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:

```
roman@roman-G5-5590:~/Рабочий стол/3 курс/ОС/Лабораторная 1/Приложение$ make
g++ -Wall  main.c fileman.c -o tfm -lm
roman@roman-G5-5590:~/Рабочий стол/3 курс/ОС/Лабораторная 1/Приложение$ ./*tfm
Use:
tfm --create [filename] - to create a file
tfm --showcontent [filename] - to show a file content
tfm --remove [filename] - to remove file
tfm --printdir [dirname] - to display directory contents
tfm --printdir - to display current directory contents
tfm --cutpaste [source_file destination_file] - to cut source_file and paste it to destination_file
tfm --copypaste [source_file destination_file] - to copy source_file and paste it to destination_file
tfm --linkcreate [old_file new_file] - to create a link
```

Рис. 1. Сборка приложения

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

```
roman@roman-G5-5590:~/Рабочий стол/3 курс/ОС/Лабораторная 1/Приложение$ ./tfm --create text.txt
roman@roman-G5-5590:~/Рабочий стол/3 курс/ОС/Лабораторная 1/Приложение$
```

```
 Левая панель     Файл     Команда     Настройки     Правая панель
<- ~/Рабочий стол/3 курс/ОС/Лабораторная 1/Приложение ----------.[^]>
.и                        Имя          Размер  Время правки
/..                                    -ВВЕРХ- окт 20 14:14
 fileman.c                               2946 окт 19 10:07
 fileman.h                                458 окт 19 10:07
 main.c                                  1789 окт 19 10:27
 makefile                                 102 окт 19 10:13
 text.txt                                   0 ноя  1 12:00
*tfm                                    17784 ноя  1 11:52
```

Рис. 2. Создания файла

Листинг

Fileman.c

```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("Content 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

```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;
}
```