

Отчёт по лабораторной работе № 6
дисциплина: Операционные системы

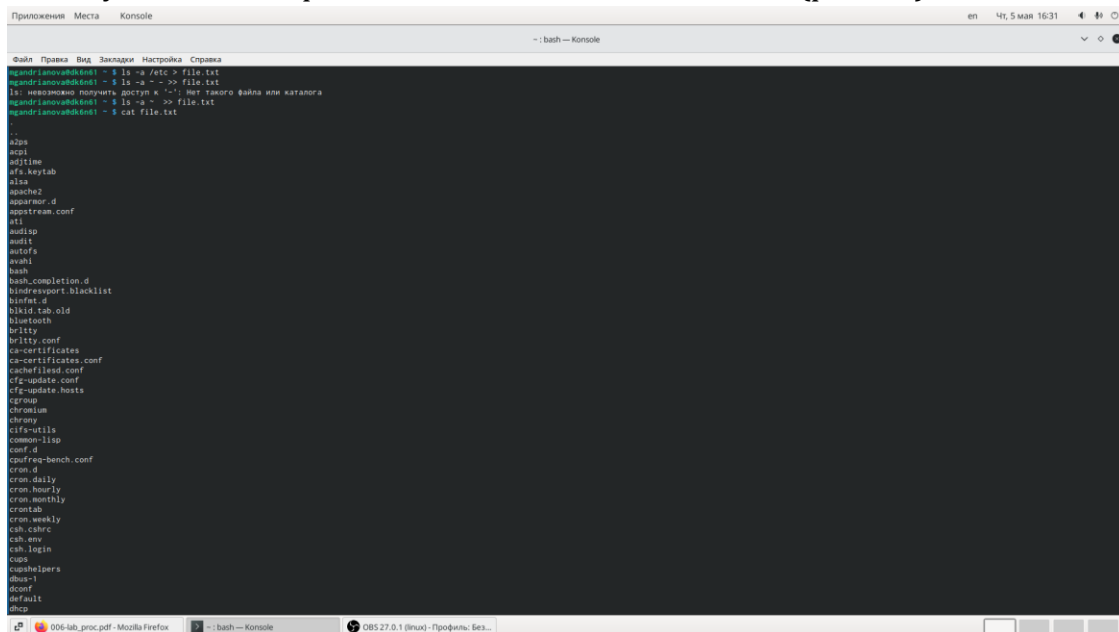
Андрианова Марина Георгиевна

Цель работы

Ознакомление с инструментами поиска файлов и фильтрации текстовых данных. Приобретение практических навыков: по управлению процессами (и заданиями), по проверке использования диска и обслуживанию файловых систем.

Выполнение лабораторной работы

1. Осуществим вход в систему,используя соответствующее имя пользователя.
2. Запишем в файл file.txt названия файлов,содержащихся в каталоге /etc. Для этого введём команду “ls -a /etc > file.txt”. Допишем в этот же файл названия файлов,содержащихся в нашем домашнем каталоге с помощью команды “ls -a ~ >> file.txt”. Затем просмотрим файл с помощью команды “cat file.txt”, чтобы убедиться в правильном выполнении действий(рис.1-2).



```
Приложение Места Консоль
en Чт, 5 мая 16:31
~: bash -- Konsole

Файл Правка Вид Закладки Настройка Справка
mrandriano@kali:~$ ls -a /etc > file.txt
mrandriano@kali:~$ ls -a ~ >> file.txt
ls: невозможно получить доступ к '~': нет такого файла или каталога
mrandriano@kali:~$ ls -a ~ >> file.txt
mrandriano@kali:~$ cat file.txt
.
..
c
a2ps
acpi
adjtime
afs.keytab
alsa
alsa2
asacme2
apparmor.d
appstream.conf
ati
audiisp
audit
autofs
avahi
bash
bash_completion.d
bindresport.blacklist
binfmt.d
blkid.tab.old
bluetooth
brltty
brltty.conf
ca-certificates
ca-certificates.conf
cachefilesd.conf
cfg-update.conf
cfg-update.hosts
cgroup
chromium
cifs-utils
common-lisp
conf.d
cpufreq-bench.conf
cron.d
cron.daily
cron.hourly
cron.monthly
crontab
cron.weekly
csh.cshrc
csh.env
csh.login
cups
cupsfilters
dbus-1
default
dhcp
```

Рис.1

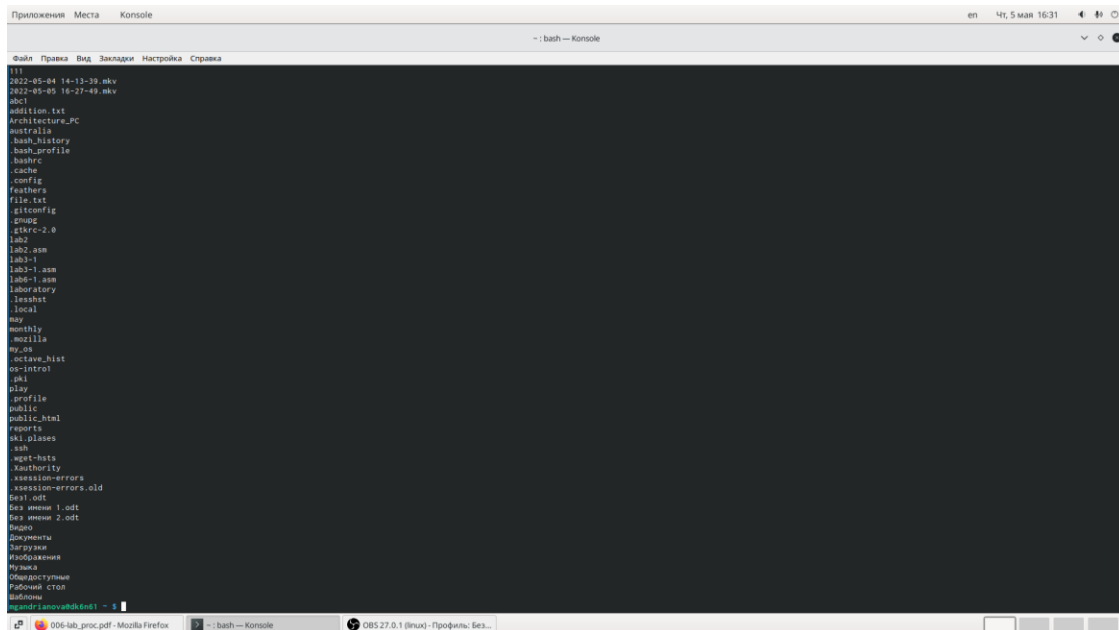


Рис.2

- Выведем имена всех файлов из file.txt, имеющих расширение .conf, после чего запишем их в новый текстовый файл conf.txt, используя команду “`grep -e '.conf$' file.txt > conf.txt`”. С помощью команды “`cat conf.txt`” проверяем правильность выполненных действий (рис.3-4).

