

# **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>21</b>
<b>5 Ответы на контрольные вопросы:</b>	<b>22</b>
<b>Список литературы</b>	<b>25</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 . . . . .	11
3.8	australia . . . . .	12
3.9	australia . . . . .	12
3.10	australia . . . . .	13
3.11	play . . . . .	13
3.12	feathers . . . . .	14
3.13	На скриншоте все ответы на данные пункты . . . . .	15
3.14	команда man . . . . .	15
3.15	mount . . . . .	16
3.16	mount . . . . .	16
3.17	mount . . . . .	16
3.18	mount . . . . .	17
3.19	mount . . . . .	17
3.20	mount . . . . .	17
3.21	fsck . . . . .	18
3.22	fsck . . . . .	18
3.23	fsck . . . . .	18
3.24	mkfs . . . . .	19
3.25	kill . . . . .	19
3.26	kill . . . . .	19
3.27	kill . . . . .	20

# **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

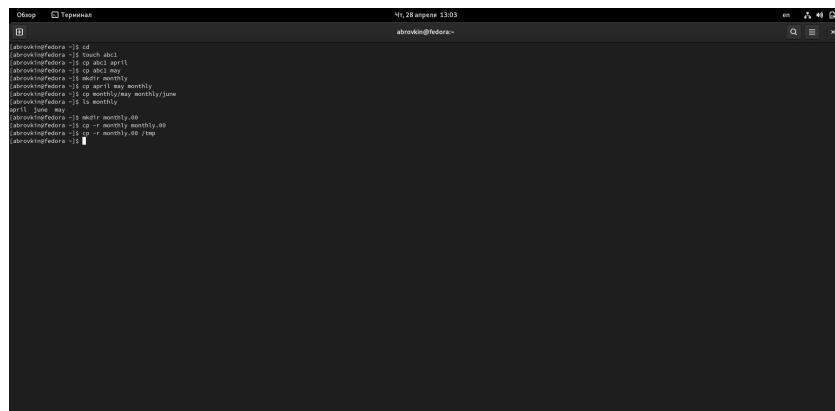
1. Выполните все примеры, приведённые в первой части описания лабораторной работы.
2. Выполните следующие действия, зафиксировав в отчёте по лабораторной работе используемые при этом команды и результаты их выполнения:
  - 2.1. Скопируйте файл '/usr/include/sys/io.h' в домашний каталог и назовите его 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).

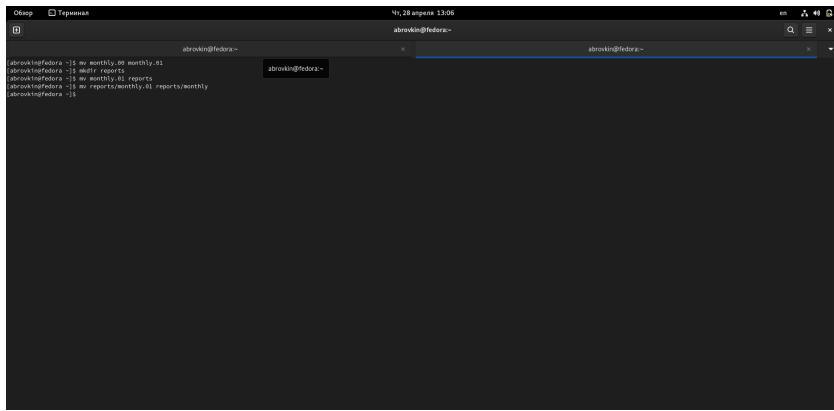


Рис. 3.2: Продолжнаю выполнять примеры

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)(см.рис. 3.4)

