

# **Лабораторная работа-05**

**Файловая система Linux**

**Бровкин Александр НБИбд-01-21**

# **Содержание**

<b>1 Цель работы</b>	<b>5</b>
<b>2 Задание</b>	<b>6</b>
<b>3 Выполнение лабораторной работы</b>	<b>8</b>
<b>4 Выводы</b>	<b>23</b>
<b>5 Ответы на контрольные вопросы:</b>	<b>24</b>
<b>Список литературы</b>	<b>27</b>

# Список иллюстраций

3.1	Выполняю примеры из лабораторной . . . . .	8
3.2	Продолжнаю выполнять примеры . . . . .	9
3.3	Продолжнаю выполнять примеры . . . . .	9
3.4	Продолжнаю выполнять примеры . . . . .	10
3.5	fsck . . . . .	10
3.6	Продолжнаю выполнять примеры . . . . .	11
3.7	Продолжнаю выполнять примеры . . . . .	12
3.8	australia . . . . .	12
3.9	На скриншоте все ответы на данные пункты . . . . .	15
3.10	команда man . . . . .	16
3.11	mount . . . . .	16
3.12	fsck . . . . .	19
3.13	mkfs . . . . .	21
3.14	kill . . . . .	21

# **Список таблиц**

# **1 Цель работы**

Ознакомление с файловой системой Linux, её структурой, именами и содержанием каталогов. Приобретение практических навыков по применению команд для работы с файлами и каталогами, по управлению процессами (и работами), по проверке использования диска и обслуживанию файловой системы.

## 2 Задание

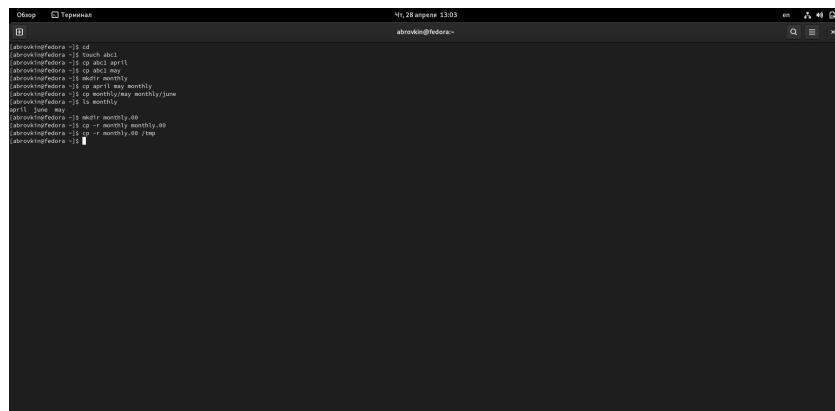
1. Выполните все примеры, приведённые в первой части описания лабораторной работы.
2. Выполните следующие действия, зафиксировав в отчёте по лабораторной работе используемые при этом команды и результаты их выполнения:
  - 2.1. Скопируйте файл '/usr/include/sys/io.h' в домашний каталог и назовите его equipment. Если файла io.h нет, то используйте любой другой файл в каталоге /usr/include/sys/ вместо него.
  - 2.2. В домашнем каталоге создайте директорию ~/ski.plases.
  - 2.3. Переместите файл equipment в каталог ~/ski.plases.
  - 2.4. Переименуйте файл ~/ski.plases/equipment в ~/ski.plases/equiplist.
  - 2.5. Создайте в домашнем каталоге файл abc1 и скопируйте его в каталог '~/ski.plases', назовите его equiplist2.
  - 2.6. Создайте каталог с именем equipment в каталоге ~/ski.plases.
  - 2.7. Переместите файлы ~/ski.plases/equiplist и equiplist2 в каталог ~/ski.plases/equipment.
  - 2.8. Создайте и переместите каталог '~/newdir' в каталог '~/ski.plases' и назовите его plans.
3. Определите опции команды chmod, необходимые для того, чтобы присвоить перечисленным ниже файлам выделенные права доступа, считая, что в начале таких прав нет:
  - 3.1. drwxr--r-- ... australia
  - 3.2. drwxr-x-x ... play
  - 3.3. -r--xr--r-- ... my\_os
  - 3.4. -rw-rw-r-- ... feathersПри необходимости создайте нужные файлы.
4. Проделайте приведённые ниже упражнения, записывая в отчёт по лабораторной работе используемые при этом команды:
- 4.1. Просмотрите содержи-

мое файла /etc/password. 4.2. Скопируйте файл ~/feathers в файл ~/file.old. 4.3. Переместите файл ~/file.old в каталог ~/play. 4.4. Скопируйте каталог ~/play в каталог ~/fun. 4.5. Переместите каталог ~/fun в каталог ~/play и назовите его games. 4.6. Лишите владельца файла ~/feathers права на чтение. 4.7. Что произойдёт, если вы попытаетесь просмотреть файл ~/feathers командой cat? 4.8. Что произойдёт, если вы попытаетесь скопировать файл ~/feathers? 4.9. Дайте владельцу файла ~/feathers право на чтение. 4.10. Лишите владельца каталога ~/play права на выполнение. 4.11. Перейдите в каталог ~/play. Что произошло? 4.12. Дайте владельцу каталога ~/play право на выполнение.

5. Прочитайте man по командам mount, fsck, mkfs, kill и кратко их охарактеризуйте, приведя примеры.

### 3 Выполнение лабораторной работы

1. Выполнил все примеры, приведённые в первой части описания лабораторной работы. Скопировал файл ~/abc1 в файл april и в файл may. Скопировал файлы april и may в каталог monthly. Скопировал файл monthly/may в файл с именем june. Скопировал каталог monthly в каталог monthly.00. Скопировал каталог monthly.00 в каталог /tmp (см.рис. 3.1)



```
abrovin@fedora: ~$ cd
abrovin@fedora: ~$ touch abc1
abrovin@fedora: ~$ cp abc1 april
abrovin@fedora: ~$ cp abc1 may
abrovin@fedora: ~$ ls -l
abrovin@fedora: ~$ cp may june
abrovin@fedora: ~$ cp monthly monthly/june
abrovin@fedora: ~$ ls monthly
june
abrovin@fedora: ~$ mv monthly june
abrovin@fedora: ~$ rm monthly
abrovin@fedora: ~$ cp -r monthly.00 /tmp
abrovin@fedora: ~$
```

Рис. 3.1: Выполняю примеры из лабораторной

Изменил название файла april на july в домашнем каталоге. Переместил файл july в каталог monthly.00. Переименовал каталог monthly.00 в monthly.01. Переместил каталог monthly.01 в каталог reports. Переименовал каталог reports/monthly.01 в reports/monthly (см.рис. 3.2).

```

abrovin@fedora: ~$ mkdir monthly
abrovin@fedora: ~$ cd monthly
abrovin@fedora: ~/monthly$ touch reports
abrovin@fedora: ~/monthly$ chmod 700 reports
abrovin@fedora: ~/monthly$ ls -l
total 0
abrovin@fedora: ~/monthly$ cd reports
abrovin@fedora: ~/monthly/reports$ touch monthly
abrovin@fedora: ~/monthly/reports$ chmod 600 monthly
abrovin@fedora: ~/monthly/reports$ ls -l
total 0

```

Рис. 3.2: Продолжна выполннять примеры

Создал файл `~/may` с правом выполнения для владельца. Лишил владельца файла `~/may` права на выполнение. Создал каталог `monthly` с запретом на чтение для членов группы и всех остальных пользователей. Создал файл `~/abc1` с правом записи для членов группы.(см.рис. 3.3)

```

abrovin@fedora: ~$ touch may
abrovin@fedora: ~$ ls -l
total 0
abrovin@fedora: ~$ chmod +x may
abrovin@fedora: ~$ ./may
abrovin@fedora: ~$ rm may
abrovin@fedora: ~$ ls -l
total 0
abrovin@fedora: ~$ cd monthly
abrovin@fedora: ~/monthly$ touch abc1
abrovin@fedora: ~/monthly$ chmod 600 abc1
abrovin@fedora: ~/monthly$ ls -l
total 0
abrovin@fedora: ~/monthly$ cd abc1
abrovin@fedora: ~/monthly/abc1$ touch monthly
abrovin@fedora: ~/monthly/abc1$ chmod 400 monthly
abrovin@fedora: ~/monthly/abc1$ ls -l
total 0