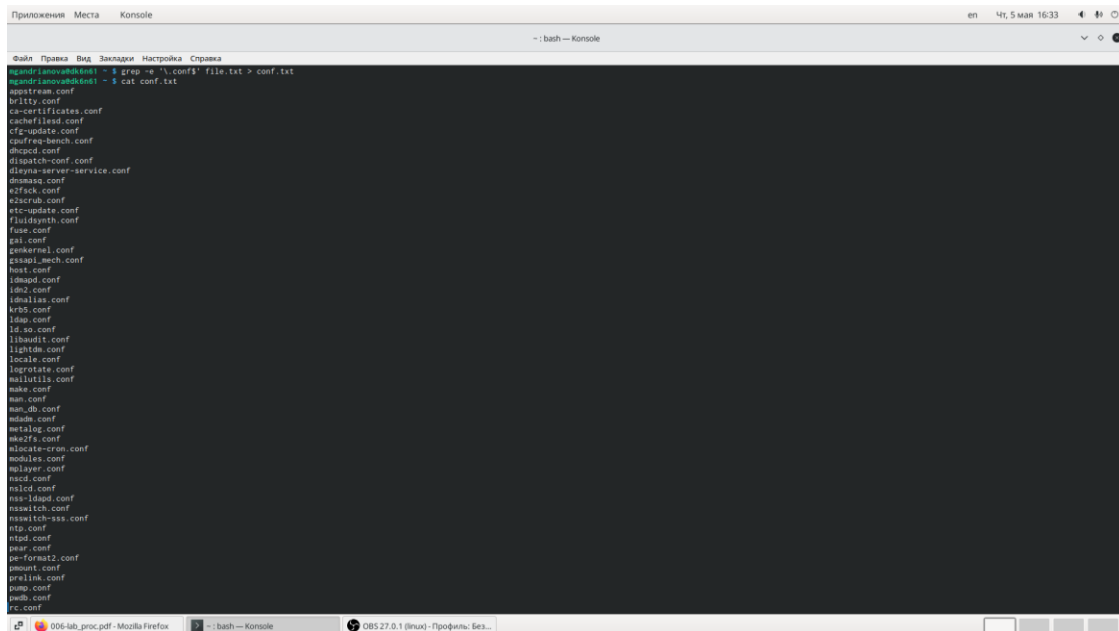
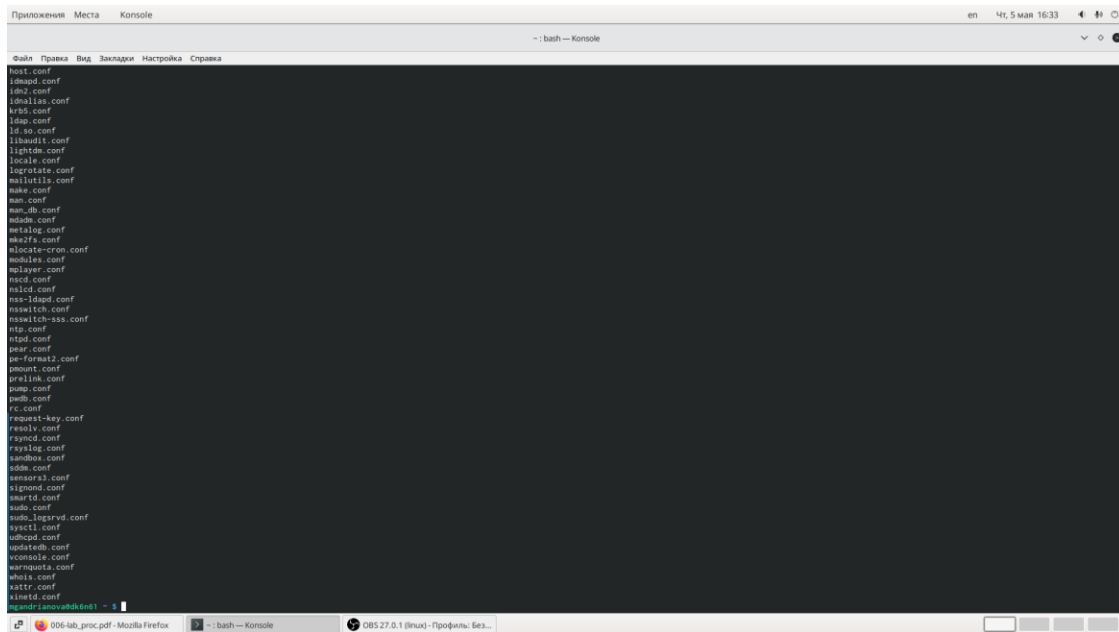


Рис.3



Puc.4

4. Определить, какие файлы в нашем домашнем каталоге имеют имена, начинающиеся с символа с, можно несколькими командами (рис.5). (Опция `maxdepth 1` необходима для того, чтобы файлы находились строго только в домашнем каталоге):

```
find ~ -maxdepth 1 -name "c*" -print
ls ~/c*
ls -a ~ | grep c*
```

```
mgandrianova@dk6n61 ~ $ ls ~/c*
/afs/.dk.sci.pfu.edu.ru/home/m/g/mgandrianova/conf.txt
mgandrianova@dk6n61 ~ $ find ~ -maxdepth 1 -name "c*" -print
/afs/.dk.sci.pfu.edu.ru/home/m/g/mgandrianova/conf.txt
mgandrianova@dk6n61 ~ $ ls -a ~ | grep c*
conf.txt
```

Puc.5

5. Чтобы вывести на экран (по странично) имена файлов из каталога /etc, начинающиеся с символа h, я использовала команду “find /etc -maxdepth 1 -name”h*” | less” (рис.6-7).

```
mgandrianova@dk6n61 ~ $ find /etc -maxdepth 1 -name "h*" | less
```