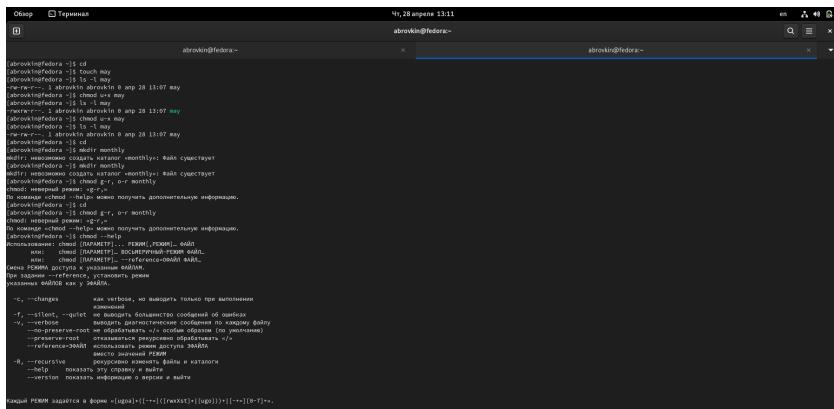
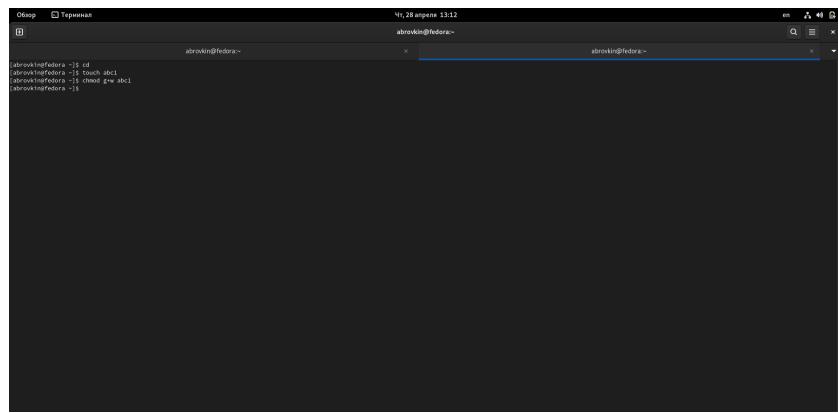


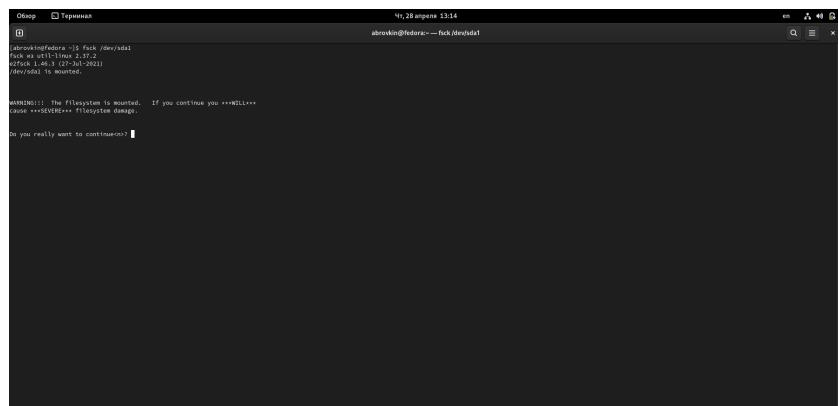
Рис. 3.3: Продолжнаю выполнять примеры



```
abroskin@fedora ~]$ cd
abroskin@fedora ~]$ touch abc1
abroskin@fedora ~]$ chmod g+w abc1
abroskin@fedora ~]$
```

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 объёма свободного пространства на файловой системе. С помощью команды fsck проверил целостность файловой системы.(см.рис. 3.5)



```
abroskin@fedora ~]$ fsck /dev/sda1
Filesystem check time: 2.37.2
Filesystem was ext3 (ext3)
Filesystem UUID: 00000000-0000-0000-0000-000000000000
/dev/sda1 is mounted.

WARNING!!: The filesystem is mounted. If you continue you ***MILLION*** cause ***SEVERE*** filesystem damage.

Do you really want to continue? [y/n] y
```

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)(см.рис. 3.7)

```
Обзор Терминал Чр, 28 апреля 13:21 abrovkin@fedorac-  
[abrovkin@fedorac- ~]$ cp /usr/include/sys/rb.h /home/abrovkin  
[abrovkin@fedorac- ~]$ cd /home/abrovkin  
[abrovkin@fedorac- ~]$ gedit rb.h  
[abrovkin@fedorac- ~]$ mv equipment ~/skiplases  
[abrovkin@fedorac- ~]$ rm -rf equipment  
[abrovkin@fedorac- ~]$ mv equipment ~/skiplases  
[abrovkin@fedorac- ~]$ rm -rf skiplasses  
[abrovkin@fedorac- ~]$ rm -rf skiplasses/equipment  
[abrovkin@fedorac- ~]$ rm -rf skiplasses/equipment/equiplist  
[abrovkin@fedorac- ~]$ rm -rf skiplasses/equipment/equiplist  
[abrovkin@fedorac- ~]$ gedit abc1  
[abrovkin@fedorac- ~]$ rm -rf abc1  
[abrovkin@fedorac- ~]$ rm -rf abc1  
[abrovkin@fedorac- ~]$ mv -r abc1 plaswa/abc1 ~/skiplases/equiplist  
[abrovkin@fedorac- ~]$
```

