

Лабораторная работа №6

Журавлев Георгий Иванович

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.

Выполнение лабораторной работы

1. Выполняем все примеры, приведённые в первой части описания лабораторной работы.

1.1. Копирование файла в текущем каталоге. Скопировать файл `~/abc1` в файл `april` и в файл `may`

```
g_zhuravlev@g ~ /github/gzhuravlev.pfur/2020-2021/OS/labs/lab05 > main cd
g_zhuravlev@g ~ touch abc1
g_zhuravlev@g ~ cp abc1 april
g_zhuravlev@g ~ cp abc1 may
g_zhuravlev@g ~ tree
```

| | | |
|---|-----|-------|
| ✓ | 384 | 13:34 |
| ✓ | 385 | 13:37 |
| ✓ | 386 | 13:37 |
| ✓ | 387 | 13:37 |

Рис. 1: Создание файлов и копирование

1.2. Копирование нескольких файлов в каталог. Скопировать файлы `april` и `may` в каталог `monthly`

1.3. Копирование файлов в произвольном каталоге. Скопировать файл `monthly/may` в файл с именем `june`

```
g_zhuravlev@g ~ mkdir monthly
g_zhuravlev@g ~ cp april may monthly
g_zhuravlev@g ~ cp monthly/may monthly/june
g_zhuravlev@g ~ ls monthly
april june may
g_zhuravlev@g ~
```

Рис. 2: Создание директории и копирование файлов

- 1.4. Копирование каталогов в текущем каталоге. Скопировать каталог `monthly` в каталог `monthly.00`
- 1.5. Копирование каталогов в произвольном каталоге. Скопировать каталог

```
g_zhuravlev@g ~$ mkdir monthly.00
g_zhuravlev@g ~$ cp -r monthly monthly.00
g_zhuravlev@g ~$ cp -r monthly.00 /tmp
```

Рис. 3: Копирование каталогов в новый каталог

- 1.6. Переименование файлов в текущем каталоге. Изменить название файла `april` на `july` в домашнем каталоге
- 1.7. Перемещение файлов в другой каталог. Переместить файл `july` в каталог `monthly.00`
- 1.8. Переименование каталогов в текущем каталоге. Переименовать каталог `monthly.00` в `monthly.01`
- 1.9. Перемещение каталога в другой каталог. Переместить каталог `monthly.01` в каталог `reports`
- 1.10. Переименование каталога, не являющегося текущим. Переименовать каталог `reports/monthly.01` в `reports/monthly.02`
- 1.11. Требуется создать файл `~/may` с правом выполнения для владельца
- 1.12. Требуется лишить владельца файла `~/may` права на выполнение
- 1.13. Требуется создать каталог `monthly` с запретом на чтение для членов группы и всех остальных пользователей
- 1.14. Требуется создать файл `~/abc1` с правом записи для членов группы.

2. Выполним следующие действия.

- 2.1. Скопируем файл `/usr/include/search.h` в домашний каталог и назовём его `equipment`.
- 2.2. В домашнем каталоге создадим директорию `~/ski.places`.
- 2.3. Переместим файл `equipment` в каталог `~/ski.places`.
- 2.4. Переименуем файл `~/ski.places/equipment` в `~/ski.places/equiplist`.
- 2.5. Создадим в домашнем каталоге файл `abc1` и скопируем его в каталог `~/ski.places`, назовём его `equiplist2`.
- 2.6. Создадим каталог с именем `equipment` в каталоге `~/ski.places`.
- 2.7. Переместим файлы `~/ski.places/equiplist` и `equiplist2` в каталог `~/ski.places/equipment`.
- 2.8. Создадим и переместим каталог `~/newdir` в каталог `~/ski.places` и назовём его `plans`.