```

Рис. 3.3: Продолжна выполннять примеры

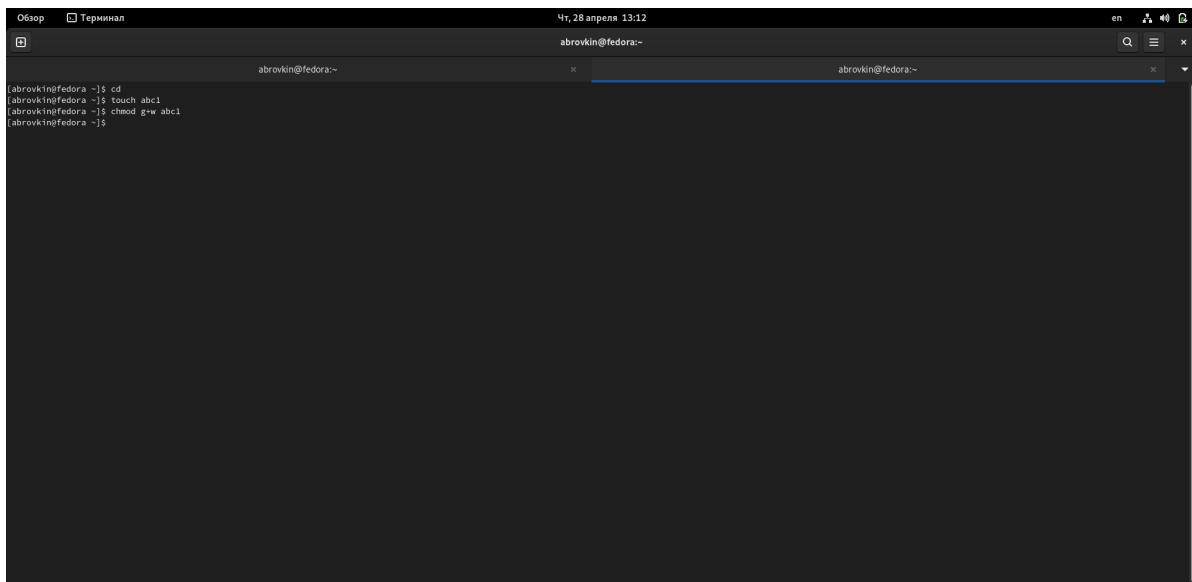


Рис. 3.4: Продолжаю выполнять примеры

Воспользовался командой df, которая выведет на экран список всех файловых систем в соответствии с именами устройств, с указанием размера и точки монтирования, для определения объема свободного пространства на файловой системе. С помощью команды fsck проверил целостность файловой системы.(см.рис. 3.5)

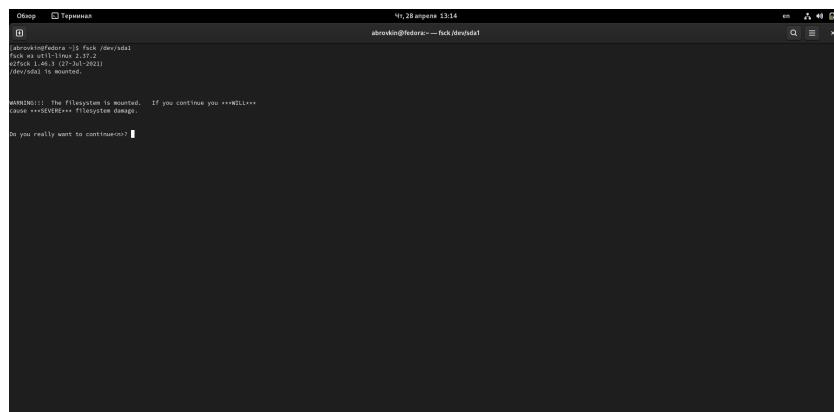


Рис. 3.5: fsck

2. Выполнил следующие действия, зафиксировав в отчёте по лабораторной работе используемые при этом команды и результаты их выполнения: 2.1. Скопировал файл /usr/include/sys/io.h в домашний каталог, с помощью команды cp и назвала его equipment, с помощью команды mv.

- 2.2. В домашнем каталоге создал директорию `~/ski.plases`.
- 2.3. Переместил файл `equipment` в каталог `~/ski.plases` командой `mv`.
- 2.4. Переименовал файл `~/ski.plases/equipment` в `~/ski.plases/equiplist` командой `mv`.
- 2.5. Создал в домашнем каталоге файл `abc1` и скопировал его в каталог `~/ski.plases` командой `cp`, назвал его `equiplist2` командой `mv`.
- 2.6. Создал каталог с именем `equipment` в каталоге `~/ski.plases` командой `mkdir`.
- 2.7. Переместил файлы `~/ski.plases/equiplist` и `equiplist2` в каталог `~/ski.plases/equipment` командой `mv`.
- 2.8. Создал и переместил каталог `~/newdir` в каталог `~/ski.plases` командами `mkdir` и `mv` и назвал его `plans` командой `mv`. (см.рис. 3.6)

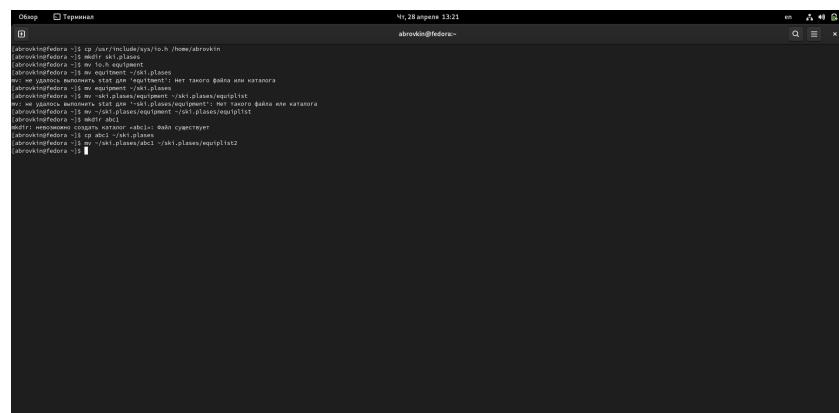
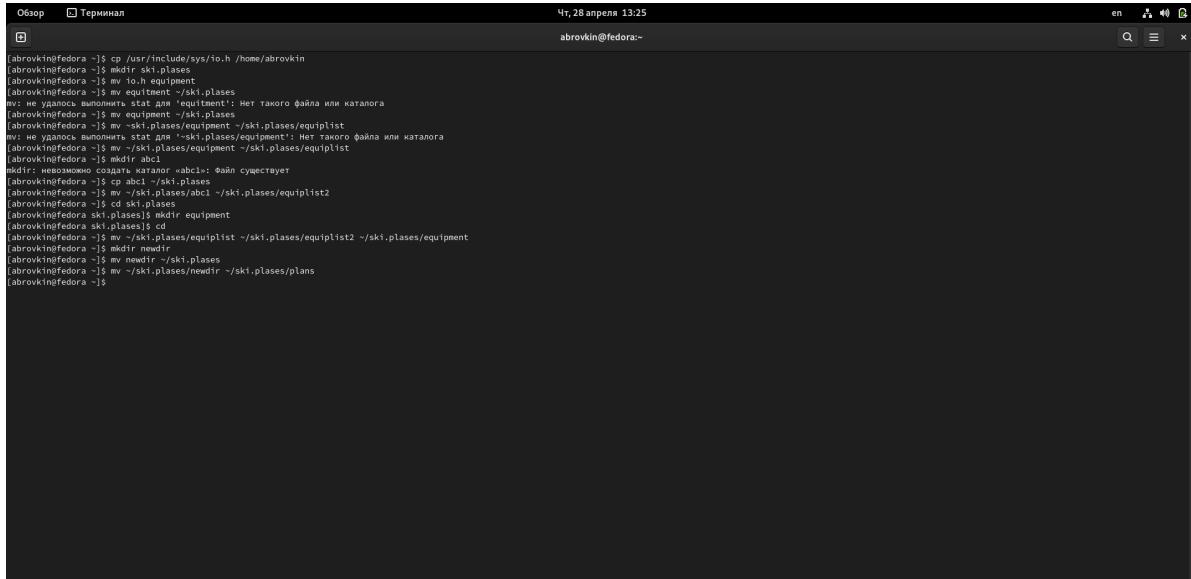
A screenshot of a terminal window titled 'Терминал' (Terminal) with the command line 'abrovkin@fedora:~'. The window shows a series of Linux commands being run and their outputs. The commands include cp, mv, mkdir, and ls, demonstrating file operations and directory navigation.