Рис. 3.6: Продолжнаю выполнять примеры

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) (см.рис. 3.9)(см.рис. 3.10)(см.рис. 3.11)(см.рис. 3.12)

### 3.1. drwxr–r– ... australia

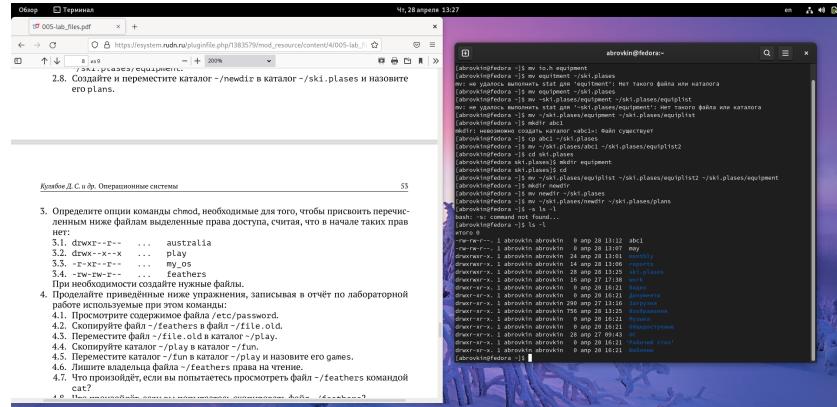


Рис. 3.8: australia

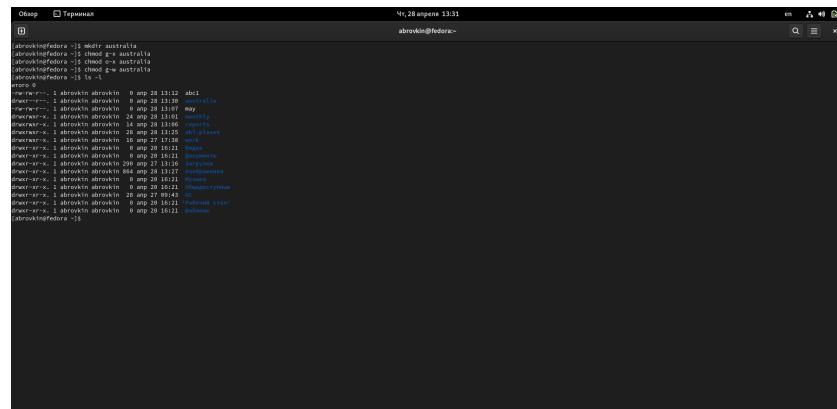


Рис. 3.9: australia

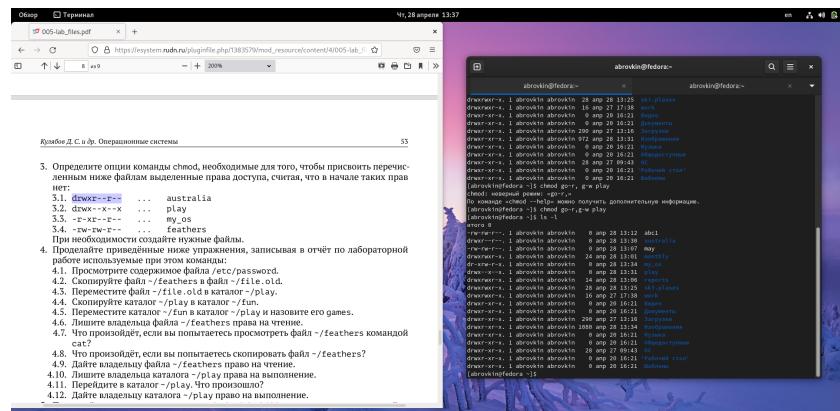


Рис. 3.10: australia

### 3.2. drwx-x-x ... play

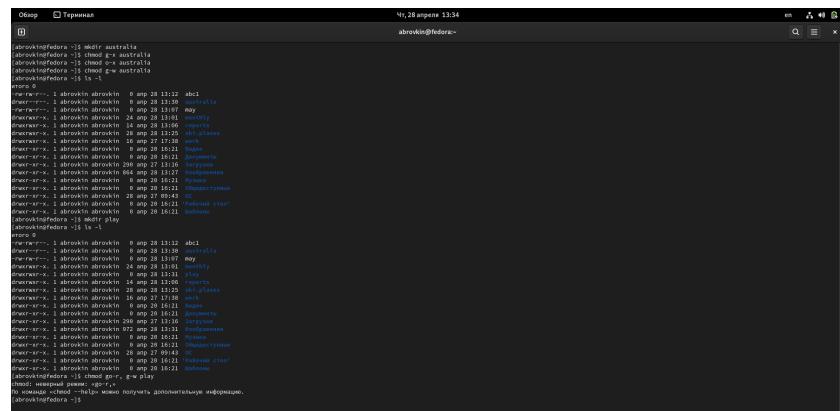


Рис. 3.11: play

### 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

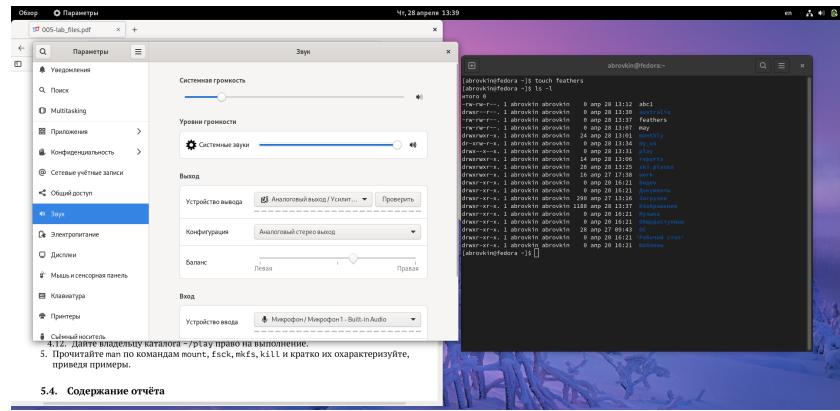


Рис. 3.12: feathers

4. Проделал приведённые ниже упражнения, записывая в отчёт по лабораторной работе используемые при этом команды:(см.рис. 3.13)

- 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.

```
abrukin@fedora: ~$ ls
abc1  .play  work  temporary  unavailable  wtf
feathers  .wtf  .wtf.old
abrukin@fedora: ~$ cp ./feathers ./feathers.old