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

начальные значения: -----

- 3.1. Создаем каталоги и файлы.
 - 3.2. изменяем права командой `chmod`.
1. for `australia`: `mkdir australia(d) -> chmod u+r, u+w, u+x, g+r, o+r australia`
 2. for `play`: `mkdir play(d) -> chmod u+r, u+w, u+x, g+x, o+x play`
 3. for `my_os`: `touch my_os(-) -> chmod u+r, u+x, g+r, o+r my_os`
 4. for `feathers`: `touch feathers -> chmod u+r, u+w, g+r, g+w, o+r feathers`

```
g_zhuravlev@g:~  
g_zhuravlev@g ~$ mv april july  
g_zhuravlev@g ~$ mv july monthly.00  
g_zhuravlev@g ~$ ls monthly.00  
july  monthly  
g_zhuravlev@g ~$ tree monthly.00  
monthly.00  
├── july  
└── monthly  
    ├── april  
    ├── june  
    └── may  
1 directory, 4 files  
g_zhuravlev@g ~$ cd monthly.00  
g_zhuravlev@g ~/monthly.00$ rm -rf monthly  
g_zhuravlev@g ~/monthly.00$ tree monthly.00  
monthly.00 [error opening dir]  
0 directories, 0 files  
g_zhuravlev@g ~/monthly.00$ tree  
├── july  
0 directories, 1 file  
g_zhuravlev@g ~/monthly.00$ cd  
g_zhuravlev@g ~$ cd monthly  
g_zhuravlev@g ~$ cp june may april monthly.00  
cp: target 'monthly.00' is not a directory  
g_zhuravlev@g ~$ cd  
g_zhuravlev@g ~$ cp -r monthly monthly.00  
g_zhuravlev@g ~$ tree monthly.00  
monthly.00  
├── july  
└── monthly  
    ├── april  
    ├── june  
    └── may  
1 directory, 4 files  
g_zhuravlev@g ~$
```

Рис. 4: Переименовываем и перемещаем файл

```
1 directory, 4 files  
g_zhuravlev@g ~$ mv monthly.00 monthly.01  
g_zhuravlev@g ~$ mkdir reports  
g_zhuravlev@g ~$ mv monthly.01 reports  
g_zhuravlev@g ~$ mv reports/monthly.01 reports/monthly
```

Рис. 5: Переименовываем с помощью mv

```
g_zhuravlev@g ~/reports$ cd  
g_zhuravlev@g ~$ touch may
```

Рис. 6: Создание файла

```
g_zhuravlev@g ~$ ls -l may
-rw-rw-r-- 1 g_zhuravlev g_zhuravlev 0 мая  2 13:49 may
g_zhuravlev@g ~$ chmod u+x may
g_zhuravlev@g ~$ ls -l may
-rwxr--r-- 1 g_zhuravlev g_zhuravlev 0 мая  2 13:49 may
g_zhuravlev@g ~$ chmod u-x may
g_zhuravlev@g ~$ ls -l may
-rw-rw-r-- 1 g_zhuravlev g_zhuravlev 0 мая  2 13:49 may
```

Рис. 7: Изменение прав

```
g_zhuravlev@g ~$ mkdir month
g_zhuravlev@g ~$ chmod g-r month ; chmod o-r month
```

Рис. 8: Изменение прав на чтение каталога

```
g_zhuravlev@g ~$ touch abc1
g_zhuravlev@g ~$ chmod g+w abc1
g_zhuravlev@g ~$ ls -l abc1
-rw-rw-r-- 1 g_zhuravlev g_zhuravlev 0 мая  2 13:59 abc1
```

Рис. 9: Изменение прав на запись файла

```
g_zhuravlev@g /usr/include$ sudo cp search.h /home/equipment
[sudo] password for g_zhuravlev:
g_zhuravlev@g /usr/include$
```

Рис. 10: Копирование каталога

```
g_zhuravlev@g ~$ mkdir ski.plases
g_zhuravlev@g ~$ mv equipment ski.plases
mv: cannot stat 'equipment': No such file or directory
g_zhuravlev@g ~$ mv equipment.h ski.plases
g_zhuravlev@g ~$ mv ski.plases/equipment.h ski.plases/equiplist.h
g_zhuravlev@g ~$ touch abc2
g_zhuravlev@g ~$ cp abc2 ski.plases
g_zhuravlev@g ~$ mv ski.plases/abc2 ski.plases/equiplist2
g_zhuravlev@g ~$ mkdir ski.plases/equipment
g_zhuravlev@g ~$ mv ski.plases/equiplist.h ski.plases/equiplist2 ski.plases/equipment
```

Рис. 11: Выполнение вышеописанных шагов

```
g_zhuravlev@g ~$ tree ski.plases
ski.plases
├── equipment
├── equiplist2
└── equiplist.h
```

Рис. 12: подтверждение результата

```
g_zhuravlev@g ~$ mkdir newdir
g_zhuravlev@g ~$ mv newdir ski.plases/plans
g_zhuravlev@g ~$ tree ski.plases
ski.plases
├── equipment
├── equiplist2
├── equiplist.h
└── plans

2 directories, 2 files
g_zhuravlev@g ~$
```