Рис.6

```
/etc/hosts
/etc/hotplug.d
/etc/hsqldb
/etc/httpd
/etc/host.conf
/etc/htdig
/etc/hotplug
/etc/hostname
/etc/hal
/etc/harbour.cfg
/etc/hosts.allow
/etc/highlight
/etc/harbour
lines 1-13/13 (END)
```

Рис.7

6. Запустим в фоновом режиме процесс, который будет записывать в файл ~/logfile файлы, имена которых начинаются с log, используя команду “find / -name “log*” > logfile& “. (рис.8).

```
mgandrianova@dk6n61 ~ $ find / -name "log*" > logfile&
[1] 10328
```

Рис.8

Командой “cat logfile” проверяю выполненные действия (рис.9-10)

```

mgandrianova@dk6n61 ~ $ cat logfile
/opt/idea-community-2021.1.3/lib/log4j.jar
/opt/idea-community-2021.1.3/license/log4j_license.txt
/opt/idea-community-2021.1.3/plugins/textmate/lib/bundles/log
/opt/idea-community-2021.1.3/plugins/textmate/lib/bundles/log/syntaxes/log.tmLanguage.json
/opt/idea-community-2021.1.3/plugins/textmate/lib/bundles/git/src/log.ts
/opt/idea-community-2021.1.3/plugins/maven/lib/maven3/conf/logging
/opt/idea-community-2021.1.3/bin/log.xml
/opt/vscode/resources/app/extensions/ms-vscode.js-debug/resources/logo.png
/opt/vscode/resources/app/extensions/ms-vscode.js-debug/resources/readme/logo-with-text.png
/opt/vscode/resources/app/extensions/ms-vscode.js-debug/resources/logo.svg
/opt/vscode/resources/app/extensions/ms-vscode.js-debug-companion/resources/logo.png
/opt/vscode/resources/app/extensions/log
/opt/vscode/resources/app/extensions/log/syntaxes/log.tmLanguage.json
/opt/vscode/resources/app/extensions/php-language-features/icons/logo.png
/opt/vscode/resources/app/extensions/ms-vscode.node-debug2/node_modules/vscode-debugadapter/lib/loggingDebugSession.js
/opt/vscode/resources/app/extensions/ms-vscode.node-debug2/node_modules/vscode-debugadapter/lib/logger.js
/opt/scilab/thirdparty/java/lib/logging.properties
/opt/scilab/thirdparty/docbook/log
/opt/scilab/include/scilab/logicalopexp.hxx
/opt/scilab/share/scilab/etc/logging.properties
/opt/scilab/share/scilab/modules/tclsci/tcl/tk8.5/images/logo64.gif
/opt/scilab/share/scilab/modules/tclsci/tcl/tk8.5/images/logo100.gif
/opt/scilab/share/scilab/modules/tclsci/tcl/tk8.5/images/logoMed.gif
/opt/scilab/share/scilab/modules/tclsci/tcl/tk8.5/images/logoLarge.gif
/opt/scilab/share/scilab/modules/tclsci/tcl/tk8.5/images/logo.eps
/opt/scilab/share/scilab/modules/elementary_functions/macros/log2.bin
/opt/scilab/share/scilab/modules/elementary_functions/macros/logspace.bin
/opt/scilab/share/scilab/modules/elementary_functions/macros/logm.bin
/opt/scilab/share/scilab/modules/elementary_functions/macros/log2.sci
/opt/scilab/share/scilab/modules/elementary_functions/macros/logspace.sci
/opt/scilab/share/scilab/modules/elementary_functions/macros/logm.sci
/opt/scilab/share/scilab/modules/elementary_functions/tests/unit_tests/logspace.dia.ref
/opt/scilab/share/scilab/modules/elementary_functions/tests/unit_tests/log2.tst
/opt/scilab/share/scilab/modules/elementary_functions/tests/unit_tests/loglp.dia.ref
/opt/scilab/share/scilab/modules/elementary_functions/tests/unit_tests/logspace.tst
/opt/scilab/share/scilab/modules/elementary_functions/tests/unit_tests/loglp.tst
/opt/scilab/share/scilab/modules/demo_tools/images/logo_scilab.png
/opt/scilab/share/scilab/modules/xcos/tests/unit_tests/Integer/logic.zcos
/opt/scilab/share/scilab/modules/xcos/tests/unit_tests/Integer/logic.dia.ref
/opt/scilab/share/scilab/modules/xcos/tests/unit_tests/Integer/logic.tst
/opt/scilab/share/scilab/modules/xcos/demos/Electrical/logic_and.dem.sce
/opt/scilab/share/scilab/modules/xcos/demos/Electrical/logic_nor.dem.sce
/opt/android-studio/lib/log4j.jar
/opt/android-studio/license/log4j_license.txt
/opt/android-studio/plugins/android/lib/templates/activities/LoginActivity/root/src/app_package/ui/login
/opt/android-studio/bin/log.xml
/opt/android-studio/bin/lldb/lib/python2.7/logging
/opt/android-studio/bin/lldb/lib/python2.7/hotshot/log.py
/opt/ns2/lib/tk8.5/images/logo64.gif
/opt/ns2/lib/tk8.5/images/logo100.gif
/opt/ns2/lib/tk8.5/images/logoMed.gif
/opt/ns2/lib/tk8.5/images/logoLarge.gif