abrukin@fedora: ~$ rm ./feathers.old
abrukin@fedora: ~$ cp -r ./play ./fun
abrukin@fedora: ~$ cp ./fun ./play
abrukin@fedora: ~$ cd ./play
abrukin@fedora: ~$ ./play
abrukin@fedora: ~$ chmod u+r feathers
abrukin@fedora: ~$ ./play
chmod: недостаточно строк к 'feathers': нет такого файла или каталога
abrukin@fedora: ~$ chmod u+r feathers
abrukin@fedora: ~$ cp ./play ./play.old
abrukin@fedora: ~$ cd ./play.old
abrukin@fedora: ~$ ./play
./play: не могу открыть файл: отказано в доступе
abrukin@fedora: ~$ chmod u+r play
abrukin@fedora: ~$
```

Рис. 3.13: На скриншоте все ответы на данные пункты

5. Прочитал man по командам mount, fsck, mkfs, kill.(см.рис. 3.14)(см.рис. 3.15)(см.рис. 3.16)(см.рис. 3.17)(см.рис. 3.18)(см.рис. 3.19)(см.рис. 3.20)

```
abrukin@fedora: ~$ man mount
abrukin@fedora: ~$ man fsck
abrukin@fedora: ~$ man mkfs
abrukin@fedora: ~$ man kill
abrukin@fedora: ~$
```

