Лабораторная работа №5

Malkov Roman Sergeevich

30.04.2022

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.

Ход работы

Выполняем копирование файла в текущем каталоге. Для этого используем следующие команды:

touch abc1
cp abc1 april
cp abc1 may

Далее выполянем копирование нескольких файлов(may и april) в каталог(monthly) одной командой(тут я немного ошибся и скопировал их двумя отдельными командами, следует быть внимательнее):

mkdir monthly
cp april may monthly

Ход работы

Копирование файлов в произвольном каталоге:

cp monthly/may monthly/june

Копирование каталога в текущем каталоге:

mkdir monthly.00
cp -r monthly monthly.00

Копирование каталогов в произвольном каталоге:

cp -r monthly.00 /tmp

Переименование файлов в текущем каталоге. Изменить название файла april на july в домашнем каталоге:

mv april july

Перемещение файлов в другой каталог. Переместить файл july в каталог monthly.00:

mv july monthly.00
ls monthly.00

Переименование каталогов в текущем каталоге.
Переименовать каталог monthly.00 в monthly.01:

mv monthly.00 monthly.01

Перемещение каталога в другой каталог. Переместить каталог monthly.01в каталог reports:

mkdir reports
mv monthly.01 reports

Переименование каталога, не являющегося текущим.
Переименовать каталог reports/monthly.01 в reports/monthly:

mv reports/monthly.01 reports/monthly

Меняем права владельца файла ~/may, дав тем самым ему права на выполнение:

chmod u+x may

Лишаем владельца прав на выполнение:

chmod u-x may

Меняем права доступа monthly, запретив чтение для всех членов группы и всех остальных пользователей:

chmod g-r, o-r monthly

Меняем права доступа к файлу ~/abc1, предоставив право записи членам группы:

chmod g+w abc1

Копируем файл /usr/include/sys/io.h в домашний каталог и называем его equipment:

cp /usr/include/sys/io.h /home/rsmaljkov
mv io.h equipment

В домашнем каталоге создаем директорию ~/ski.plases:

mkdir ski.places

Перещаем файл equipment в каталог ~/ski.plases и переименовывыем в equiplist:

mv equipment ~/ski.places/equiplist

Создаем в домашнем каталоге файл abc1 и копируем его в каталог ~/ski.plases, называем его equiplist2:

cp abc1 ~/ski.places
mv ~/ski.places/equiplist2

Создаем каталог с именем equipment в каталоге ~/ski.plases:

mkdir ski.places/equipment

Перемещаем файлы ~/ski.plases/equiplist и equiplist2 в каталог ~/ski.plases/equipment:

mv ~/ski.places/equiplist ~/ski.places/equipment/equiplist
mv ~/ski.places/equiplist2 ~/ski.places/equipment/equiplist2

Создаем и переместите каталог ~/newdir в каталог ~/ski.plases и назовите его plans:

mkdir newdir

mv newdir ski.places/plans

Определяем опции команды chmod, необходимые для того, чтобы присвоить выделенные права доступа, предварительно создав эти файлы(Скриншоты 4 - 8):

mkdir australia
mkdir play
touch my_os
touch features

chmod 744 australia
chmod 711 play
chmod 544 my_os
chmod 654 features

Просматриваем содержимое файла /etc/password, затем копируем файл ~/features в файл ~/file.old

```
[rsmaljkov@rsmaljkov /]$ sudo -s
[sudo] password for rsmaljkov:
[root@rsmaljkov /]# ls etc/password
[root@rsmaljkov /]# cp home/rsmaljkov/features etc/password/file.old
[root@rsmaljkov /]# ls etc/password
file.old
[root@rsmaljkov /]#
```

Пермещаем ~/file.old в каталог ~/play

```
[root@rsmaljkov /]# mv etc/password/file.old home/rsmaljkov/play
```