```

Puc.9

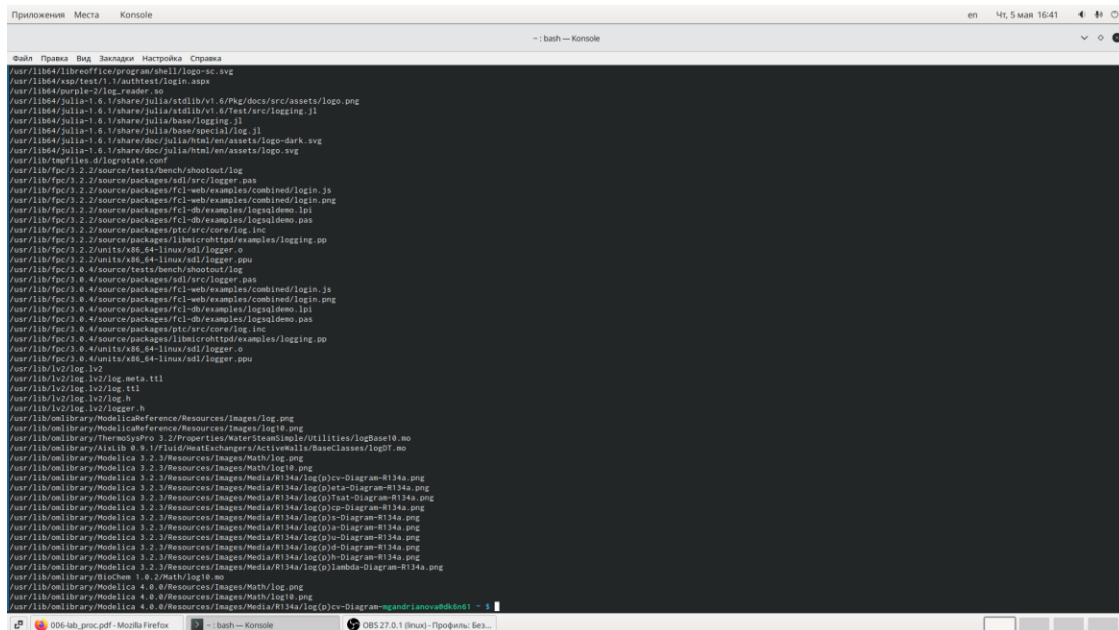


Рис.10

- Удалим файл ~/logfile с помощью команды(рис.11):
`rm logfile`

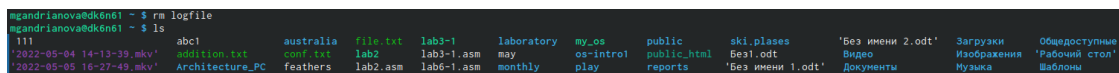


Рис.11

- Запустим из консоли в фоновом режиме редактор gedit с помощью команды “`gedit&`”(рис.12). Появится окно редактора(рис.13).

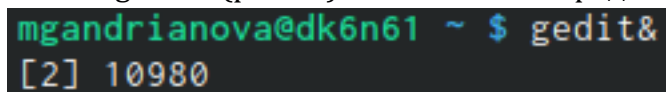


Рис.12

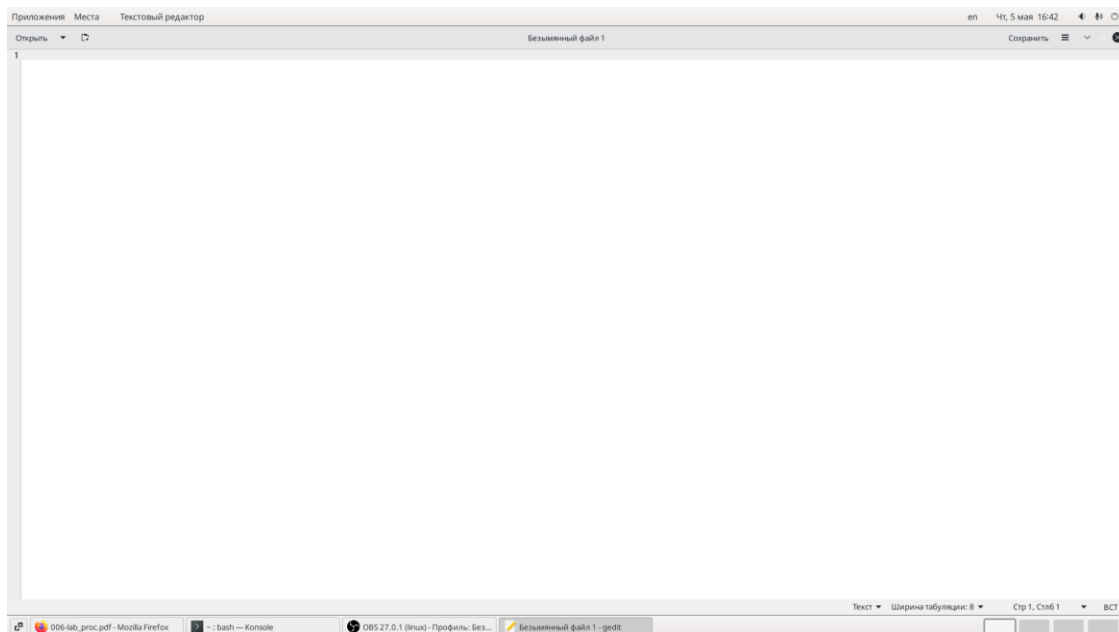


Рис.13

9. Чтобы определить идентификатор процесса gedit, используем команду "ps | grep -i "gedit" ". Наш процесс имеет PID 10980. (рис.14). Определить идентификатор процесса можно и другими командами (рис.14):

```
pgrep gedit  
pidof gedit
```

```
mgandrianova@dk6n61 ~ $ ps | grep -i "gedit"  
10980 pts/0    00:00:02 gedit  
[3]+  Завершён      gedit  
mgandrianova@dk6n61 ~ $ pgrep gedit  
10980  
mgandrianova@dk6n61 ~ $ pidof gedit  
10980
```

Рис.14

10. Прочитав справку (man) команды kill с помощью команды "man kill" (рис.15-17), используем её для завершения процесса gedit (команда "kill %10980") (рис.18).

```
mgandrianova@dk6n61 ~ $ man kill
```