Рис. 3.14: команда man

mount

```

SYNOPSIS
        mount [-t type] [device] [mountpoint]
        mount [-t type] [options] [device] [mountpoint]
        mount [-t type] [options] [source] [target]
        mount --bind [target] [source]
        mount --make [shared|slave|private|unbindable|shared|slave|private|unbindable]
        mount --rbind [target]
        mount --remount [options] [device] [mountpoint]
        mount --move [source] [target]
        mount --rmove [source] [target]

DESCRIPTION
        All files accessible in a Unix system are arranged to form big tree. The file hierarchy, rooted at /, contains files and serves several services. The mount command serves to attach the filesystem found on some device to the big file tree. Conversely, the umount command removes the device from the tree. 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. The mount command is usually able to guess the type of the filesystem, so it's possible to omit it. The mount command is root privileged, so it's necessary to run it as root. The mount command can also be used to remount a filesystem by default. See section "Mounting and remounting" below for more details. The mount command (if any) and owner and mode of dir become invisible, and as long as this filesystem remains mounted, the pathname dir refers to the root of the filesystem on device.
        If only the directory or the device is given, for example:
        mount /dev
        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 mount page. Mount(8) lists all devices in /proc/mounts.

```

Рис. 3.15: mount

```

If only the directory or the device is given, for example:
mount /dev
mount looks for a mountpoint (and if no 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

Listing the mounts
mount works in mounting mode for backward compatibility only.
For portability and consistency mount uses fstab(5), 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 (-t type)
The option -t 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 192.168.1.1:/home/nfs.
The device names of disk partitions are unstable; hardware reconfiguration, and adding or removing a device can cause changes in names. This is the reason why it's strongly recommended to use tags or partition identifiers like UUID or LABEL. Currently supported identifiers (tags):
LABEL=label
        Human readable filesystem identifier. See also -L.
UUID=uuid
        Unique universally unique identifier. The format of the UUID is usually a series of hex digits separated by hyphens. See also -U.

```

Рис. 3.16: mount

```

Note that mount uses UUIDs as strings. The UUIDs from the command line or from /etc/fstab are not converted to tokenized 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 extfs or msdos operations. It's supported for GUID Partition Tables (GPT).
PARTUUID=uuid
        Unique universally unique identifier. This identifier is independent on filesystem and does not change by extfs or msdos operations. It's supported for GUID Partition Tables (GPT).

Disk IDs
        Hardware block device ID as generated by udev. This identifier is usually based on memory, and storage identifiers assigned by the kernel. The identifier is stored in /dev/disk/by-id and can be used to identify the device and running udev is required. This identifier is not recommended for generic use as the identifier is not unique across different systems.

The command blkid --list provides an overview of filesystems, labels and UUIDs on available block devices. The command blkid -p device provides details about a filesystem on the specific device.

Don't forget that there is no guarantee that UUID and label are really unique, especially if you move them. To copy the device use blkid -w 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 label. The label identifier is stored in the /etc/fstab file. Tags are more readable, robust and portable. The mount(8) command internally uses path and label symbols in getfstab(). getfstab() has no advantage over tags. For more details see blkid(8).

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

```

Рис. 3.17: mount

```
Oknop Терминал 14, 28 април 13:46
abrovkin@fedoravm:— man mount

--fake
        In conjunction with --no -o, fake off a new information of mount for each device,
        this will do the mounts on different devices or different NFS servers to parallel.
        This has the advantage that it is faster; also NFS timeouts proceed in parallel. A
        downside is that the order of the mounts is undefined. Then, you
        cannot use this option if you want to mount both local and network devices.

--fake
        Causes everything to be done except for the actual system call; if it's not
        desired to have the kernel do the actual work, then this option can be used in conjunction
        with the -w flag to determine what the mount command is trying to do. It can also
        be useful for testing the mount command's logic without actually mounting anything.
        The -f option checks for an existing record in fstab and fails when the record
        already exists (with a regular non-fake mount, this check is done by the kernel).

--internally
        Don't call the /sbin/mount.filesystem helper even if it exists.

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

--allow-labels
        Add the labels in the mount output. mount must have permission to read the disk
        device. This option is useful for mounting ext4 partitions using ext4, for ext3 using
        ext3, ext or ext4 using the extlabel() utility, or for xfs using xfs_admin(), or
        for reiserfs using reiserfsprobe().

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

--remountab
        Mount without writing in /etc/mtab. This is necessary for example when /etc is on a
        read-only filesystem.

--remountns
        Perform a remount 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/mtab, writes /etc/mtab
        (it writes to /run/mtab) 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 libmount files, as the mount namespace is necessary to execute the umount(2) call.

        See mount(2) for more information.
        mount --remountns <ns> --no -o opt <file> <dir>
        For help (q) or to quit:
```