Рис. 3.6: Продолжаю выполнять примеры

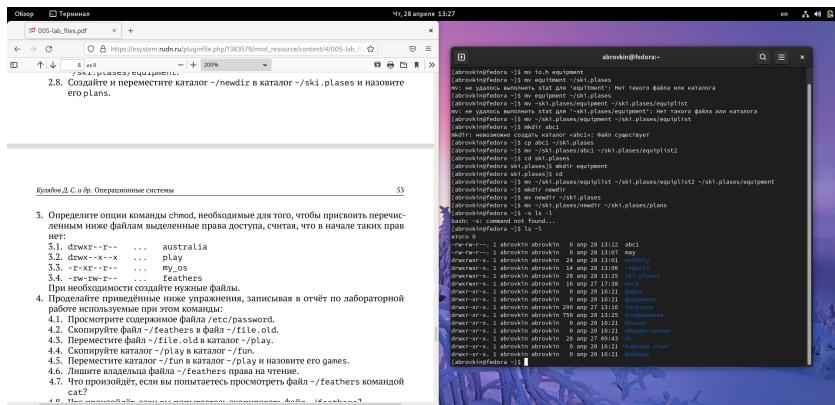


```
[abrovkin@fedora ~]$ cp /usr/include/sys/iio.h /home/abrovkin
[abrovkin@fedora ~]$ cd /home/abrovkin/ski.plases
[abrovkin@fedora ~]$ mv iio.h equipment
[abrovkin@fedora ~]$ ls
equipment
[abrovkin@fedora ~]$ mv equipment ~/ski.plases/equipment
[abrovkin@fedora ~]$ mv -v ~/ski.plases/equipment ~/ski.plases/equipment
[abrovkin@fedora ~]$ mv -v ~/ski.plases/equipment ~/ski.plases/equipment
[abrovkin@fedora ~]$ rm -rf ./ski.plases/equipment
[abrovkin@fedora ~]$ cd ..
[abrovkin@fedora ~]$ rm abc1
[abrovkin@fedora ~]$ rm abc1
[abrovkin@fedora ~]$ mv ~/ski.plases abc1 ~/ski.plases/equipment
[abrovkin@fedora ~]$ cd abc1
[abrovkin@fedora abc1]$ ls
[abrovkin@fedora abc1]$ cd ..
[abrovkin@fedora ~]$ cp ./ski.plases/equipment/eqiplist ~/ski.plases/equipment
[abrovkin@fedora ~]$ mv -v newdir ~/ski.plases
[abrovkin@fedora ~]$ rm -rf ./ski.plases/plans
[abrovkin@fedora ~]$
```

Рис. 3.7: Продолжаю выполнять примеры

3. Определил опции команды chmod, необходимые для того, чтобы присвоить перечисленным ниже файлам выделенные права доступа, считая, что в начале таких прав нет. При необходимости создал нужные файлы.(см.рис. 3.8)

### 3.1. drwxr--r... australia



The browser shows a task titled "Создайте и переместите каталог ~newdir в каталог ~/ski.plases и назовите его plans." Below it, the terminal window shows the user navigating to their home directory, creating a new directory named "plans", changing ownership to "australia", and setting permissions to "drwxr--r--". The terminal also lists other files in the directory.

```
[abrovkin@fedora ~]$ cd
[abrovkin@fedora ~]$ ls
[abrovkin@fedora ~]$ cd
[abrovkin@fedora ~]$ mv -v ./plans ~newdir
[abrovkin@fedora ~]$ cd
[abrovkin@fedora ~]$ mv ~newdir ~ski.plases
[abrovkin@fedora ~]$ cd
[abrovkin@fedora ~]$ rm -rf ~newdir
[abrovkin@fedora ~]$ rm -rf ~plans
[abrovkin@fedora ~]$ touch ~plans
[abrovkin@fedora ~]$ chmod 775 ~plans
[abrovkin@fedora ~]$ ls
[abrovkin@fedora ~]$ ls ~ski.plases
[abrovkin@fedora ~]$ rm -rf ~ski.plases
```

Рис. 3.8: australia

Обзор Терминал

Чт, 28 апреля 13:31  
abrovkin@fedora:~

```
abrovkin@fedora ~$ chmod a+r australis
(abrovkin@fedora ~$) chmod g-r australis
(abrovkin@fedora ~$) chmod o-x australis
(abrovkin@fedora ~$) ls -l
итого 0
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:12 abc1
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:38 australis
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:07 may
drwxrwxr-- 1 abrovkin abrovkin 24 апр 28 13:01 monthly
drwxrwxr-- 1 abrovkin abrovkin 14 апр 28 13:06 reports
drwxrwxr-- 1 abrovkin abrovkin 16 апр 28 13:01 places
drwxr--r-- 1 abrovkin abrovkin 16 апр 27 17:38 work
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 16:21 видео
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 16:21 документы
drwxr--r-- 1 abrovkin abrovkin 200 апр 27 13:12 загрузки
drwxr--r-- 1 abrovkin abrovkin 100 апр 27 13:12 изображения
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 16:21 музыка
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 16:21 общедоступные
drwxr--r--x 1 abrovkin abrovkin 28 апр 27 09:43 ос
drwxr--r--x 1 abrovkin abrovkin 0 апр 28 16:21 рабочий стол
drwxr--r--x 1 abrovkin abrovkin 0 апр 28 16:21 вебленки
(abrovkin@fedora ~$)
```

Обзор Терминал

Чт, 28 апреля 13:37  
abrovkin@fedora:~

005-lab\_files.pdf

https://esystem.rudn.ru/pluginfile.php/1383579/mod\_resource/content/4/005-lab\_1.pdf

Кулабов Д. С. и др. Операционные системы 55

3. Определите опции команды chmod, необходимые для того, чтобы присвоить перечисленным ниже файлам выделенные права доступа, считая, что в начале таких прав нет:

- 3.1. drwxr--r-- ... australis
- 3.2. drwx--x-x ... play
- 3.3. -r--x--r-- ... my\_os
- 3.4. -rw-rw-r-- ... feathers

При необходимости создайте нужные файлы.

4. Проделайте приведённые ниже упражнения, записывая в отчёт по лабораторной работе используемые при этом команды:

- 4.1. Просмотрите содержимое файла /etc/password.
- 4.2. Скопируйте файл ~/feathers в файл ~/file.old.
- 4.3. Переместите файл ~/file.old в каталог ~/play.
- 4.4. Скопируйте каталог ~/play в каталог ~/fun.
- 4.5. Переместите каталог ~/fun в каталог ~/play и назовите его games.
- 4.6. Лишите владельца файла ~/feathers права на чтение.
- 4.7. Что произойдёт, если вы попытаетесь просмотреть файл ~/feathers командой cat?
- 4.8. Что произойдёт, если вы попытаетесь скопировать файл ~/feathers?
- 4.9. Дайте владельцу файла ~/feathers право на чтение.
- 4.10. Лишите владельца каталога ~/play права на выполнение.
- 4.11. Переидите в каталог ~/play. Что произошло?
- 4.12. Дайте владельцу каталога ~/play право на выполнение.

abrovkin@fedora:~

```
abrovkin@fedora:~$ chmod go+r, g-w play
chmod: неверный режим: «go+r,»
Помощь команда «chmod --help» можно получить дополнительную информацию.
(abrovkin@fedora ~$) chmod go+r, g-w play
(abrovkin@fedora ~$) ls -l
итого 0
-rw-r--r-- 1 abrovkin abrovkin 0 апр 28 13:12 abc1
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:38 australis
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:07 may
drwxrwxr-- 1 abrovkin abrovkin 24 апр 28 13:01 monthly
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:34 my_os
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:31 play
drwxrwxr-- 1 abrovkin abrovkin 14 апр 28 13:06 reports
drwxrwxr-- 1 abrovkin abrovkin 16 апр 28 13:01 places
drwxr--r-- 1 abrovkin abrovkin 16 апр 27 17:38 work
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 16:21 видео
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 16:21 документы
drwxr--r-- 1 abrovkin abrovkin 200 апр 27 13:12 загрузки
drwxr--r-- 1 abrovkin abrovkin 100 апр 27 13:12 изображения
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 16:21 музыка
drwxr--r--x 1 abrovkin abrovkin 0 апр 28 16:21 общедоступные
drwxr--r--x 1 abrovkin abrovkin 28 апр 27 09:43 ос
drwxr--r--x 1 abrovkin abrovkin 0 апр 28 16:21 рабочий стол
drwxr--r--x 1 abrovkin abrovkin 0 апр 28 16:21 вебленки
(abrovkin@fedora ~$)
```