Рис.15

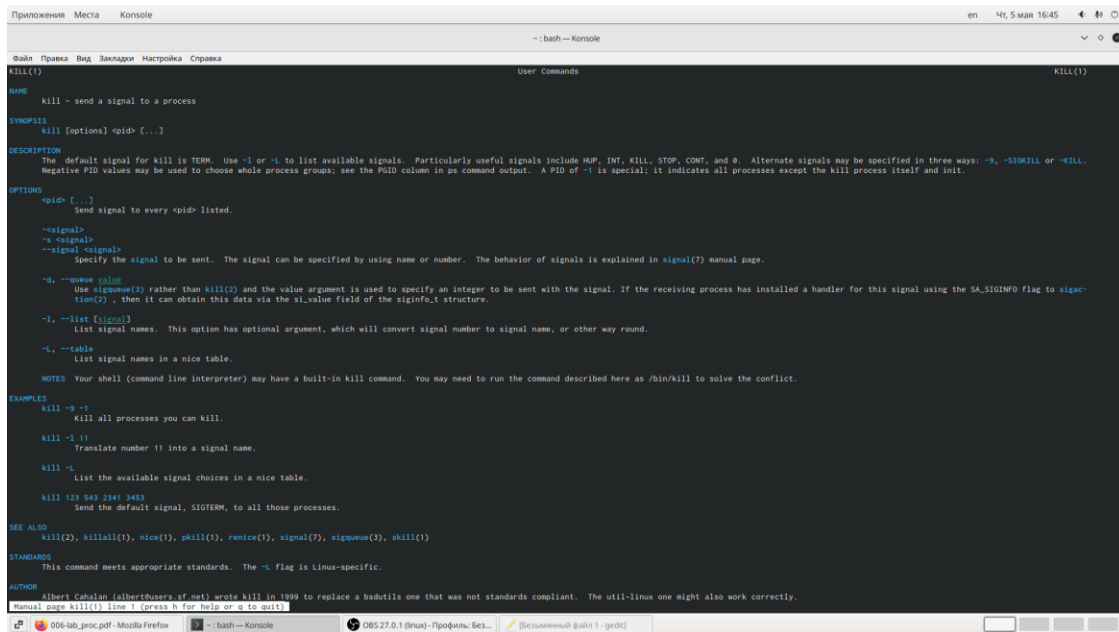


Рис.16



Рис.17

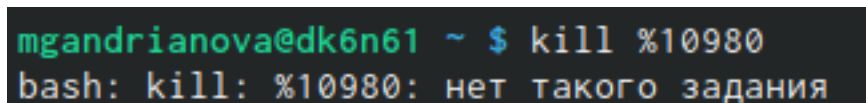
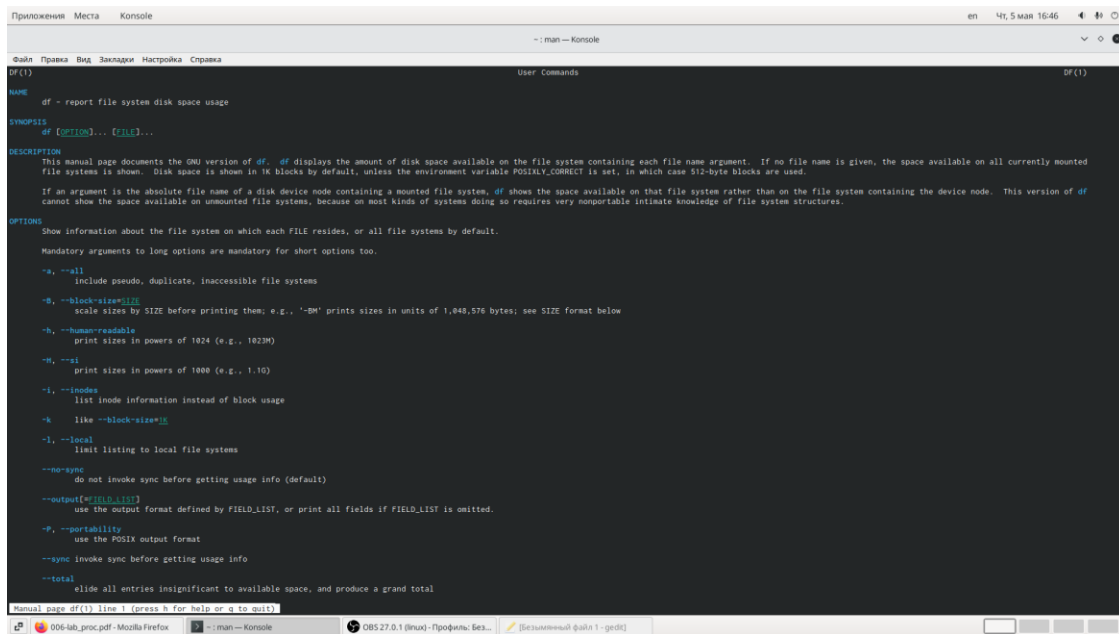


Рис.18

11. Выполним команды `df`(рис.22) и `du`(рис.23-24),предварительно получив более подробную информацию об этих командах(рис.20-21),с помощью команд “`man df`”(рис.19) и “`man du`”(рис.19).

```
mgandrianova@dk6n61 ~ $ man df
mgandrianova@dk6n61 ~ $ man du
```