Рис. 3.18: mount

```
Obsp [ ] Терминал 41, 28 апгуст 13:46 abrokin@fedorac:— man mount

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

--options-mode mode
Combines multiple 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 the command line are appended to those in the file, the default
value is prepend. It means command line options are evaluated after stab options.
Note that the last option wins if there are conflicting ones.

--options+source source
Source for additional options. source is a comma-separated list of fstab, stab and
disable disable fstab and stab and disables --options+source=force.
The default value is fstab,stab.
--options-source-force
This option overrides options from fstab/mtab even if both device and dir are specified.

--rmdir
Remove a subtree and all possible subunits 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 to -o ro.
Note that, depending on the Filesystem type, sync 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 the device with noatime and nodiratime options or set the block device
to read-only mode, see the blockdev 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 autofs mount helper only.

--source device
If the device 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 which argument is the mount source.

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

Рис. 3.19: mount

It is possible for a corrupted filesystem to cause a crash.

Some Linux filesystems don't support `-o sync` and `-o atime=` (like ext2, ext3, ext4, fat and vfat filesystems) do support synchronous updates (4 to 800) when mounted with the `sync` option.

The `-o remount` may not be able to change mount parameters (all ext2fs-specific parameters are considered incompatible with a remount, for example, but you can change `gid` or `umask` for the fat32).

It is possible that the file `/etc/fstab` and `/proc/mounts` don't match on systems with a regular `statfs` file. The `fstab` file is based only on the `mount` command options, but the `statfs` file is based on the actual state of the mounted file system. This can happen if a remote NFS server — in certain cases the `mount` command may report unreliable information — has been unmounted and then remounted with different options (more reliable information). This is another reason to replace the `statfs` file with a symlink to the `/proc/mounts` file.

Checking files on NFS filesystems referenced by file descriptors (i.e. the `fd` and `file descriptor` options) will lead to inconsistent results due to the lack of a consistency check in the kernel even if the `noatime` option is used.

The `loop` option with the `offset` or `startoffset` options used may fail when using older kernels as the `mount` command can't confirm the size of the block device has been covered by the offset. It is recommended to be done and tested by the `losetup(4)` command manually before calling `mount` with the configured loop device.

**AUTHORS**

Original author: dzakir@redhat.com

**SEE ALSO**

`mount(2)`, `umount(2)`, `Filesystems(5)`, `fsck(8)`, `ufs(5)`, `mount namespaces(7)`, `mount(8)`, `stablabel(8)`, `findmnt(8)`, `balld(8)`, `seekfs(8)`, `mountd(8)`, `mountinfo(5)`, `mountstats(5)`, `umountd(8)`

**REPORTING BUGS**

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

**AVAILABILITY**

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

Рис. 3.20· mount

**fsck**(см.рис. 3.21)(см.рис. 3.22)(см.рис. 3.23)

The screenshot shows the man page for the fsck(8) command. The title is 'fsck - check and repair Linux filesystems'. It includes sections for SYNOPSIS, DESCRIPTION, and NOTES. The NOTES section provides detailed information about how fsck handles multiple filesystems, including how it checks them sequentially or in parallel based on options like -A, -a, and -r. It also discusses the compatibility with /etc/fstab and the use of /etc/fsckrc.

Рис. 3.21: fsck

This screenshot is another view of the fsck(8) man page. It highlights the 'OPTIONS' section, which lists various command-line flags such as -A, -a, -c, -e, -f, -l, -L, -n, -p, -r, -t, -v, and -w. Each option is described with its purpose and usage.

Рис. 3.22: fsck

This screenshot shows the bottom half of the fsck(8) man page, containing the 'ENVIRONMENT' and 'FILES' sections. The 'ENVIRONMENT' section details environment variables like FCK\_FORCE\_ALL\_PARALLEL, FCK\_MAX\_INSTANCES, and PATH. The 'FILES' section lists the /etc/fstab file as the primary configuration source.

Рис. 3.23: fsck

**mkfs**(см.рис. 3.24)