Рис. 13: Создание и перемещение каталога

```
g_zhuravlev@g ~$ mkdir australia play my_os feathers
g_zhuravlev@g ~$ ls -l australia play my_os feathers
australia:
total 0
feathers:
total 0
my_os:
total 0
play:
total 0
g_zhuravlev@g ~$ rmdir my_os feathers
g_zhuravlev@g ~$ touch my_os feathers
g_zhuravlev@g ~$ ls -l australia play my_os feathers
-rw-rw-r-- 1 g_zhuravlev g_zhuravlev  0 мая 2 14:21 feathers
-rw-rw-r-- 1 g_zhuravlev g_zhuravlev  0 мая 2 14:21 my_os
australia:
total 0
play:
total 0
g_zhuravlev@g ~$ chmod u+x australia play my_os ; chmod g-w australia my_os ; chmod g-r play ; c
hmod o-r play; chmod o+x play
g_zhuravlev@g ~$ ls -l australia play my_os feathers
-rw-rw-r-- 1 g_zhuravlev g_zhuravlev  0 мая 2 14:21 feathers
-rwxr--r-- 1 g_zhuravlev g_zhuravlev  0 мая 2 14:21 my_os
australia:
total 0
play:
total 0
g_zhuravlev@g ~$ chmod u-w my_os
g_zhuravlev@g ~$ ls -l australia play my_os feathers
-rw-rw-r-- 1 g_zhuravlev g_zhuravlev  0 мая 2 14:21 feathers
-r--xr--r-- 1 g_zhuravlev g_zhuravlev  0 мая 2 14:21 my_os
australia:
total 0
play:
total 0
g_zhuravlev@g ~$
```

Рис. 14: Создание и изменение прав

4. Проведем приведённые упражнения.

4.1. Посмотрим содержимое файла `/etc/passwd`. (команда `sudo cat`)

```
g_zhuravlev@g ~$ cat /etc/passwd/ 1 494 14:27:35
cat: /etc/passwd/: No such file or directory
g_zhuravlev@g ~$ sudo cat /etc/passwd 1 495 14:27:39
cat: /etc/passwd: No such file or directory
g_zhuravlev@g ~$ sudo cat password 1 496 14:27:49
cat: password: No such file or directory
g_zhuravlev@g ~$ sudo cat passwords 1 497 14:28:09
cat: passwords: No such file or directory
```

Рис. 15: Просмотр содержимого

4.2. Скопируем файл `~/feathers` в файл `~/file.old`.

4.3. Переместим файл `~/file.old` в каталог `~/play`.

4.4. Скопируем каталог `~/play` в каталог `~/fun`.

4.5. Переместим каталог `~/fun` в каталог `~/play` и назовём его `games`.

```
g_zhuravlev@g ~$ cp feathers file.old 500 14:29:25
g_zhuravlev@g ~$ mv file.old play 501 14:29:42
g_zhuravlev@g ~$ cp -r play fun 502 14:29:56
g_zhuravlev@g ~$ mv fun play/games 503 14:30:10
```

Рис. 16: Выполнение действий

4.6. Лишим владельца файла `~/feathers` права на чтение.

```
g_zhuravlev@g ~$ chmod u-r feathers 507 14:32:15
g_zhuravlev@g ~$ ls -l feathers 508 14:32:32
--w-rw-r-- 1 g_zhuravlev g_zhuravlev 0 мая 2 14:21 feathers
```

Рис. 17: Лишение прав

4.7. Посмотрим этот файл с помощью `cat` и 4.8. Попробуем скопировать

Ответ: нехватка прав.

4.9. Дадим владельцу файла `~/feathers` право на чтение.

4.10. Лишим владельца каталога `~/play` права на выполнение.

4.11. Перейдём в каталог `~/play` -> нехватка прав.

4.12. Дадим владельцу каталога `~/play` право на выполнение.

5. Прочитаем ман по командам `mount`, `fsck`, `mkfs`, `kill` и кратко их охарактеризуем, приведя примеры.

1. `mount`

```
NAME
mount - mount a filesystem
```

```

g_zhuravlev@g ~$ cat feathers
cat: feathers: Permission denied
g_zhuravlev@g ~$ cp feathers copy
cp: cannot open 'feathers' for reading: Permission denied
g_zhuravlev@g ~$ chmod u+r feathers
g_zhuravlev@g ~$ chmod u-x play
g_zhuravlev@g ~$ cd play
cd: permission denied: play
g_zhuravlev@g ~$ chmod u+x play
g_zhuravlev@g ~$

```

Рис. 18: Просмотр файла и проба копирования

```

g_zhuravlev@g ~$ cat feathers
cat: feathers: Permission denied
g_zhuravlev@g ~$ cp feathers copy
cp: cannot open 'feathers' for reading: Permission denied
g_zhuravlev@g ~$ chmod u+r feathers
g_zhuravlev@g ~$ chmod u-x play
g_zhuravlev@g ~$ cd play
cd: permission denied: play
g_zhuravlev@g ~$ chmod u+x play
g_zhuravlev@g ~$