Puc.19



The screenshot shows a terminal window with the title bar "Приложения Места Консоль". The terminal content displays the man page for the 'df' command. The page includes sections for NAME, SYNOPSIS, DESCRIPTION, and OPTIONS. The OPTIONS section lists various flags like --all, --block-size=SIZE, --human-readable, --si, --inodes, --k, --local, --no-sync, --output=FIELD_LIST, --portability, --sync, and --total, each with a brief description of its function.

```
df(1)
NAME
  df - report file system disk space usage

SYNOPSIS
  df [OPTION]... [FILE]...

DESCRIPTION
  This manual page documents the GNU version of df.  df displays the amount of disk space available on the file system containing each file name argument.  If no file name is given, the space available on all currently mounted file systems is shown.  Disk space is shown in 1K blocks by default, unless the environment variable POSIXLY_CORRECT is set, in which case 512-byte blocks are used.

  If an argument is the absolute file name of a disk device node containing a mounted file system, df shows the space available on that file system rather than on the file system containing the device node.  This version of df cannot show the space available on unmounted file systems, because on most kinds of systems doing so requires very nonportable intimate knowledge of file system structures.

OPTIONS
  Show information about the file system on which each FILE resides, or all file systems by default.

  Mandatory arguments to long options are mandatory for short options too.

  -a, --all
      include pseudo, duplicate, inaccessible file systems

  -B, --block-size=SIZE
      scale sizes by SIZE before printing them; e.g., '-BM' prints sizes in units of 1,048,576 bytes; see SIZE format below

  -h, --human-readable
      print sizes in powers of 1024 (e.g., 1023M)

  -m, --si
      print sizes in powers of 1000 (e.g., 1.1G)

  -i, --inodes
      list inode information instead of block usage

  -k, --block-size=K
      like --block-size=1K

  -l, --local
      limit listing to local file systems

  --no-sync
      do not invoke sync before getting usage info (default)

  --output=FIELD_LIST
      use the output format defined by FIELD_LIST, or print all fields if FIELD_LIST is omitted.

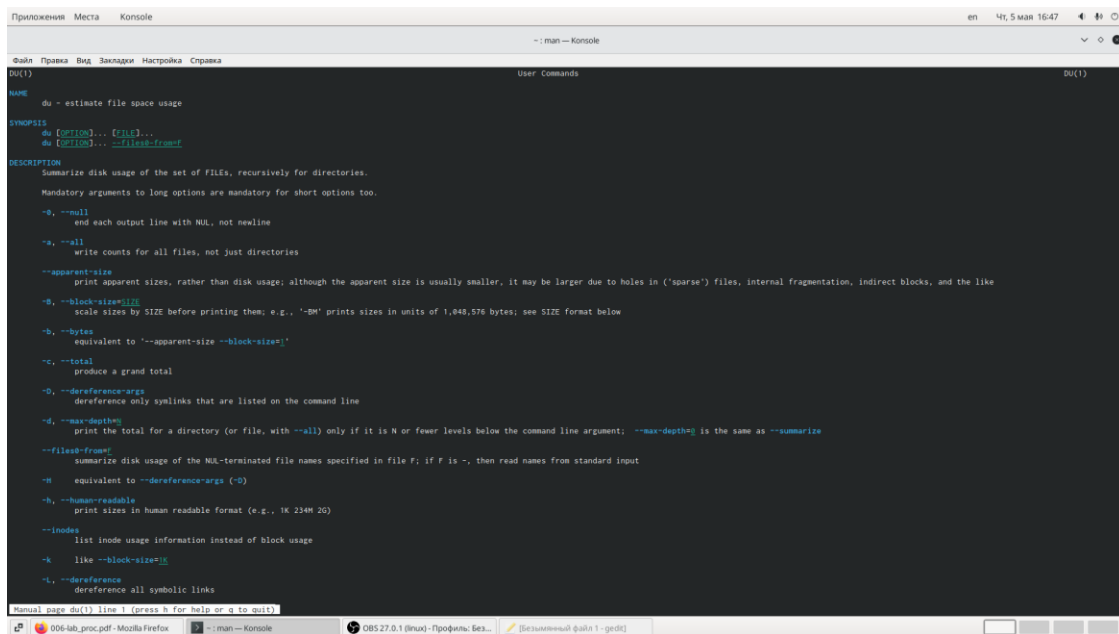
  -P, --portability
      use the POSIX output format

  --sync
      invoke sync before getting usage info

  --total
      elide all entries insignificant to available space, and produce a grand total

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

Puc.20



The screenshot shows a terminal window with the title bar "Приложения Места Консоль". The terminal content displays the man page for the 'du' command. The page includes sections for NAME, SYNOPSIS, DESCRIPTION, and OPTIONS. The OPTIONS section lists various flags like --null, --all, --apparent-size, --block-size=SIZE, --bytes, --total, --dereference-args, --max-depth, --files0-from=F, --inodes, --k, and --dereference, each with a brief description of its function.

```
du(1)
NAME
  du - estimate file space usage

SYNOPSIS
  du [OPTION]... [FILE]...
  du [OPTION]... --files0-from=F

DESCRIPTION
  Summarize disk usage of the set of FILES, recursively for directories.

  Mandatory arguments to long options are mandatory for short options too.

  -0, --null
      end each output line with NUL, not newline

  -a, --all
      write counts for all files, not just directories

  --apparent-size
      print apparent sizes, rather than disk usage; although the apparent size is usually smaller, it may be larger due to holes in ('sparse') files, internal fragmentation, indirect blocks, and the like

  -B, --block-size=SIZE
      scale sizes by SIZE before printing them; e.g., '-BM' prints sizes in units of 1,048,576 bytes; see SIZE format below

  -b, --bytes
      equivalent to '--apparent-size --block-size=1'

  -c, --total
      produce a grand total

  -d, --dereference-args
      dereference only symlinks that are listed on the command line

  -d, --max-depth=N
      print the total for a directory (or file, with --all) only if it is N or fewer levels below the command line argument; --max-depth=0 is the same as --summarize

  --files0-from=F
      summarize disk usage of the NUL-terminated file names specified in file F; if F is -, then read names from standard input

  -H, --dereference
      equivalent to --dereference-args (-d)

  -h, --human-readable
      print sizes in human readable format (e.g., 1K 234M 2G)

  --inodes
      list inode usage information instead of block usage

  -k, --block-size=K
      like --block-size=1K

  -L, --dereference
      dereference all symbolic links

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

Puc.21

```

mgandrianova@k6n61 ~ $ df
Файловая система      1К-блоков  Использовано  Доступно  Использовано%  Смонтировано в
none                   4000180      36672        3963508         1% /run
udev                   3890156         0        3890156         0% /dev
tmpfs                   4000180      157744       3842436         4% /dev/shm
/dev/sda8              491812356    96408672    370397940        21% /
/dev/sda6              91557952     434636     86449372         1% /var/cache/openafs
tmpfs                   4000180      77676       3922504         2% /tmp
mark.sci.pfu.edu.ru:/com/lib/portage 1048320000 504344576 543975424        49% /com/lib/portage
mark.sci.pfu.edu.ru:/usr/local/share/portage 18350080 5858816 11009792        35% /usr/local/share/portage
mark.sci.pfu.edu.ru:/usr/portage 18350080 5858816 11009792        35% /usr/portage
AFS                    2147483647      0     2147483647         0% /afs
tmpfs                   800036         232       799804         1% /run/user/4141
mark.sci.pfu.edu.ru:/usr/portage 18350080 5858816 11009792        35% /usr/portage
/dev/sdb1              7557136     247632     7309504         4% /run/media/mgandrianova/USB_DISK