```

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 new filesystem on a device, usually a hard disk partition. The device argument is either the device name (e.g., /dev/hda), 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.*type) 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
                print 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
        Original code: David Holland, Eric W. Vaughan, Michael Ellerman, Alan Cox, Matt Mackall <esmelle@clix.unl.edu>, Alan Cox

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

SEE ALSO
        fd(5), badblocks(8), fsck(8), mkdeffs(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) Time 1.00ms h for help or q to quit]

```

Рис. 3.24: mkfs

kill(см.рис. 3.25)(см.рис. 3.26)(см.рис. 3.27)

```

kill - terminate a process

SYNOPSIS
        kill [-signal|-#signal|-n] [-q value] [-w timeout] signal [-gids...]
        kill -l [number] | -v

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 cleanup 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 cleanup before terminating.

        Most modern shells have a built-in kill command, with a usage rather similar to that of the command described here. The -n, -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.

OPTIONS
        The list of processes to be signalled can be a mixture of names and PIDs.

        pid
                Each pid can be expressed in one of the following ways:
                0           where 0 is larger than 1. The process with PID 0 is signalled.
                n           All processes in the current process group are signalled.
                -$          All processes with a PID larger than 1 are signalled.
                --pid      where $ is larger than 1. All processes in process group $ are signalled. When an argument of the form '-m' 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 signalled.

OPTIONS
        -s, --signal signal
                Send 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.

        -w, --timeout timeout
                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
                Prints PIDs that will be signalled with KILL along with the signal.

        -q, --quiet
                Sends the signal using sigqueue() rather than kill(). 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 sigqueue(), then it can obtain this data via the _SIGINFO field of the signal_info structure.

        --timeout milliseconds
                Used to signal the user 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 -lqsigterm -t1000 -sQUIT -sTERM -sKILL

EXIT STATUS
        kill has the following exit status values:
        0
        [Manual page kill(1) Time 1.00ms h for help or q to quit]

```

Рис. 3.25: kill

```

kill - terminate a process

SYNOPSIS
        kill [-signal|-#signal|-n] [-q value] [-w timeout] signal [-gids...]
        kill -l [number] | -v

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

OPTIONS
        The list of processes to be signalled can be a mixture of names and PIDs.

        pid
                Each pid can be expressed in one of the following ways:
                0           where 0 is larger than 1. The process with PID 0 is signalled.
                n           All processes in the current process group are signalled.
                -$          All processes with a PID larger than 1 are signalled.
                --pid      where $ is larger than 1. All processes in process group $ are signalled. When an argument of the form '-m' 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 signalled.

OPTIONS
        -s, --signal signal
                Send 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.

        -w, --timeout timeout
                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
                Prints PIDs that will be signalled with KILL along with the signal.

        -q, --quiet
                Sends the signal using sigqueue() rather than kill(). 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 sigqueue(), then it can obtain this data via the _SIGINFO field of the signal_info structure.

        --timeout milliseconds
                Used to signal the user 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 -lqsigterm -t1000 -sQUIT -sTERM -sKILL

EXIT STATUS
        kill has the following exit status values:
        0
        [Manual page kill(1) Time 1.00ms h for help or q to quit]

```

Рис. 3.26: kill

```

kill(2) Linux Programmer's Manual

kill(2) [ man page ] [ search ] [ help ] [ exit ]

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 than 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 -v -SIGQUIT --timeout 1000 TERM --timeout 1000 KILL \
    --signal QUIT 1245

EXIT STATUS
  kill has the following exit status values:
  0       success
  1       failure
  >      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) and not to the specific thread. This is because the kernel only explicitly tracks the 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
  Documentation: christophe.leroy@cvelentegeut.edu, karel.zak@zku.cz
  The original version was taken from BSD 4.4.

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

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

AVAILABILITY
  The kill command is part of the util-linux package which can be downloaded from https://www.kernel.org/pub/linux/utils/util-linux/.
  kill(2) last updated: 2021-06-02
  kill(2) [ man page ] [ search ] [ help ] [ exit ]

```

Рис. 3.27: kill

**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]