### 3.2. drwxr-x-x ... play

```

Чт, 28 апреля 13:34
abrovkin@fedora:~>

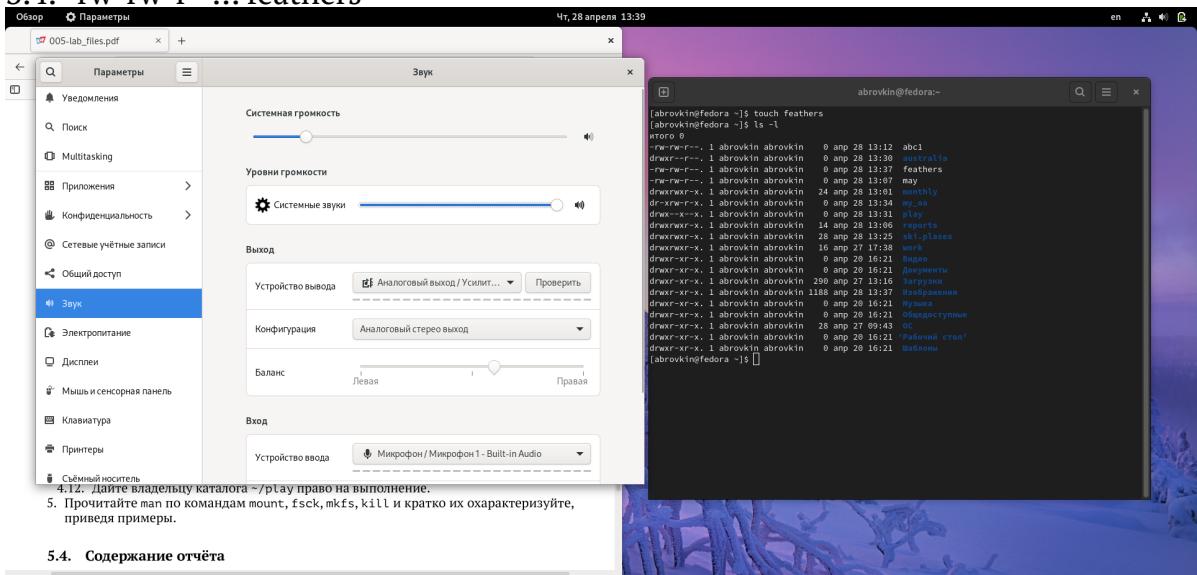
[abrovkin@fedora ~]$ ls -l
total 0
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:12 abc1
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:30 australia
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:07 may
drwxr--r-- 1 abrovkin abrovkin 24 апр 28 13:01 monthly
drwxr--r-- 1 abrovkin abrovkin 14 апр 28 13:06 reports
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:25 skt_plases
drwxr--r-- 1 abrovkin abrovkin 16 апр 27 17:38 work
drwxr--r-- 1 abrovkin abrovkin 0 апр 20 16:21 видео
drwxr--r-- 1 abrovkin abrovkin 290 апр 27 13:37 загрузки
drwxr--r-- 1 abrovkin abrovkin 0 апр 27 13:37 изображения
drwxr--r-- 1 abrovkin abrovkin 0 апр 20 16:21 музыка
drwxr--r-- 1 abrovkin abrovkin 0 апр 20 16:21 общедоступные
drwxr--r-- 1 abrovkin abrovkin 28 апр 27 09:43 ос
drwxr--r-- 1 abrovkin abrovkin 0 апр 20 16:21 рабочий стол
drwxr--r-- 1 abrovkin abrovkin 0 апр 20 16:21 вебблони
[abrovkin@fedora ~]$ ls -l
total 0
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:12 abc1
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:30 australia
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:07 may
drwxr--r-- 1 abrovkin abrovkin 24 апр 28 13:01 monthly
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:31 play
drwxr--r-- 1 abrovkin abrovkin 14 апр 28 13:06 reports
drwxr--r-- 1 abrovkin abrovkin 0 апр 28 13:25 skt_plases
drwxr--r-- 1 abrovkin abrovkin 16 апр 27 17:38 work
drwxr--r-- 1 abrovkin abrovkin 0 апр 20 16:21 видео
drwxr--r-- 1 abrovkin abrovkin 290 апр 27 13:37 загрузки
drwxr--r-- 1 abrovkin abrovkin 972 апр 28 13:31 изображения
drwxr--r-- 1 abrovkin abrovkin 0 апр 20 16:21 музыка
drwxr--r-- 1 abrovkin abrovkin 0 апр 20 16:21 общедоступные
drwxr--r-- 1 abrovkin abrovkin 28 апр 27 09:43 ос
drwxr--r-- 1 abrovkin abrovkin 0 апр 20 16:21 рабочий стол
drwxr--r-- 1 abrovkin abrovkin 0 апр 20 16:21 вебблони
[abrovkin@fedora ~]$ chmod go+r, g+w play
chmod: неверный режим: «go+r»
По команде «chmod --help» можно получить дополнительную информацию.
[abrovkin@fedora ~]$

```

### 3.3. -r-xr-r- ... my\_os

С помощью команды ls показало, все что нам необходимо, просто почему-то скриншот не записался.

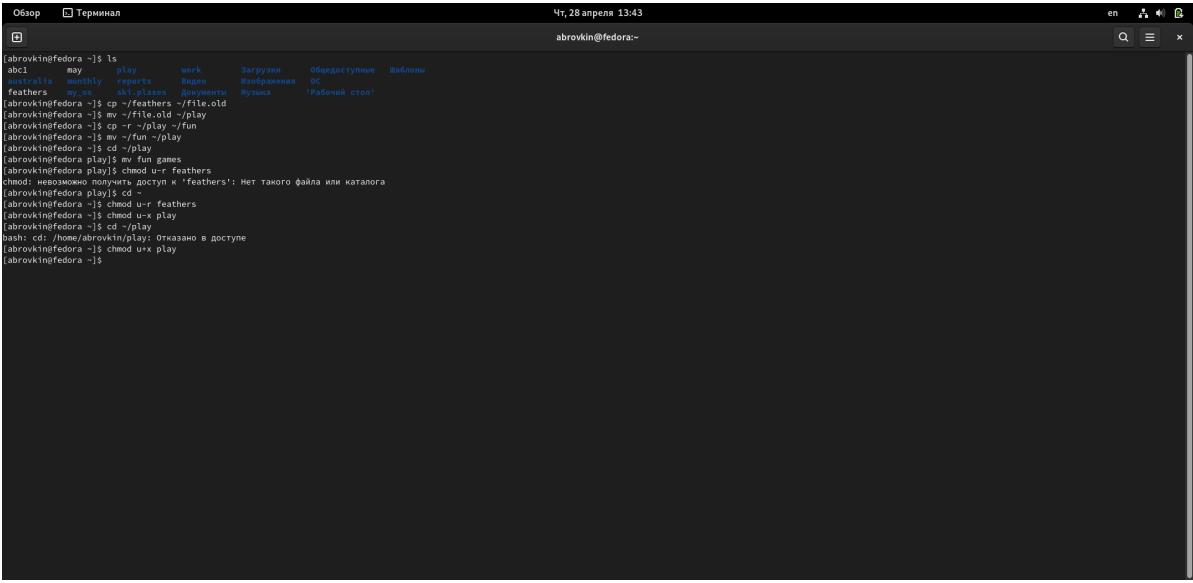
### 3.4. -rw-rw-r- ... feathers



4. Проделал приведённые ниже упражнения, записывая в отчёт по лабораторной работе используемые при этом команды:

- 4.1. Не просмотрел содержимое файла /etc/password, так как у меня его нет.
- 4.2. Скопировал файл ~/feathers в файл ~/file.old командой cp.

- 4.3. Переместил файл ~/file.old в каталог ~/play командой mv.
- 4.4. Скопировал каталог ~/play в каталог ~/fun командой cp -r.
- 4.5. Переместил каталог ~/fun в каталог ~/play командой mv и назвал его games командой mv.
- 4.6. Лишил владельца файла ~/feathers права на чтение командой chmod u-r.
- 4.7. Если попытаться скопировать файл ~/feathers командой cp, то выведется:
- 4.8. Дал владельцу файла ~/feathers право на чтение командой chmod u+r.
- 4.9. Лишил владельца каталога ~/play права на выполнение командой chmod u-x.
- 4.10. Попытался перейти в каталог ~/play командой cd.
- 4.11. Дал владельцу каталога ~/play право на выполнение командой chmod u+x.



```
Обзор Терминал Чт, 28 апреля 13:43
abrovkin@fedora:~$ ls
abrovkin  may play    work    Загрузки   Общедоступные   Шаблоны
australia monthly reports  Видео   Изображения   ОС
feathers my_as skypeas  Документы  Музыка   'Рабочий стол'
abrovkin@fedora:~$ cp ./feathers ./file.old
abrovkin@fedora:~$ cp -r ./file.old ./play
abrovkin@fedora:~$ cp -r ./play ./fun
abrovkin@fedora:~$ mv ./fun ./play
abrovkin@fedora:~$ cd ./play
abrovkin@fedora:~/play$ rm ./feathers
abrovkin@fedora:~/play$ chmod u+r feathers
chmod: невозможно получить доступ к 'feathers': Нет такого файла или каталога
(abrovkin@fedora play)$ cd ..
abrovkin@fedora:~$ chmod u-r feathers
abrovkin@fedora:~$ cp ./play ./play
abrovkin@fedora:~$ cd ./play
bash: cd: /home/abrovkin/play: Отказано в доступе
(abrovkin@fedora:~/play$ chmod u+x play
abrovkin@fedora:~$
```

Рис. 3.9: На скриншоте все ответы на данные пункты

5. Прочитал man по командам mount, fsck, mkfs, kill.