```

Puc.22

```

Приложения Места Консоль
en 4ч, 5 мин 16:47
~: bash -- Konsole

Файл Правка Вид Залочки Настройка Справка
mgandrianova@k6n61 ~ $ du
2 ./public/public_html
4 ./public
2 ./local/share/keyrings
3 ./local/share/gnome-shell
2 ./local/share/evolution/addressbook/trash
2 ./local/share/evolution/addressbook/system/photos
88 ./local/share/evolution/addressbook/system
92 ./local/share/evolution/addressbook
2 ./local/share/evolution/calendar/trash
3 ./local/share/evolution/calendar/system
7 ./local/share/evolution/calendar
2 ./local/share/evolution/mail/trash
4 ./local/share/evolution/mail
2 ./local/share/evolution/memos/trash
4 ./local/share/evolution/memos
2 ./local/share/evolution/tasks/trash
3 ./local/share/evolution/tasks/system
7 ./local/share/evolution/tasks
116 ./local/share/evolution
319 ./local/share/gnome-metad
2 ./local/share/flatpak/db
4 ./local/share/flatpak
3 ./local/share/applications
2 ./local/share/sounds
4 ./local/share/telepathy/mission-control
6 ./local/share/telepathy
2 ./local/share/gnome-settings-daemon
2274 ./local/share/haloo
4 ./local/share/icc
850 ./local/share/tracker/data
952 ./local/share/tracker
2 ./local/share/ksmone
2 ./local/share/mc/mcedit
7 ./local/share/mc
2 ./local/share/webkitgtk/deviceidhashalts/1
4 ./local/share/webkitgtk/deviceidhashalts
2 ./local/share/webkitgtk/databases/indexeddb/v1
6 ./local/share/webkitgtk/databases/indexeddb
6 ./local/share/webkitgtk/databases
2 ./local/share/webkitgtk/localstorage
2 ./local/share/webkitgtk/serviceworkers
28 ./local/share/webkitgtk
2 ./local/share/nautilus/scripts
321 ./local/share/nautilus/tags
329 ./local/share/nautilus
12 ./local/share/recentdocuments
2 ./local/share/nautilusmanager/resources/test-backup
107 ./local/share/nautilusmanager/resources/working-backup
106 ./local/share/nautilusmanager/resources
162 ./local/share/nautilusmanager
22 ./local/share/http
1 ./local/share/kscreen/outputs
6 ./local/share/kscreen
7 ./local/share/kscreen/lecard

```

Puc.23

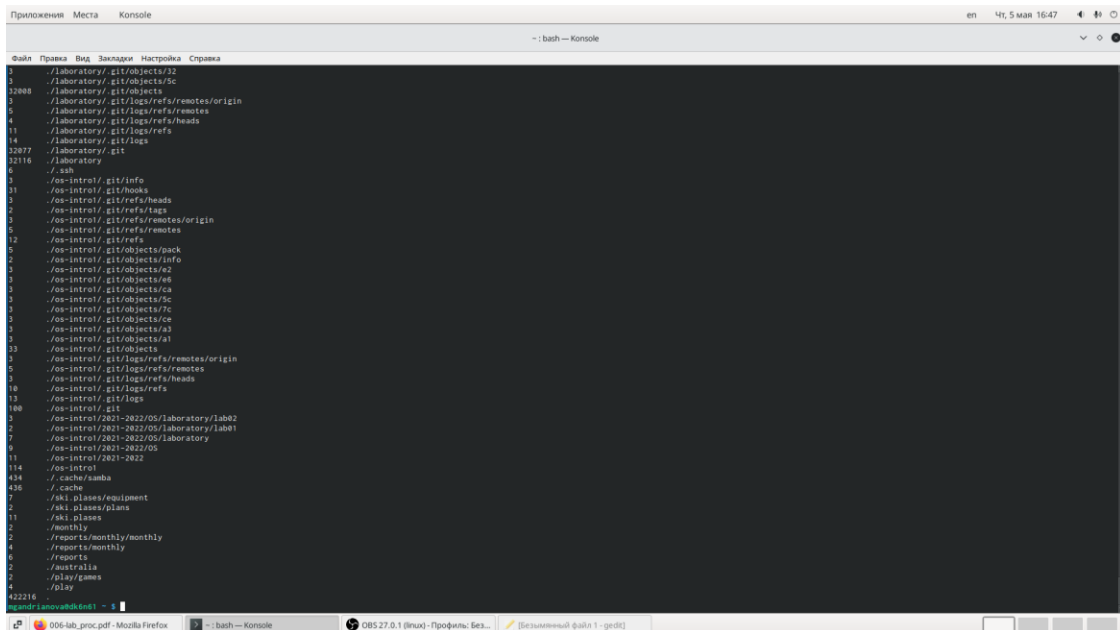


Рис.24

12. Воспользуемся справкой команды `find`(рис.26), введя команду “`man find`”(рис.25).Затем выведем имена всех директорий,имеющихся в нашем домашнем каталоге, с помощью команды “`find ~ -type d`” (рис.27-28).



Рис.25

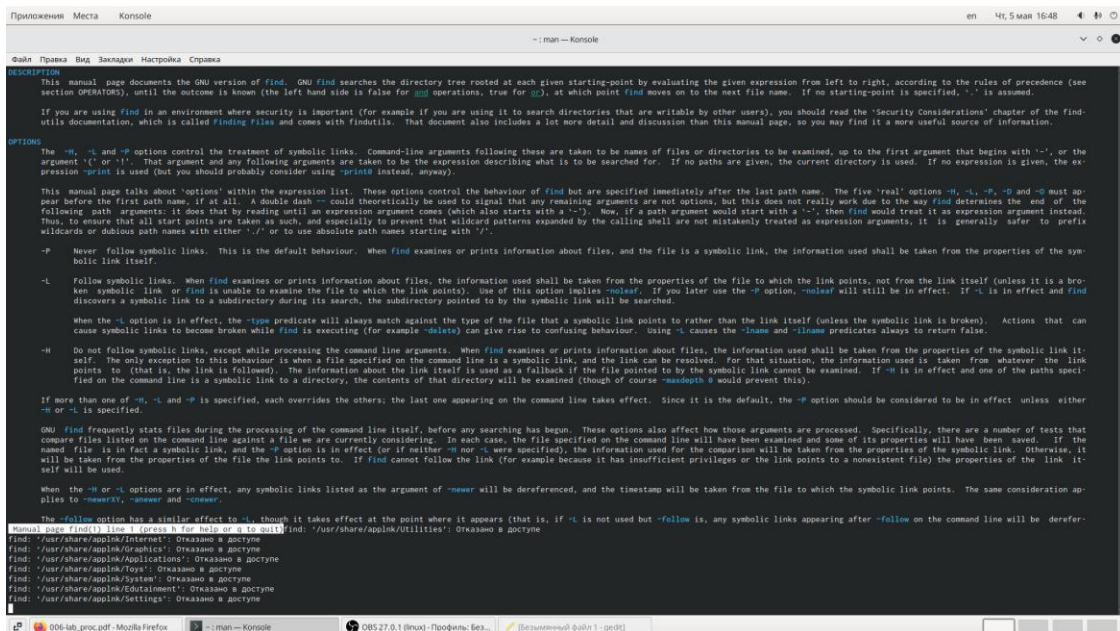


Рис.26

- stdin —стандартный поток ввода(по умолчанию: клавиатура),файловый дескриптор 0;
- stdout —стандартный поток вывода (по умолчанию: консоль),файловый дескриптор 1;
- stderr —стандартный поток вывод сообщений об ошибках (по умолчанию: консоль),файловый дескриптор 2.