```

Рис. 19: Выполнение действий

```

-a, --all
    Mount all filesystems (of the given types) mentioned in fstab (except for those
    whose line contains the noauto keyword). The filesystems are mounted following
    their order in fstab. The mount command compares filesystem source, target (and fs
    root for bind mount or btrfs) to detect already mounted filesystems. The kernel
    table with already mounted filesystems is cached during mount --all. It means that all
    duplicated fstab entries will be mounted.

    The option --all is possible to use for remount operation too. In this case all fil-
    ters (-t and -O) are applied to the table of already mounted filesystems.

    Note that it is a bad practice to use mount -a for fstab checking. The recommended
    solution is findmnt --verify.

-B, --bind
    Remount a subtree somewhere else (so that its contents are available in both
    places). See above, under Bind mounts.

-c, --no-canonicalize
    Don't canonicalize paths. The mount command canonicalizes all paths (from command
    line or fstab) by default. This option can be used together with the -f flag for
    already canonicalized absolute paths. The option is designed for mount helpers
    which call mount -i. It is strongly recommended to not use this command-line option
    for normal mount operations.

    Note that mount(8) does not pass this option to the /sbin/mount.type helpers.

-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 go in parallel. A dis-
    advantage is that the mounts are done in undefined order. 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 'fakes' mounting the filesystem. This option is useful in conjunction with
    the -v flag to determine what the mount 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 al-
    ready 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 de-

```

```

    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 mount in namespace specified by ns. ns is either PID of process running in
    that namespace or special file representing that namespace.

    mount(8) switches to the namespace when it reads /etc/fstab, writes /etc/mtab (or
    writes to /run/mount) and calls mount(2) system call, otherwise it runs in the orig-
    inal namespace. It means that the target namespace does not have to contain any li-
    braries or another requirements necessary to execute mount(2) command.

    See namespaces(7) for more information.

-o, --test-opts opts
    Limit the set of filesystems to which the -a option applies. In this regard it is
    like the -t option except that -o is useless without -a. For example, the command:

        mount -a -o no_netdev

    mounts all filesystems except those which have the option _netdev specified in the
    options field in the /etc/fstab file.

    It is different from -t in that each option is matched exactly; a leading no at the
    beginning of one option does not negate the rest.

    The -t and -o options are cumulative in effect; that is, the command

        mount -a -t ext2 -o _netdev

    mounts all ext2 filesystems with the _netdev option, not all filesystems that are
    either ext2 or have the _netdev option specified.

-o, --options opts
    Use the specified mount options. The opts argument is a comma-separated list. For
    example:

        mount LABEL=mydisk -o noatime,nodev,nosuid

    For more details, see the FILESYSTEM-INDEPENDENT MOUNT OPTIONS and FILESYSTEM-SPE-
    CIFIC MOUNT OPTIONS sections.

--options-mode mode
    Controls how to combine options from fstab/mtab with options from 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 command line. Default value is
    append.

```

Описание: Команда mount монтирует устройство и позволяет присоединить хранящиеся на нем файлы к
 Структура: mount [device_name] [mount_point]

2. fsck

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. `filesystems` 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 `LA-BEL=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 code 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 code returned when multiple filesystems are checked is the bit-wise OR of the exit codes 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.

OPTIONS

`-l` Create an exclusive `flock(2)` lock file (`/run/fsck/<diskname>.lock`) for whole-disk device. This option can be used with one device only (this means that `-A` and `-l` are mutually exclusive). This option is recommended when more `fsck(8)` instances are executed in the same time. The option is ignored when used for multiple devices or for non-rotating disks. `fsck` does not lock underlying devices when executed to check stacked devices (e.g. MD or DM) - this feature is not implemented yet.

`-r [fd]` Report certain statistics for each `fsck` when it completes. These statistics include the exit status, the maximum run set size (in kilobytes), the elapsed all-clock time and the user and system CPU time used by the `fsck` run. For example:

```

    or other errors. The -f flag option may be used to force 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 code 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.)

    -T
        Don't show the title on startup.

    -V
        Produce verbose output, including all filesystem-specific commands that are executed.

    -?, --help
        Display help text and exit.

    --version
        Display version information and exit.