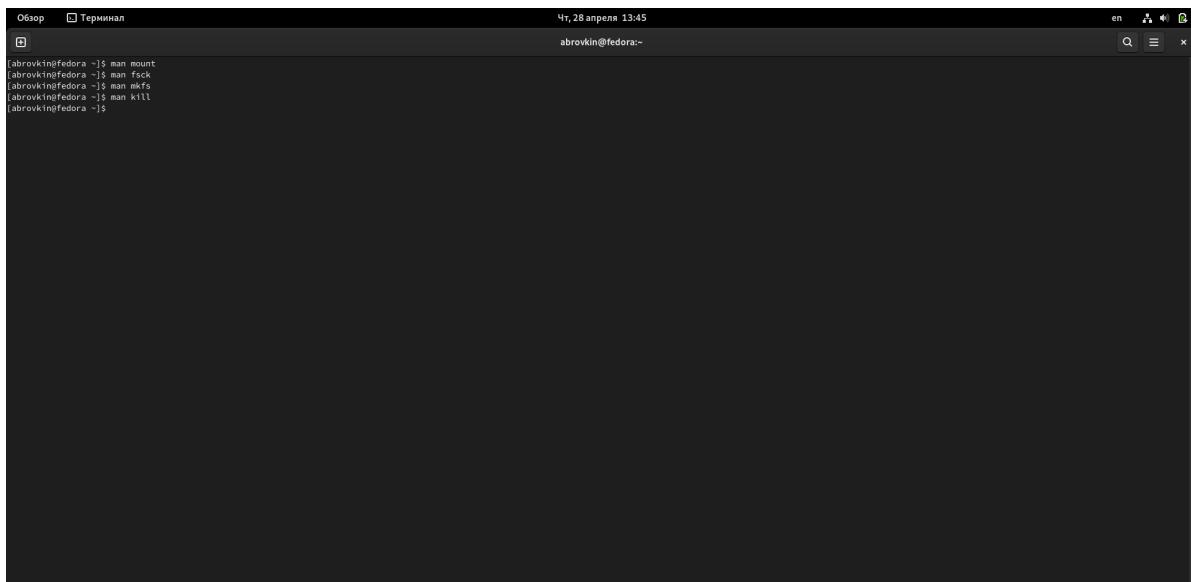


Рис. 3.10: команда man

## mount

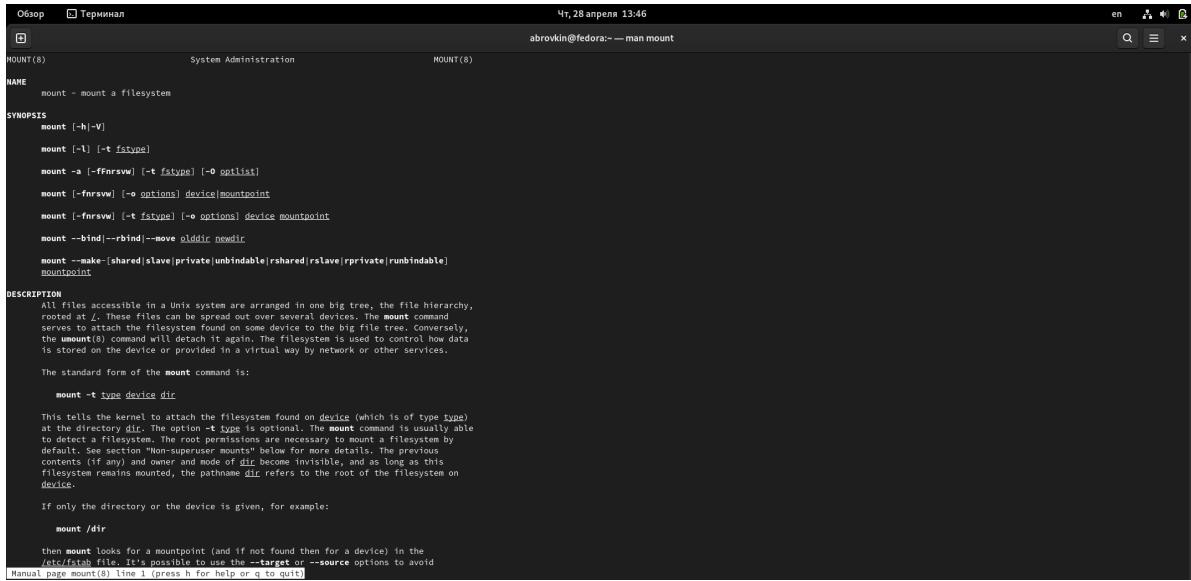


Рис. 3.11: mount

```

Обзор Терминал 4r, 28 апреля 13:46
abrovkin@fedora:— man mount

If only the directory or the device is given, for example:
mount /dir

then mount looks for a mountpoint (and if not found then for a device) in the
/etc/fstab file. It's possible to use the --target or --source options to avoid
ambiguous interpretation of the given argument. For example:
mount --target /mountpoint

The same filesystem may be mounted more than once, and in some cases (e.g., network
filesystem) the same filesystem may be mounted on the same mountpoint multiple times.
The mount command does not implement any policy to control this behavior. All behavior
is controlled by the kernel and it is usually specific to the filesystem driver. The
exception is --all, in this case already mounted filesystems are ignored (see --all
below for more details).

Listing the mounts
The listing mode is maintained for backward compatibility only.

For more robust and customizable output use findmnt(8), especially in your scripts.
Note that control characters in the mountpoint name are replaced with '?'.

The following command lists all mounted filesystems (of type type):
mount [-l] [-t type]

The option -l adds labels to this listing. See below.

Indicating the device and filesystem
Most devices are indicated by a filename (of a block special device), like /dev/sda1,
but there are other possibilities. For example, in the case of an NFS mount, device may
look like /nfs/nih.lnt/dirc.

The device names of disk partitions are unstable; hardware reconfiguration, adding
or removing drives can cause changes in names. This is the reason why it's strongly
recommended to use filesystem or partition identifiers like UUID or LABEL. Currently
supported identifiers (tags):

LABEL=label
Human readable filesystem identifier. See also -L.

UUID=uuid
Filesystem universally unique identifier. The format of the UUID is usually a
series of hex digits separated by hyphens. See also -U.

Manual page mount(8) line 41 (press h for help or q to quit)

```

```

Обзор Терминал 4r, 28 апреля 13:46
abrovkin@fedora:— man mount

Note that mount uses UUIDs as strings. The UUIDs from the command line or from
fstab(5) are not converted to internal binary representation. The string
representation of the UUID should be based on lower case characters.

PARTLABEL=label
Human readable partition identifier. This identifier is independent on filesystem
and does not change by mkfs or mkswap operations. It's supported for example for
GUID Partition Tables (GPT).

PARTUUID=uuid
Partition universally unique identifier. This identifier is independent on
filesystem and does not change by mkfs or mkswap operations. It's supported for
example for GUID Partition Tables (GPT).

ID=id
Hardware block device ID as generated by udev. This identifier is usually based on
MAC (unique storage identifier) and assigned by the hardware manufacturer. See ls
/dev/disk/by-id for more details, this directory and running udevd is required.
This identifier is not recommended for generic use as the identifier is not
strictly defined and it depends on udev, udev rules and hardware.

The command blkid --fs provides an overview of filesystems, LABELs and UUIDs on
available block devices. The command blkid -p <device> provides details about a
filesystem on the specified device.

Don't forget that there is no guarantee that UUIDs and labels are really unique,
especially if you move, share or copy the device. Use blkid -o +UUID,PARTUUID to verify
that the UUIDs are really unique in your system.

The recommended setup is to use tags (e.g. UUID=uuid) rather than
/dev/disk/by-label/label_id or PARTLABEL=partition_label. They symbolic links in the /etc/fstab file.
Tags are more readable, robust and portable. The mount(8) command internally uses udev
symbolic links, so the use of symbolic links in /etc/fstab has no advantage over tags. For more
details see libblkid(3).

The proc filesystem is not associated with a special device, and when mounting it, an
arbitrary keyword – for example, proc – can be used instead of a device specification.
(The customary choice none is less fortunate: the error message 'none already mounted'
from mount can be confusing.)

The files /etc/fstab, /etc/mtab and /proc/mounts
The file /etc/fstab (see fstab(5)), may contain lines describing what devices are
usually mounted where, using which options. The default location of the fstab(5) file
can be overridden with the --fstab path command-line option (see below for more
details).

Manual page mount(8) line 86 (press h for help or q to quit)

