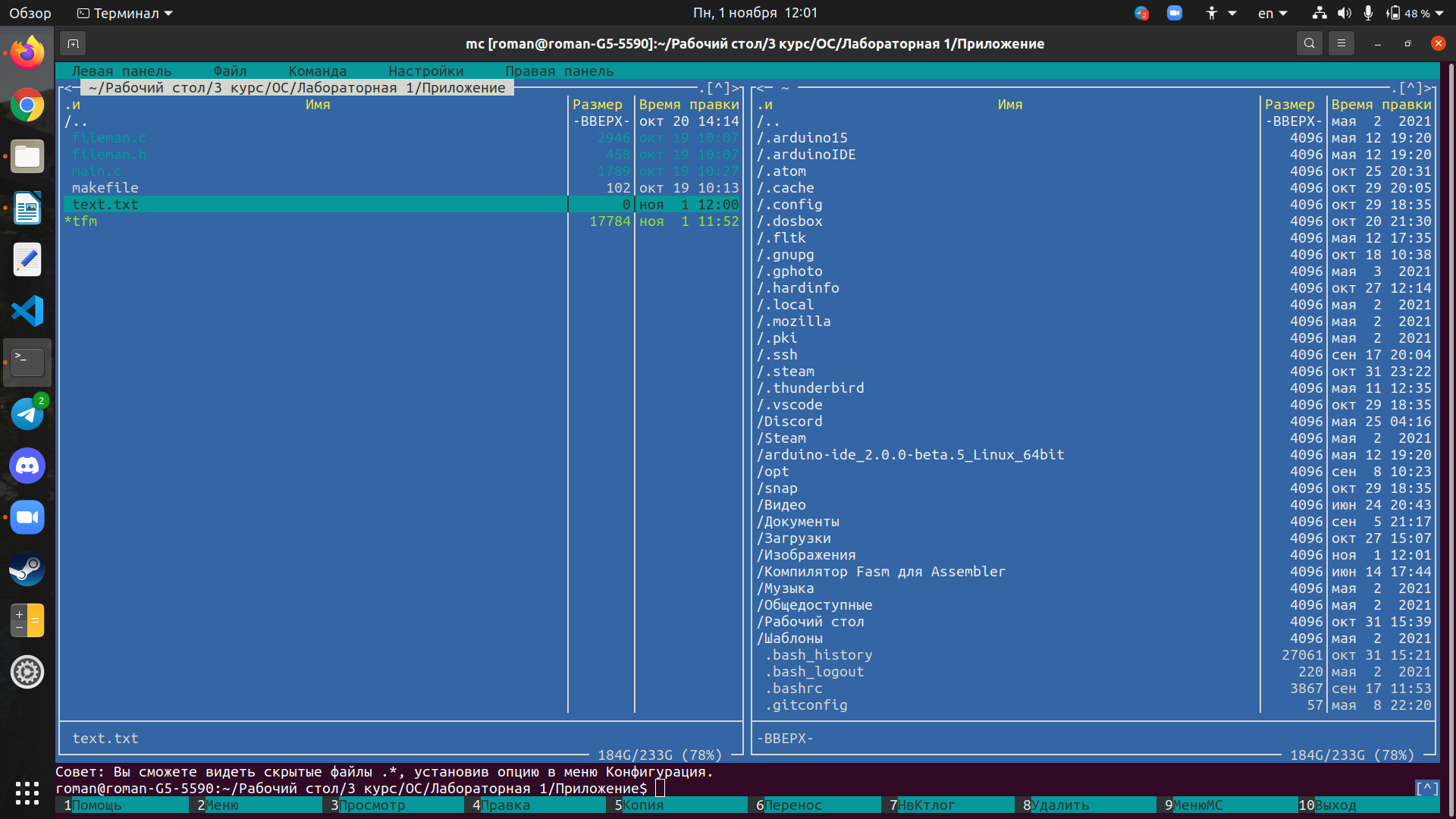
Для доступа к файловой системе из вашей программы нужно использовать либо непосредственно системные вызовы open(), write(), read(), close(), fsync() ..., либо стандартную библиотеку stdio (fopen(), fwrite(), printf() ...). Никаких вызовов system()! >:(. Выбранный вариант так же не влияет на оценку, но при сдаче нужно быть в состоянии объяснить разницу.