После этого, копируем каталог ~/play в каталог ~/fun, перемещаем каталог ~/fun в каталог ~/play и называем его games

```
[root@rsmaljkov /]# mkdir home/rsmaljkov/fun
[root@rsmaljkov /]# cp home/rsmaljkov/play home/rsmaljkov/fun
cp: -r not specified; omitting directory 'home/rsmaljkov/play'
[root@rsmaljkov /]# cp -r  home/rsmaljkov/play home/rsmaljkov/fun
```

```
[root@rsmaljkov /]# mv home/rsmaljkov/fun home/rsmaljkov/play/games
```

Далее мы лишаем владельца файла ~/features права на чтение, пытыемся открыть и скопировать файл, и смотрим на результат, после этого возвращаем право на чтение.

```
[root@rsmaljkov rsmaljkov]# chmod u-r features
[root@rsmaljkov rsmaljkov]#
```

```
[root@rsmaljkov rsmaljkov]# chmod u-r features
[root@rsmaljkov rsmaljkov]# ls -l features
--wxrwxrwx. 1 rsmaljkov rsmaljkov 0 May  4 12:33 features
[root@rsmaljkov rsmaljkov]#
```

Рис. 1: Screenshot_17

```
[rsmaljkov@rsmaljkov ~]$ cat features
cat: features: Permission denied
[rsmaljkov@rsmaljkov ~]$ cp features etc
cp: cannot open 'features' for reading: Permission denied
[rsmaljkov@rsmaljkov ~]$
```

Затем лишаем владельца права на выполнение каталога ~/play, переходим в каталог, смотрим результат, а после возвращаем право на выполнение

```
[root@rsmaljkov rsmaljkov]# chmod u-x play
[root@rsmaljkov rsmaljkov]#
```

```
[rsmaljkov@rsmaljkov ~]$ cd play
bash: cd: play: Permission denied
[rsmaljkov@rsmaljkov ~]$
```