```

Описание:Fsck команда взаимодействует с соответствующей файловой системой конкретных FSCK кома

1. Проверка на наличие ошибок и подсказывает пользователю интерактивное решение, как решить инди
2. Проверка на наличие ошибок и постарается автоматически исправить все ошибки;
3. Проверка на наличие ошибок без возможности восстановить их, но тогда выдаст ошибки на стандартн

Структура: fsck

3. mkfs

Описание: “make file system” (создать файловую систему). Команда обычно используется для управления

Структура: mkfs -t [fs type] [target device]

4. kill

Описание: Когда вы выполняете команду "kill", то фактически вы посылаете системе сигнал, чтобы заст

Вы можете просмотреть все сигналы с помощью команды: \$ kill -l

Структура: kill [SIGKILL] PID

Вывод

Благодаря этой лабораторной работе, я научился: пользоваться некоторыми командами; копировать каталоги и удалять их; перемещать файлы и каталоги; копировать файлы и каталоги; изменять права доступа к файлам и каталогам.

```

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 code 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 Kenpen (waltje@uwal.nl.mugnet.org)
    Ron Sommeling (sommel@sci.kun.nl)

```

Рис. 20: маңуал команды mkfs

```

NAME
    kill - send a signal to a process

SYNOPSIS
    kill [options] <pid> [...]

DESCRIPTION
    The default signal for kill is TERM. Use -l or -L to list available signals. Particularly
    useful signals include HUP, INT, KILL, STOP, CONT, and 0. Alternate signals may be speci-
    fied in three ways: -9, -SIGKILL or -KILL. Negative PID values may be used to choose whole
    process groups; see the PGID column in ps command output. A PID of -1 is special; it indi-
    cates all processes except the kill process itself and init.

OPTIONS
    <pid> [...]
        Send signal to every <pid> listed.

    -<signal>
    -s <signal>
    --signal <signal>
        Specify the signal to be sent. The signal can be specified by using name or number.
        The behavior of signals is explained in signal\(7\) manual page.

    -l, --list [signal]
        List signal names. This option has optional argument, which will convert signal
        number to signal name, or other way round.

    -L, --table
        List signal names in a nice table.

NOTES
    Your shell (command line interpreter) may have a built-in kill command. You may
    need to run the command described here as /bin/kill to solve the conflict.

```

Рис. 21: мануал команды kill

Контрольные вопросы:

1 и 2. Файловая система в дисплейном классе содержит следующие каталоги первого уровня:

/bin - Основные программы, необходимые для работы в системе: командные оболочки shell, основные утилиты.

/boot - Каталог, который содержит ядро системы— главную программу, загружающую и исполняющую в

/dev - Каталог, в котором содержатся псевдофайлы устройств. С точки зрения Linux все физические устрой

/etc - В этом каталоге содержатся системные конфигурационные файлы — текстовые файлы, которые счит

/home - В структуре файловой системы Linux каждый пользователь имеет отдельный личный каталог для

/mnt - Каталоги для монтирования файловых систем сменных устройств и внешних файловых систем.

/proc - Файловая система на виртуальном устройстве, её файлы содержат информацию о текущем состояни

/root - Каталог администратора системы.

/sbin - Системные утилиты.

/usr - Программы и библиотеки, доступные пользователю.

/var - Рабочие файлы программ, различные временные данные: очереди (письма на отправку, файлы на пе

/tmp - Временные файлы.

3. Чтобы содержимое некоторой файловой системы было доступно операционной системе должно быть выполнено следующее:

4. Основные причины нарушения целостности файловой системы:

- Один блок адресуется несколькими inode (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. mkfs создаёт новую файловую систему.

6. Характеристика команд, которые позволяют просмотреть текстовые файлы:

- для просмотра небольших файлов - cat.
- для просмотра больших файлов - less — она позволяет осуществлять постраничный просмотр файлов.
- для просмотра начала файла - head[-n], по умолчанию она выводит первые 10 строк файла.
- команда tail[-n] 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.
- -i в команде cp выведет на экран запрос подтверждения о перезаписи файла, если на место целевого файла

Команда `cp` с опцией `r` (recursive) позволяет рекурсивно копировать каталоги вместе с входящими в них файлами.

8. Команды `mv` и `mkdir` предназначены для перемещения и переименования файлов и каталогов.

Формат команды: `mv [-option] старый_файл новый_файл`.

Для получения предупреждения перед переписыванием файла стоит использовать опцию `i`.

9. Права доступа определяют, кто и что может делать с содержимым файла. Существуют три группы прав:
= установить право;

- лишить права;

+ дать право;

`r` чтение;

`w` запись;

`x` выполнение;

`u` (user) владелец файла;

`g` (group) группа, к которой принадлежит владелец файла;

`o` (others) все остальные.