## Результат работы программы

Собираем приложение — компилируем и запускаем:

Рис. 1. Сборка приложения

Тут видим хелпер с доступными командами. Чтобы перейти к исполнению команд, делаем следующее:

Рис. 2. Создания файла

## Листинг

**Fileman.c**

#include <dirent.h>

#include <errno.h>

#include <fcntl.h>

#include <stdio.h>

#include <string.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int create\_file(const char \*pathToFile) {

int fd;

fd = open(pathToFile, O\_CREAT | O\_EXCL, S\_IRUSR | S\_IWUSR);

if (fd == -1) {

perror("File creation error");

return -1;

}

if (close(fd) == -1) {

perror("File closing error");

return -1;

}

return 0;

}

int view\_file\_content(const char \*pathToFile) {

int fd;

fd = open(pathToFile, O\_RDONLY);

if (fd == -1) {

perror("Сontent view error");

return -1;

}

ssize\_t ret;

char ch;

while ((ret = read(fd, &ch, 1)) > 0) {

putchar(ch);

}

if (ret == -1) {

perror("Error");

return -1;

}

if (close(fd) == -1) {

perror("File closing error");

return -1;

}

return 0;

}

int remove\_file(const char \*pathToFile) {

if (unlink(pathToFile) == -1) {

perror("File deletion error");

return -1;

}

return 0;

}

int cut\_paste\_file(const char \*srcPathToFile, const char \*dstPathToFile) {

if (rename(srcPathToFile, dstPathToFile) == -1) {

perror("Cut/paste error");

return -1;

}

return 0;

}

int copy\_paste\_file(const char \*srcPathToFile, const char \*dstPathToFile) {

int fdsrc, fddst;

fdsrc = open(srcPathToFile, O\_RDONLY);

if (fdsrc == -1) {

perror("Copy/paste error");

return -1;

}

fddst = open(dstPathToFile, O\_CREAT | O\_EXCL | O\_WRONLY | O\_APPEND,

S\_IRUSR | S\_IWUSR);

if (fddst == -1) {

perror("Copy/paste error");

return -1;

}

ssize\_t ret, nr;

char ch;

while ((ret = read(fdsrc, &ch, 1)) > 0) {

nr = write(fddst, &ch, 1);

if (nr == -1) {

perror("Copy/paste error");

return -1;

}

}

if (ret == -1) {

perror("Copy/paste error");

return -1;

}

if (close(fdsrc) == -1) {

perror("File closing error");

return -1;

}

if (close(fddst) == -1) {

perror("File closing error");

return -1;

}

return 0;

}

int link\_create(const char \*oldPathToFile, const char \*newPathToFile) {

if (link(oldPathToFile, newPathToFile) == -1) {

perror("Link create error");

return -1;

}

return 0;

}

int print\_dir(const char \*path) {

struct dirent \*entry;

DIR \*dir;

dir = opendir(path);

if (dir == NULL) {

perror("Print dir error");

return -1;

}

errno = 0;

while ((entry = readdir(dir)) != NULL) {

if (strcmp(".", entry->d\_name) != 0 &&

strcmp("..", entry->d\_name) != 0) {

printf("%s ", entry->d\_name);

}

}

printf("**\n**");

if (errno && !entry) {

perror("Print dir error");

}

closedir(dir);

return 0;

}

**main.c**

#include "fileman.h"

#include <stdio.h>

#include <string.h>

void help() {

printf("Use:**\n**");

printf("tfm --create [filename] - to create a file**\n**");

printf("tfm --showcontent [filename] - to show a file content**\n**");

printf("tfm --remove [filename] - to remove file**\n**");

printf("tfm --printdir [dirname] - to display directory contents**\n**");

printf("tfm --printdir - to display current directory contents**\n**");

printf("tfm --cutpaste [source\_file destination\_file] - to cut source\_file and paste it to destination\_file**\n**");

printf("tfm --copypaste [source\_file destination\_file] - to copy source\_file and paste it to destination\_file**\n**");

printf("tfm --linkcreate [old\_file new\_file] - to create a link**\n**");

}

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

if (argc == 2) {

if (strcmp(argv[1], "--printdir") == 0) {

print\_dir(".");

} else {

help();

}

} else if (argc == 3) {

if (strcmp(argv[1], "--create") == 0) {

create\_file(argv[2]);

} else if (strcmp(argv[1], "--showcontent") == 0) {

view\_file\_content(argv[2]);

} else if (strcmp(argv[1], "--remove") == 0) {

remove\_file(argv[2]);

} else if (strcmp(argv[1], "--printdir") == 0) {

print\_dir(argv[2]);

} else {

help();

}

} else if (argc == 4) {

if (strcmp(argv[1], "--cutpaste") == 0) {

cut\_paste\_file(argv[2], argv[3]);

} else if (strcmp(argv[1], "--copypaste") == 0) {

copy\_paste\_file(argv[2], argv[3]);

} else if (strcmp(argv[1], "--linkcreate") == 0) {

link\_create(argv[2], argv[3]);

} else {

help();

}

} else {

help();

}

return 0;

}