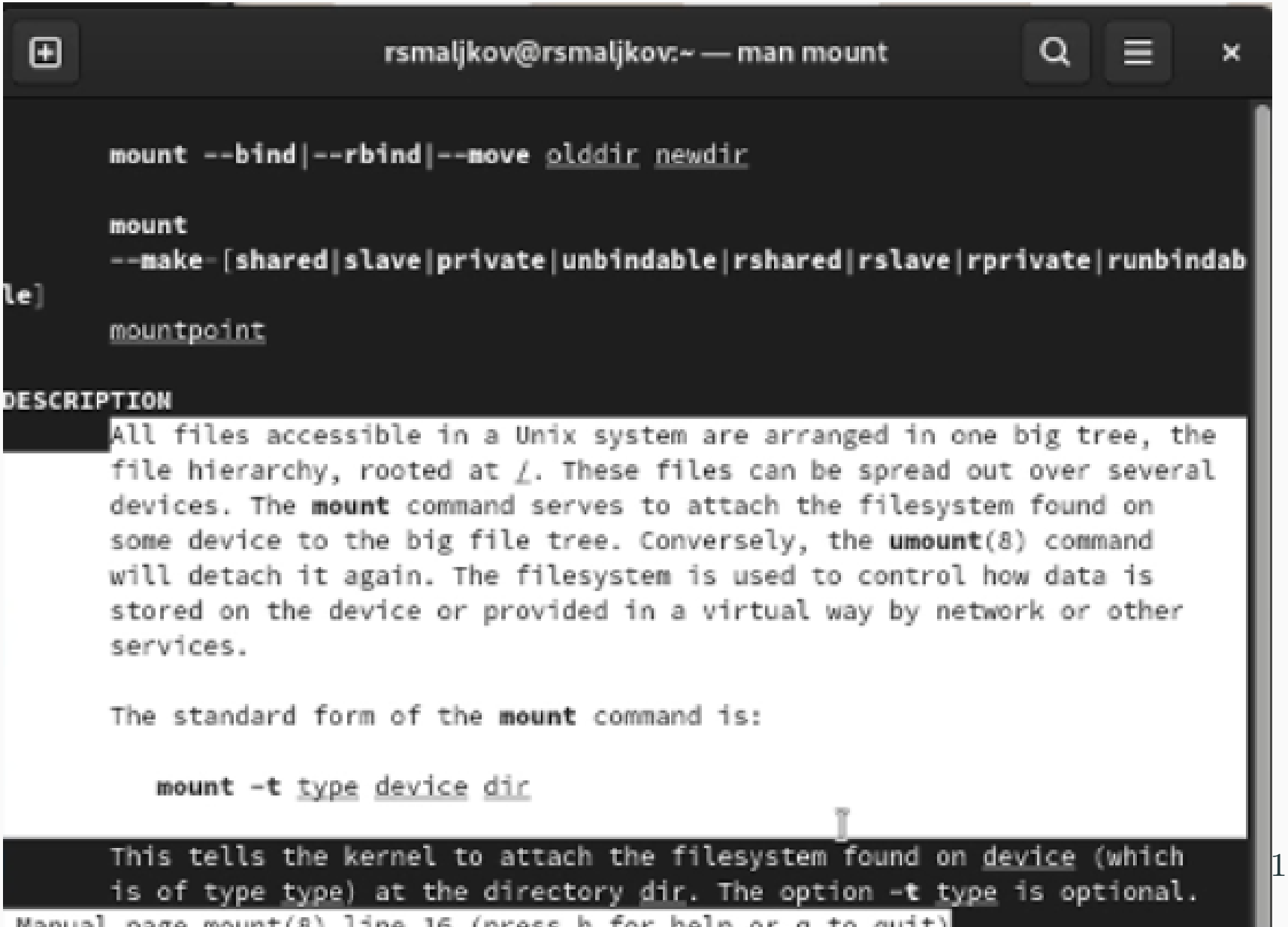
Рис. 2: Screenshot_9

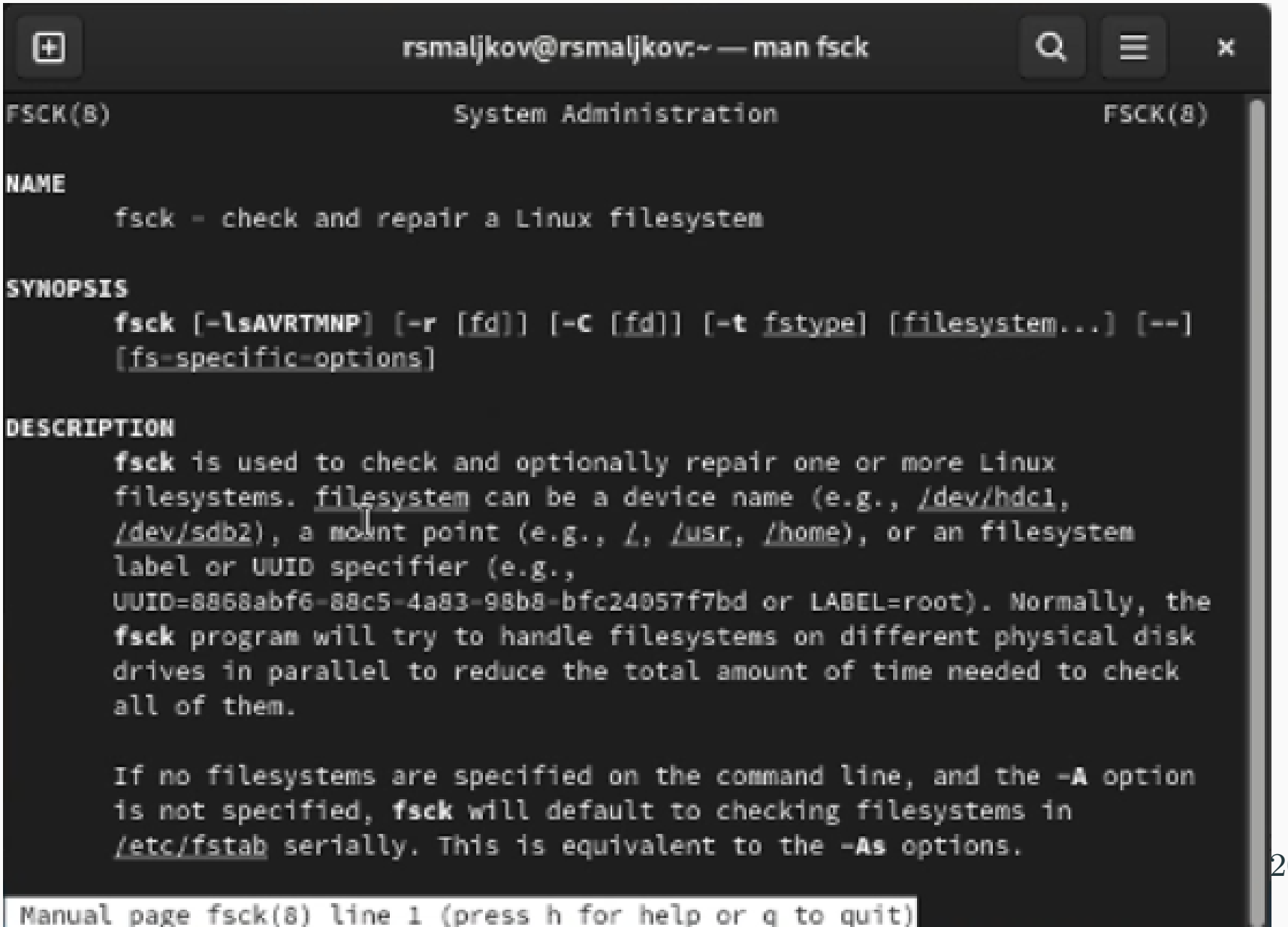
```
[root@rsmaljkov rsmaljkov]# ls -l features
--wxrwxrwx. 1 rsmaljkov rsmaljkov 0 May  4 12:33 features
[root@rsmaljkov rsmaljkov]# chmod u+r features
[root@rsmaljkov rsmaljkov]# ls -l features
-rwxrwxrwx. 1 rsmaljkov rsmaljkov 0 May  4 12:33 features
```