```

```

Обзор Терминал 4r, 28 апреля 13:46
abrovkin@fedora:— man mount

-F, --fork
  (Used in conjunction with -a.) Fork off a new incarnation of mount for each device.
  This will do the mounts on different devices or different NFS servers in parallel.
  This has the advantage that it is faster; also NFS timeouts proceed in parallel. A
  disadvantage is that the order of the mount operations is undefined. Thus, you
  cannot use this option if you want to mount both /usr and /usr/spool.

-f, --fake
  Causes everything to be done except for the actual system call; if it's not
  obvious, this is taken to mean mounting the filesystem. This is useful in conjunction
  with script to determine what the command is trying to do. It can also
  be used to add entries for devices that were mounted earlier with the -n option.
  The -f option checks for an existing record in /etc/mtab and fails when the record
  already exists (with a regular non-fake mount, this check is done by the kernel).

-i, --internal-only
  Don't call the /sbin/mount.filesystem helper even if it exists.

-l, --label label
  Mount the partition that has the specified label.

-l, --show-labels
  Add the labels in the mount output. mount must have permission to read the disk
  device (e.g. be set-user-ID root) for this to work. One can set such a label for
  ext2, ext3 or ext4 using the xlabel(8) utility, or for XFS using xfs_admin(8), or
  for reiserfs using reiserfstune(8).

-M, --move
  Move a subtree to some other place. See above, the subsection The move operation.

-n, --no-mtab
  Mount without writing in /etc/mtab. This is necessary for example when /etc is on a
  read-only filesystem.

-N, --namespace ns
  Perform the mount operation in the mount namespace specified by ns. ns is either
  PID of process running in that namespace or special file representing that
  namespace.

  mount switches to the mount namespace when it reads /etc/fstab, writes /etc/mtab
  (or writes to /run/mount) and calls the mount(2) system call, otherwise it runs in
  the original mount namespace. This means that the target namespace does not have to
  contain any libraries or other requirements necessary to execute the mount(2) call.

  See mount_namespaces(7) for more information.

Manual page mount(8) line 306 (press h for help or q to quit)
Обзор Терминал 4r, 28 апреля 13:46
abrovkin@fedora:— man mount

mount LABEL=mydisk -o noatime,noexec,nosuid
For more details, see the FILESYSTEM-INDEPENDENT MOUNT OPTIONS and
FILESYSTEM-SPECIFIC MOUNT OPTIONS sections.

--options-mode mode
Controls how to combine options from fstab/mtab with options from the command line.
mode can be one of ignore, append, prepend or replace. For example, append means
that options from fstab are appended to options from the command line. The default
value is prepend – it means command line options are evaluated after fstab options.
Note that the last option wins if there are conflicting ones.

--options-source source
Source of default options. source is a comma-separated list of fstab, mtab and
disable. disable disables fstab and mtab and disables --options-source-force. The
default value is fstab,mtab.

--options-source-force
Use options from fstab/mtab even if both device and dir are specified.

-R, --rbind
Remount a subtree and all possible submounts somewhere else (so that its contents
are available in both places). See above, the subsection Bind mounts.

-r, --read-only
Mount the filesystem read-only. A synonym is --ro.

Note that, depending on the filesystem type, state and kernel behavior, the system
may still write to the device. For example, ext3 and ext4 will replay the journal
if the filesystem is dirty. To prevent this kind of write access, you may want to
mount an ext3 or ext4 filesystem with the ro,noauto mount options or set the block
device itself to read-only mode, see the blockdev(8) command.

-s
Tolerate sloppy mount options rather than failing. This will ignore mount options
not supported by a filesystem type. Not all filesystems support this option.
Currently it's supported by the mount.info mount helper only.

--source device
If only one argument for the mount command is given, then the argument might be
interpreted as the target (mountpoint) or source (device). This option allows you
to explicitly define that the argument is the mount source.

--target directory
If only one argument for the mount command is given, then the argument might be
interpreted as the target (mountpoint) or source (device). This option allows you

Manual page mount(8) line 430 (press h for help or q to quit)

```

```

47, 28 апреля 13:46
abrovkin@fedora: ~ man mount

BUGS
It is possible for a corrupted filesystem to cause a crash.
Some Linux filesystems don't support -o sync and -o dirsync (the ext2, ext3, ext4, fat and vfat filesystems do support synchronous updates (à la BSD) when mounted with the sync option).
The -o remount may not be able to change mount parameters (all ext2fs-specific parameters, except sb, are changeable with a remount, for example, but you can't change gid or umask for the fatfs).
It is possible that the files /etc/mtab and /proc/mounts don't match on systems with a regular stab file. The first file is based only on the mount command options, but the content of the second file also depends on the kernel and others settings (e.g. on a remote server). In certain cases the mount command may report unreliable information (an NFS mount point and the /proc/mount file usually contains more reliable information.) This is another reason to replace the stab file with a symlink to the /proc/mounts file.
Checking files on NFS filesystems referenced by file descriptors, (i.e. the fentl and fextl families of functions) may lead to inconsistent results due to the lack of a consistency check in the kernel even if the nosync mount option is used.
The loop option with the offset or sizelimit options used may fail when using older kernels if the mount command can't confirm that the size of the block device has been configured as requested. This situation can be worked around by using the losetup(8) command manually before calling mount with the configured loop device.

AUTHORS
Karel Zak <kzak@redhat.com>

SEE ALSO
mount(2), umount(2), filesystems(5), fstab(5), nfs(5), xfs(5), mount_namespaces(7), xattr(7), ezlabel(8), findmnt(8), losetup(8), lsblk(8), mke2fs(8), mountd(8), nfssd(8), swapon(8), tune2fs(8), umount(8), xfs_admin(8)

REPORTING BUGS
For bug reports, use the issue tracker at
https://github.com/karelzak/util-linux/issues.

AVAILABILITY
The mount command is part of the util-linux package which can be downloaded from Linux Kernel Archive <https://www.kernel.org/pub/linux/utils/util-linux/>.

util-linux_2.37.2 2021-09-16 MOUNT(8)
Manual page mount(8) line 2045/2090 (END) [press h for help or q to quit]

```

## fsck

```

47, 28 апреля 13:46
abrovkin@fedora: ~ man fsck

FSCK(8)                                         System Administration                                         FSCK(8)

NAME
   fsck - check and repair a Linux filesystem

SYNOPSIS
   fsck [-ASVRTHM] [-f [fd]] [-C [fd]] [-t fstype] [filesystem...] [--] [fs=specific-options]

DESCRIPTION
   fsck is used to check and optionally repair one or more Linux filesystems. filesystem can be a device name (e.g., /dev/hdc1, /dev/sdb2), a mount point (e.g., /, /usr, /home), or an filesystem label or UUID specifier (e.g., UUID=8686abf6-88c5-4a83-98b8-bfc240577bd or LABEL=root). Normally, the fsck program will try to handle filesystems on different physical disk drives in parallel to reduce the total amount of time needed to check all of them. If no filesystems are specified on the command line, and the -A option is not specified, fsck will default to checking filesystems in /etc/fstab serially. This is equivalent to the -As options.

   The exit status returned by fsck is the sum of the following conditions:
   0      No errors
   1      Filesystem errors corrected
   2      System should be rebooted
   4      Filesystem errors left uncorrected
   8      Operational error
   16     Usage or syntax error
   32     Checking canceled by user request
   128    Shared-library error

   The exit status returned when multiple filesystems are checked is the bit-wise OR of the exit statuses for each filesystem that is checked.

   In actuality, fsck is simply a front-end for the various filesystem checkers (fsck.fstype) available under Linux. The filesystem-specific checker is searched for in the PATH environment variable. If the PATH is undefined then fallback to /sbin.
   Please see the filesystem-specific checker manual pages for further details.

Manual page fsck(8) line 1 (press h for help or q to quit)