Большинство используемых в консоли команд и программ записывают результаты своей работы в стандартный поток вывода stdout.

2. ‘>’ означает перенаправление вывода (stdout) в файл. ‘>>’ означает перенаправление вывода в файл и открытие файла в режиме добавления(данные добавляются в конец файла).
3. Конвейер (pipe) служит для объединения простых команд или утилит в цепочки,в которых результат работы предыдущей команды передаётся последующей. Синтаксис следующий:

команда 1 | команда 2

означает, что вывод команды 1 передаётся на ввод команде 2

4. Процесс рассматривается операционной системой как заявка на потребление всех видов ресурсов, кроме одного – процессорного времени. Этот последний важнейший ресурс распределяется операционной системой между другими единицами работы – потоками, которые и получили свое название благодаря тому, что они представляют собой последовательности (потоки выполнения) команд. Процесс – это выполнение программы. Он считается активной сущностью и реализует действия, указанные в программе. Программа представляет собой статический набор команд, а процесс это набор ресурсов и данных, использующихся при выполнении программы.
5. pid: идентификатор процесса (PID) процесса (processID), к которому вызывают метод gid: идентификатор группы UNIX, в котором работает программа.
6. Любую выполняющуюся в консоли команду или внешнюю программу можно запустить в фоновом режиме. Для этого следует в конце имени команды указать знак амперсанда &. Запущенные фоном программы называются задачами (jobs). Ими можно управлять с помощью команды jobs, которая выводит список запущенных в данный момент задач.
7. top – это консольная программа, которая показывает список работающих процессов в системе. Программа в реальном времени отсортирует запущенные процессы по их нагрузке на процессор.
htop – это продвинутый консольный мониторинг процессов. Утилита выводит постоянно меняющийся список системных процессов, который сортируется в зависимости от нагрузки на ЦПУ. Если делать сравнение с top, то htop показывает абсолютно все процессы в системе, время их непрерывного использования, загрузку процессоров и расход оперативной памяти.
8. find – это команда для поиска файлов и каталогов на основе специальных условий. Ее можно использовать в различных обстоятельствах, например, для

поиска файлов по разрешениям, владельцам, группам, типу, размеру и другим подобным критериям. Команда find имеет такой синтаксис:
find[папка][параметры] критерий шаблон [действие]

Папка – каталог в котором будем искать

Параметры – дополнительные параметры, например, глубина поиска, и т д.

Критерий – по какому критерию будем искать: имя, дата создания, права, владелец и т д.

Шаблон – непосредственно значение по которому будем отбирать файлы. Основные параметры:

-P никогда не открывать символические ссылки

-L - получает информацию о файлах по символическим ссылкам. Важно для дальнейшей обработки, чтобы обрабатывалась не ссылка, а сам файл.

-maxdepth - максимальная глубина поиска по подкаталогам, для поиска только в текущем каталоге установите 1.

-depth - искать сначала в текущем каталоге, а потом в подкаталогах

-mount искать файлы только в этой файловой системе.

-version - показать версию утилиты

find -print - выводить полные имена файлов

-type f - искать только файлы

-type d - поиск папки в Linux

Основные критерии:

-name - поиск файлов по имени

-perm - поиск файлов в Linux по режиму доступа

-user - поиск файлов по владельцу

-group - поиск по группе

-mtime - поиск по времени модификации файла

-atime - поиск файлов по дате последнего чтения

-nogroup - поиск файлов, не принадлежащих ни одной группе

-nouser - поиск файлов без владельцев

-newer - найти файлы новее чем указанный

-size - поиск файлов в Linux по их размеру

Примеры: `find~ -type d` поиск директорий в домашнем каталоге

`find~ -type f -name ".*"` поиск скрытых файлов в домашнем каталоге

9. Файл по его содержимому можно найти с помощью команды `grep`: «`grep -r`» слово/выражение, которое нужно найти».

10. Утилита `df`, позволяет проанализировать свободное пространство на всех подключенных к системе разделах.

11. При выполнении команды `du` (без указания папки и опции) можно получить все файлы и папки текущей директории с их размерами. Для домашнего каталога: `du ~/`

12. Основные сигналы (каждый сигнал имеет свой номер), которые используются для завершения процесса:

- **SIGINT**–самый безобидный сигнал завершения, означает `Interrupt`. Он отправляется процессу, запущенному из терминала с помощью сочетания клавиш `Ctrl+C`. Процесс правильно завершает все свои действия и возвращает управление;
- **SIGQUIT**–это еще один сигнал, который отправляется с помощью сочетания клавиш, программе, запущенной в терминале. Он сообщает ей что нужно завершиться и программа может выполнить корректное завершение или проигнорировать сигнал. В отличие от предыдущего, она генерирует дампы памяти. Сочетание клавиш `Ctrl+/\`;
- **SIGHUP**–сообщает процессу, что соединение с управляющим терминалом разорвано, отправляется, в основном, системой при разрыве соединения с интернетом;
- **SIGTERM**–немедленно завершает процесс, но обрабатывается программой, поэтому позволяет ей завершить дочерние процессы и освободить все ресурсы;
- **SIGKILL**–тоже немедленно завершает процесс, но, в отличие от предыдущего варианта, он не передается самому процессу, а обрабатывается ядром. Поэтому ресурсы и дочерние процессы остаются запущенными. Также для передачи сигналов процессам в Linux используется утилита `kill`, её синтаксис:

`kill [-сигнал] [pid_процесса]` (PID – уникальный идентификатор процесса). Сигнал представляет собой один из выше перечисленных сигналов для завершения процесса. Перед тем, как выполнить остановку процесса, нужно определить его PID. Для этого используют команды `ps` и `grep`.

Команда `ps` предназначена для вывода списка активных процессов в системе и информации о них.

Команда `grep` запускается одновременно с `ps` и будет выполнять поиск по результатам команды `ps`.

Утилита `kill` – это оболочка для `kill`, она ведет себя точно так же, и имеет тот же синтаксис, только в качестве идентификатора процесса ей нужно передать его имя.

`killall` работает аналогично двум предыдущим утилитам. Она тоже принимает имя процесса в качестве параметра и ищет его PID в директории `/proc`. Но эта утилита обнаружит все процессы с таким именем и завершит их.