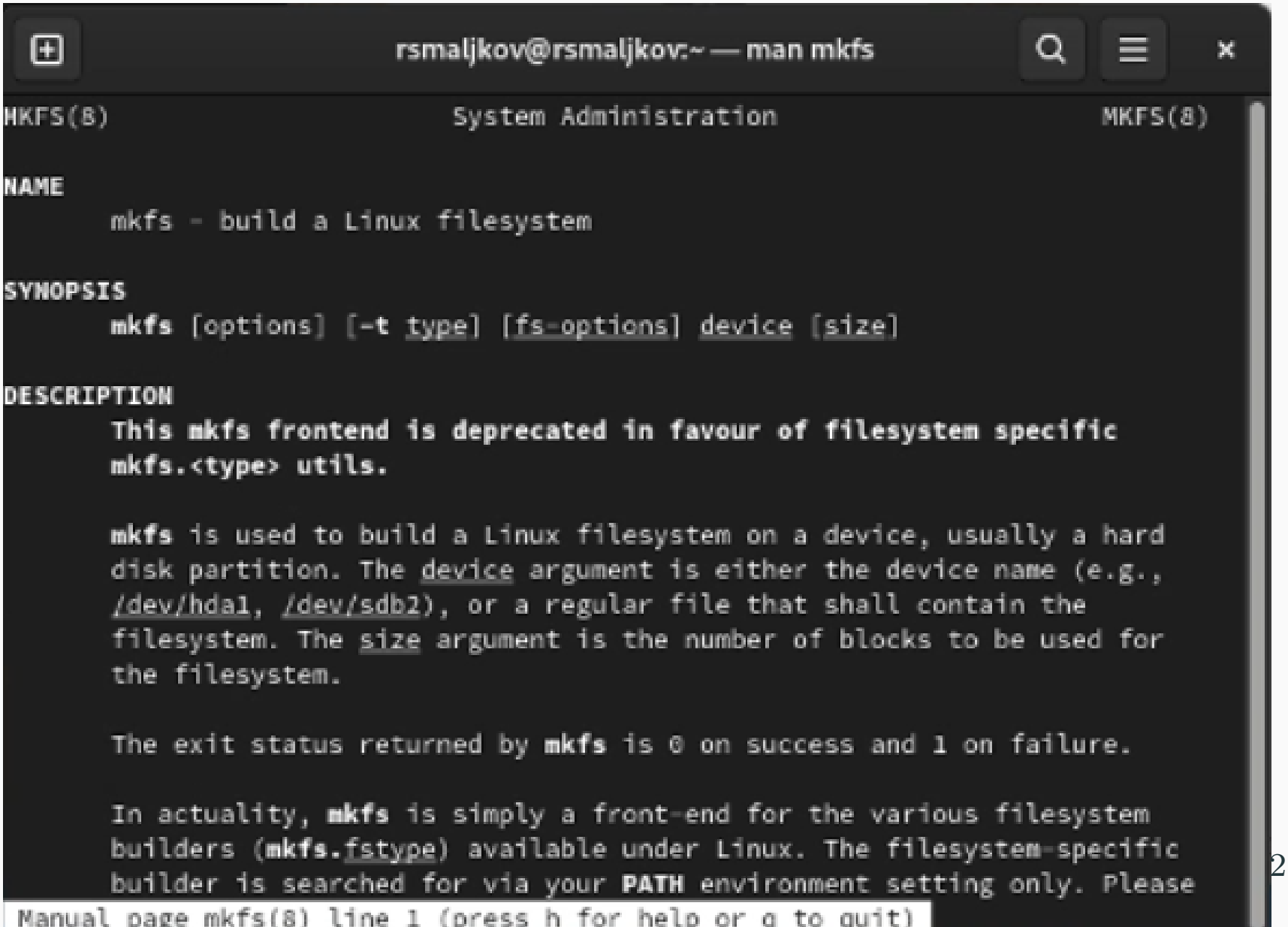
Рис. 3: Screenshot_10

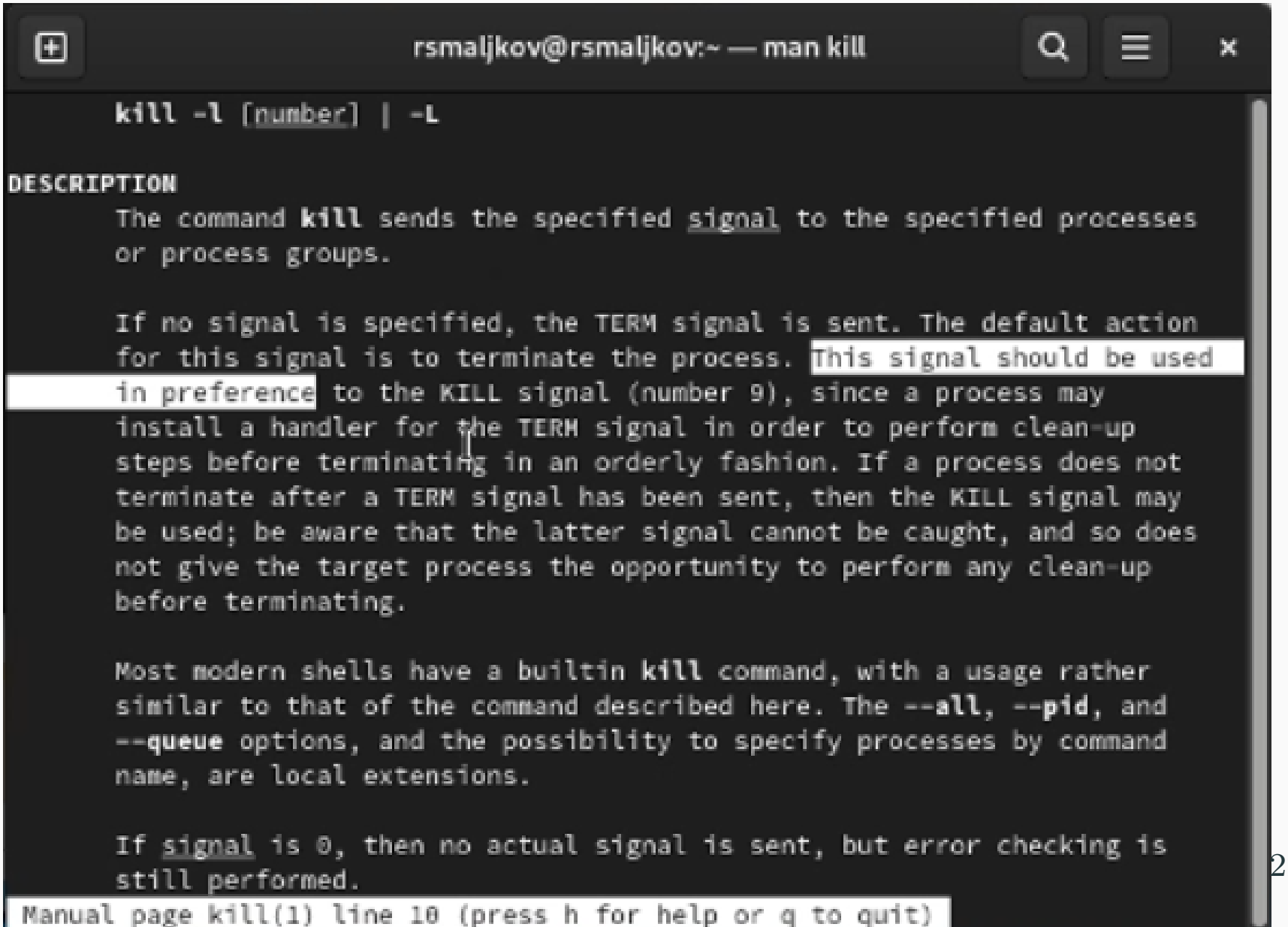
```
[root@rsmaljkov rsmaljkov]# chmod u+x play
[root@rsmaljkov rsmaljkov]#
```

Рис. 4: Screenshot_11


rsmaljkov@rsmaljkov:~ — man mount
mount --bind|--rbind|--move olddir newdir
mount
--make-[shared|slave|private|unbindable|rshared|rslave|rprivate|runbindable]
mountpoint
DESCRIPTION
All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command serves to attach the filesystem found on some device to the big file tree. Conversely, the umount(8) command will detach it again. The filesystem is used to control how data is stored on the device or provided in a virtual way by network or other services.
The standard form of the mount command is:
mount -t type device dir
This tells the kernel to attach the filesystem found on device (which is of type type) at the directory dir. The option -t type is optional.
Manual page mount(8) line 16 (press h for help or q to quit)


rsmaljkov@rsmaljkov:~ — man fsck
FSCK(8) System Administration FSCK(8)
NAME
fsck - check and repair a Linux filesystem
SYNOPSIS
fsck [-lsAVRTMNP] [-r [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=8868abf6-88c5-4a83-98b8-bfc24057f7bd 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.
Manual page fsck(8) line 1 (press h for help or q to quit)


rsmaljkov@rsmaljkov:~ — man mkfs
MKFS(8) System Administration MKFS(8)
NAME
mkfs - build a Linux filesystem
SYNOPSIS
mkfs [options] [-t type] [fs-options] device [size]
DESCRIPTION
This mkfs frontend is deprecated in favour 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/hda1, /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
Manual page mkfs(8) line 1 (press h for help or q to quit)


rsmaljkov@rsmaljkov:~ — man kill
kill -l [number] | -L
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 in an orderly fashion. 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.
Manual page kill(1) line 10 (press h for help or q to quit)

Вывод

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