```

Рис. 3.12: fsck

```

Обзор Терминал 4r, 28 апнена 13:46
abrovkin@fedora:-- man fsck
prefixed by a negation operator, then only those listed filesystems will be checked.

Options specifiers may be included in the comma-separated fslist. They must have the format opts=fs:option. If an options specifier is present, then only filesystems which contain fs:option in their mount options field of /etc/fstab will be checked. If the options specifier is prefixed by a negation operator, then only those filesystems that do not have fs:option in their mount options field of /etc/fstab will be checked.

For example, if opts+ro appears in fslist, then only filesystems listed in /etc/fstab with the ro option will be checked.

For compatibility with Mandrake distributions whose boot scripts depend upon an unauthorized UI change to the fsck program, if a filesystem type of loop is found in fslist, it is treated as if opts=loop were specified as an argument to the -t option.

Normally, the filesystem type is deduced by searching for filesys in the /etc/fstab file and using the corresponding entry. If the type cannot be deduced, and there is only a single filesystem given as an argument to the -t option, fsck will use the specified filesystem type. If this type is not available, then the default filesystem type (currently ext2) is used.

-A
Walk through the /etc/fstab file and try to check all filesystems in one run. This option is typically used from the /etc/rc system initialization file, instead of multiple commands for checking a single filesystem.

The root filesystem will be checked first unless the -R option is specified (see below). After that, filesystems will be checked in the order specified by the fs_passno (the sixth) field in the /etc/fstab file. Filesystems with a fs_passno value of 0 are skipped and are not checked at all. Filesystems with a fs_passno value of greater than zero will be checked in order, with filesystems with the lowest fs_passno number being checked first. If there are multiple filesystems with the same pass number, fsck will attempt to check them in parallel, although it will avoid running multiple filesystem checks on the same physical disk.

fsck does not check stacked devices (RAIDS, dm-crypt, ... ) in parallel with any other device. See below for FSCK_FORCE_ALL_PARALLEL setting. The /sys filesystem is used to determine dependencies between devices.

Hence, a very common configuration in /etc/fstab files is to set the root filesystem to have a fs_passno value of 1 and to set all other filesystems to have a fs_passno value of 2. This will allow fsck to automatically run filesystem checkers in parallel if it is advantageous to do so. System administrators might choose not to use this configuration if they need to avoid multiple filesystem checks running in parallel for some reason - for example, if the machine in question is short on memory so that excessive paging is a concern.

fsck normally does not check whether the device actually exists before calling a filesystem specific checker. Therefore non-existing devices may cause the system to enter filesystem repair mode during boot if the filesystem specific checker returns a fatal error. The /etc/fstab mount option nofail may be used to have fsck skip non-existing devices. fsck also skips non-existing devices that have the special filesystem type auto.

-C [fd]
Display completion/progress bars for those filesystem checkers (currently only for ext[234]) which support them. fsck will manage the filesystem checkers so that only one of them will display a progress bar at a time. GUI front-ends may specify a file descriptor fd, in which case the progress bar information will be sent to that file descriptor.

-M
Do not check mounted filesystems and return an exit status of 0 for mounted filesystems.

-N
Don't execute, just show what would be done.

-P
When the -A flag is set, check the root filesystem in parallel with the other filesystems. This is not the safest thing in the world to do, since if the root filesystem is in doubt things like the e2fsck(8) executable might be corrupted! This option is mainly provided for those sysadmins who don't want to repartition the root filesystem to be small and compact (which is really the right solution).

-R
When checking all filesystems with the -A flag, skip the root filesystem. (This is useful in case the root filesystem has already been mounted read-write.)

Manual page fsck(8) line 71 [press h for help or q to quit]

```

Обзор Терминал 4r, 28 апнена 13:46
abrovkin@fedora:-- man fsck

Options and arguments which follow the *--* are treated as filesystem-specific options to be passed to the filesystem-specific checker.

Please note that *fsck* is not designed to pass arbitrarily complicated options to filesystem-specific checkers. If you're doing something complicated, please just execute the filesystem-specific checker directly. If you pass *fsck* some horribly complicated options and arguments, and it doesn't do what you expect, don't bother reporting it as a bug. You're almost certainly doing something that you shouldn't be doing with *fsck*. Options to different filesystem-specific *fsck*'s are not standardized.

**ENVIRONMENT**

The *fsck* program's behavior is affected by the following environment variables:

- FSCK\_FORCE\_ALL\_PARALLEL**
- If this environment variable is set, *fsck* will attempt to check all of the specified filesystems in parallel, regardless of whether the filesystems appear to be on the same device. (This is useful for RAID systems or high-end storage systems such as those sold by companies such as IBM or EMC.) Note that the *fs\_passno* value is still used.
- FSCK\_MAX\_INST**
- This environment variable will limit the maximum number of filesystem checkers that can be running at one time. This allows configurations which have a large number of disks to avoid *fsck* starting too many filesystem checkers at once, which might overload CPU and memory resources available on the system. If this value is zero, then an unlimited number of processes can be spawned. This is currently the default, but future versions of *fsck* may attempt to automatically determine how many filesystem checks can be run based on gathering accounting data from the operating system.
- PATH**
- The *PATH* environment variable is used to find filesystem checkers.
- FSTAB\_FILE**
- This environment variable allows the system administrator to override the standard location of the */etc/fstab* file. It is also useful for developers who are testing *fsck*.
- LIBBLKID\_DEBUG=all**
- enables libblkid debug output.
- LIMOUNT\_DEBUG=all**
- enables libmount debug output.

**FILES**

*/etc/fstab*

**AUTHORS**

Theodore Ts'o <tytso@mit.edu>, Karel Zak <kzak@redhat.com>

**SEE ALSO**

*fstab(5)*, *mkfs(8)*, *fsck.ext2(8)* or *fsck.ext3(8)* or *e2fsck(8)*, *fsck.cramfs(8)*, *fsck.jfs(8)*, *fsck.nfs(8)*, *fsck.minix(8)*, *fsck.msdos(8)*, *fsck.vfat(8)*, *fsck.xfs(8)*, *reiserfsck(8)*

**REPORTING BUGS**

For bug reports, use the issue tracker at <https://github.com/karelzak/util-linux/issues>.

**AVAILABILITY**

The *fsck* command is part of the *util-linux* package which can be downloaded from <https://www.kernel.org/pub/linux/utils/util-linux/>.

util-linux 2.37.2 2021-07-20

Manual page fsck(8) line 134/179 (END) [press h for help or q to quit]

mkfs

mkfs (8)

**NAME** mkfs - build a Linux filesystem

**SYNOPSIS** `mkfs [options] [-t type] [fs-options] device [size]`

**DESCRIPTION** This mkfs frontend is deprecated in favor of filesystem specific `mkfs.<type>` utils.

`mkfs` is used to build a Linux Filesystem on a device, usually a hard disk partition. The `device` argument is either the device name (e.g., `/dev/hd1a`, `/dev/sdb2`), or a regular file that shall contain the filesystem. The `size` argument is the number of blocks to be used for the filesystem.

The exit status returned by `mkfs` is 0 on success and 1 on failure.

In actuality, `mkfs` is simply a front-end for the various filesystem builders (`mkfs.<fstype>`) available under Linux. The filesystem-specific builder is searched for via your `PATH` environment setting only. Please see the filesystem-specific builder manual pages for further details.

**OPTIONS**

- `-t, --type type`: Specify the `type` of filesystem to be built. If not specified, the default filesystem type (currently ext2) is used.
- `fs-options`: Filesystem-specific options to be passed to the real filesystem builder.
- `-V, --verbose`: Produce verbose output, including all filesystem-specific commands that are executed. Specifying this option more than once inhibits execution of any filesystem-specific commands. This is really only useful for testing.
- `-v, --version`: Display version information and exit. (Option `-V` will display version information only when it is the only parameter, otherwise it will work as `--verbose`.)
- `-h, --help`: Display help text and exit.

**BUGS** All generic options must precede and not be combined with filesystem-specific options. Some filesystem-specific programs do not automatically detect the device size and require the `size` parameter to be specified.

**AUTHORS** David Engel <david@ods.com>, Fred N. van Kempen <waltje@walt.nl.mugnet.org>, Ron Sommeling <sommel@sci.kun.nl>.

The manual page was shamelessly adapted from Remy Card's version for the ext2 filesystem.

**SEE ALSO** `fs(5)`, `badblocks(8)`, `fsck(8)`, `mkdosfs(8)`, `mke2fs(8)`, `mkfs.btrfs(8)`, `mkfs.ext2(8)`, `mkfs.ext3(8)`, `mkfs.ext4(8)`, `mkfs.minix(8)`, `mkfs.msdos(8)`, `mkfs.vfat(8)`, `mkfs.xfs(8)`

Manual page `mkfs(8)` line 1 (press h for help or q to quit)

Рис. 3.13: mkfs

kill

KILL(1)

**NAME** kill - terminate a process

**SYNOPSIS** `kill [-signal|-s signal|-p] [-q value] [-a] [--timeout milliseconds signal] [-- pid|name...`

`kill -l [number] |-4`

**DESCRIPTION** The command `kill` sends the specified `signal` to the specified processes or process groups.

If no signal is specified, the `TERM` signal is sent. The default action for this signal is to terminate the process. This signal should be used in preference to the `KILL` signal (number 9), since a process may install a handler for the `TERM` signal in order to perform clean-up steps before terminating. If a process does not terminate after a `TERM` signal has been sent, then the `KILL` signal may be used; be aware that the latter signal cannot be caught, and so does not give the target process the opportunity to perform any clean-up before terminating.

Most modern shells have a builtin `kill` command, with a usage rather similar to that of the command described here. The `--all`, `--pid`, and `--queue` options, and the possibility to specify processes by command name, are local extensions.

If `signal` is 0, then no actual signal is sent, but error checking is still performed.

**ARGUMENTS** The list of processes to be signaled can be a mixture of names and PIDs.

`pid`: Each `pid` can be expressed in one of the following ways:

- `0`: where `g` is larger than 0. The process with PID `g` is signaled.
- `0`: All processes in the current process group are signaled.
- `-1`: All processes with a PID larger than 1 are signaled.
- `-n`: where `g` is larger than 1. All processes in process group `g` are signaled. When an argument of the form '`-n`' is given, and it is meant to denote a process group, either a signal must be specified first, or the argument must be preceded by a '`-`' option, otherwise it will be taken as the signal to send.
- `name`: All processes invoked using this `name` will be signaled.

**OPTIONS**

- `-s, --signal signal`

Manual page `kill(1)` line 1 (press h for help or q to quit)

Рис. 3.14: kill

```

Обзор Терминал 4r, 28 апреля 13:47
abrovkin@fedora:~— man kill

      name
          All processes invoked using this name will be signaled.

OPTIONS
  -s, --signal signal
      The signal to send. It may be given as a name or a number.

  -l, --list [number]
      Print a list of signal names, or convert the given signal number to a name. The signals can be found in /usr/include/linux/signal.h.

  -t, --table
      Similar to -l, but it will print signal names and their corresponding numbers.

  -a, --all
      Do not restrict the command-name-to-PID conversion to processes with the same UID as the present process.

  -p, --pid
      Only print the process ID (PID) of the named processes, do not send any signals.

  --verbose
      Print PID(s) that will be signaled with kill along with the signal.

  -q, --queue value
      Send a signal using sigqueue(3) rather than kill(2). The value argument is an integer that is sent along with the signal. If the receiving process has installed a handler for this signal using the SA_SIGINFO flag to sigaction(2), then it can obtain this data via the si_value field of the siginfo_t structure.

  --timeout milliseconds signal
      Send a signal defined in the usual way to a process, followed by an additional signal after a specified delay. The --timeout option causes kill to wait for a period defined in milliseconds before sending a follow-up signal to the process. This feature is implemented using the Linux kernel PID file descriptor feature in order to guarantee that the follow-up signal is sent to the same process or not sent if the process no longer exists.

      Note that the operating system may re-use PIDs and implementing an equivalent feature in a shell using kill and sleep would be subject to races whereby the follow-up signal might be sent to a different process that used a recycled PID.

      The --timeout option can be specified multiple times: the signals are sent sequentially with the specified timeouts. The --timeout option can be combined with the --queue option.

      As an example, the following command sends the signals QUIT, TERM and KILL in sequence and waits for 1000 milliseconds between sending the signals:
      kill --verbose --timeout 1000 TERM --timeout 1000 KILL \
          --signal QUIT 12345

EXIT STATUS
  kill has the following exit status values:

  0
  Manual page kill(1) line 41 (press h for help or q to quit)

  Note that the operating system may re-use PIDs and implementing an equivalent feature in a shell using kill and sleep would be subject to races whereby the follow-up signal might be sent to a different process that used a recycled PID.

  The --timeout option can be specified multiple times: the signals are sent sequentially with the specified timeouts. The --timeout option can be combined with the --queue option.

  As an example, the following command sends the signals QUIT, TERM and KILL in sequence and waits for 1000 milliseconds between sending the signals:
  kill --verbose --timeout 1000 TERM --timeout 1000 KILL \
      --signal QUIT 12345

EXIT STATUS
  kill has the following exit status values:

  0
      success
  1
      failure
  64
      partial success (when more than one process specified)

NOTES
  Although it is possible to specify the TID (thread ID, see gettid(2)) of one of the threads in a multithreaded process as the argument of kill, the signal is nevertheless directed to the process (i.e., the entire thread group). In other words, it is not possible to send a signal to an explicitly selected thread in a multithreaded process. The signal will be delivered to an arbitrarily selected thread in the target process that is not blocking the signal. For more details, see signal(7) and the description of CLONE_THREAD in clone(2).

  Various shells provide a builtin KILL command that is preferred in relation to the kill(1) executable described by this manual. The easiest way to ensure one is executing the command described in this page is to use the full path when calling the command, for example: /bin/kill --version

AUTHORS
  Salvatore Valente <svalente@mit.edu>, Karel Zak <kzak@redhat.com>

  The original version was taken from BSD 4.4.

SEE ALSO
  bash(1), tcsh(1), sigaction(2), kill(2), sigqueue(3), signal(7)

REPORTING BUGS
  For bug reports, use the issue tracker at https://github.com/karelzak/util-linux/issues.

AVAILABILITY
  The kill command is part of the util-linux package which can be downloaded from Linux Kernel Archive https://www.kernel.org/pub/linux/utils/util-linux/.
util-linux 2.37.2
Manual page kill(1) line 72/117 (END) (press h for help or q to quit)
2021-06-02
KILL(1)

```

**Краткая характеристика:** - mount применяется для монтирования файловых систем. - fsck восстанавливает повреждённую файловую систему или проверяет на целостность. - mkfs создаёт новую файловую систему. - kill используется для принудительного завершения работы приложений.

## **4 Выводы**

Ознакомился с файловой системой Linux, её структурой, именами и содержанием каталогов. Приобрел практические навыки по применению команд для работы с файлами и каталогами, по управлению процессами (и работами), по проверке использования диска и обслуживанию файловой системы.

## 5 Ответы на контрольные вопросы:

1.Характеристика файловой системы, которая использовалась в данной лабораторной работе: Файлы: abc1, april, may, june, july, isdv4.h, equipment, equiplist, equiplist2, my\_os, feathers, file.old. Каталоги: monthly, monthly.00, tmp, monthly.01, reports, usr, ski.plases, equipment, newdir, plans, australia, play, etc, fun, games.

2.Пример общей структуры файловой системы: /home/pdarzhankina/monthly/april, где /home/pdarzhankina – домашний каталог, /monthly – каталог, находящийся в домашнем и содержащий файл, /april – файл, находящийся в каталоге.

3. Чтобы содержимое некоторой файловой системы было доступно операционной системе должно быть выполнено монтирование тома.
4. Основные причины нарушения целостности файловой системы:

- Один блок адресуется несколькими mode (принадлежит нескольким файлам).
- Блок помечен как свободный, но в то же время занят (на него ссылается onode).
- Блок помечен как занятый, но в то же время свободен (ни один inode на него не ссылается).
- Неправильное число ссылок в inode (недостаток или избыток ссылающихся записей в каталогах).
- Несовпадение между размером файла и суммарным размером адресуемых inode блоков.

- Недопустимые адресуемые блоки (например, расположенные за пределами файловой системы).
- “Потерянные” файлы (правильные inode, на которые не ссылаются записи каталогов).
- Недопустимые или неразмещенные номера inode в записях каталогов. Чтобы устранить повреждения файловой системы используется команда fsck.

5. Команда mkfs создаёт новую файловую систему.

6. Характеристика команд, которые позволяют просмотреть текстовые файлы:

- для просмотра небольших файлов удобно пользоваться командой cat.
- для просмотра больших файлов используйте команду less — она позволяет осуществлять постраничный просмотр файлов.
- для просмотра начала файла можно воспользоваться командой head, по умолчанию она выводит первые 10 строк файла.
- команда tail выводит несколько (по умолчанию 10) последних строк файла.

7. Основные возможности команды cp:

- копирование файла в текущем каталоге.
- копирование нескольких файлов в каталог.
- копирование файлов в произвольном каталоге. Опция i в команде cp выведет на экран запрос подтверждения о перезаписи файла, если на место целевого файла вы поставите имя уже существующего файла. Команда cp с опцией r (recursive) позволяет копировать каталоги вместе с входящими в них файлами и каталогами.

8. Характеристика команд перемещения и переименования файлов и каталогов:

- переименование файлов в текущем каталоге. mv

- перемещение файлов в другой каталог. mv Если необходим запрос подтверждения о перезаписи файла, то нужно использовать опцию i.
  - переименование каталогов в текущем каталоге. mv
  - перемещение каталога в другой каталог. mv
  - переименование каталога, не являющегося текущим. mv <каталог/новое\_название\_каталога>
9. Каждый файл или каталог имеет права доступа: чтение (разрешены просмотр и копирование файла, разрешён просмотр списка входящих в каталог файлов), запись (разрешены изменение и переименование файла, разрешены создание и удаление файлов каталога), выполнение (разрешено выполнение файла, разрешён доступ в каталог и есть возможность сделать его текущим). Они могут быть изменены командой chmod.

# **Список литературы**

1. [Лаб-05]