

Отчет по лабораторной работе 6

Лабораторная работа 6

Репина Ангелина Олеговна

Список иллюстраций

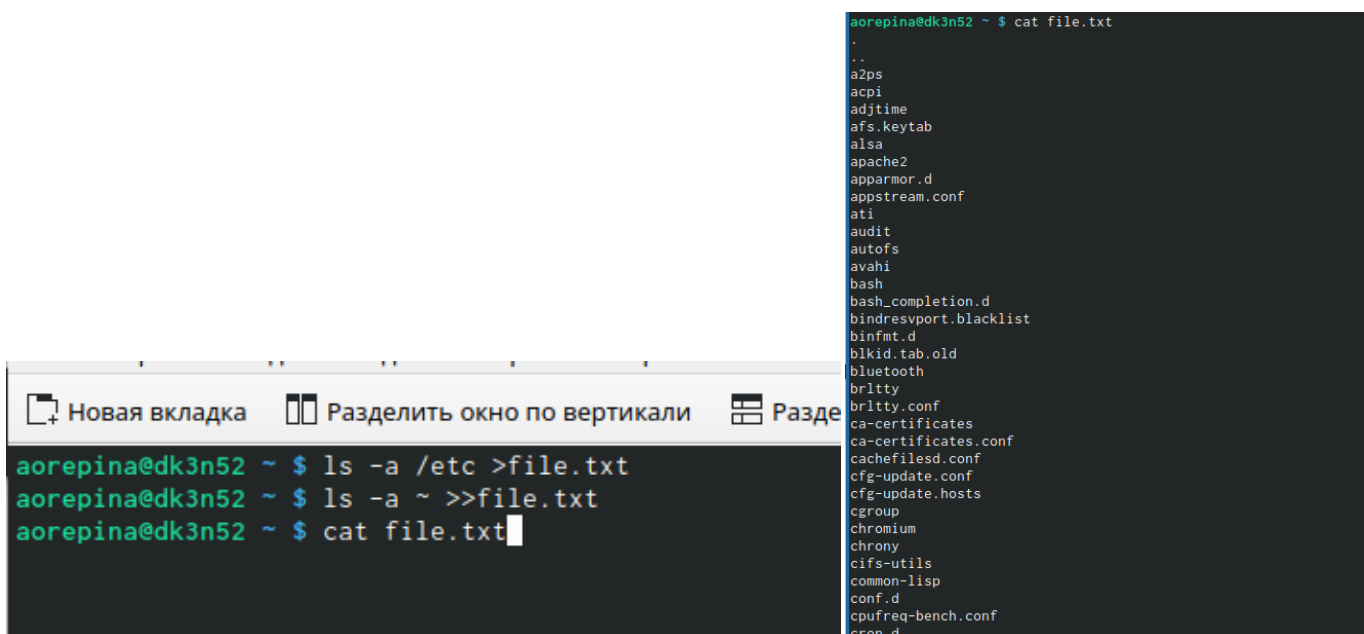
0.1	5	5
0.2	6	6
0.3	7	6
0.4	9	6
0.5	10	7

Цель работы

Цель работы - Ознакомиться с инструментами поиска файлов и фильтрации текстовых данных. Приобрести практические навыки: по управлению процессами (и заданиями), по проверке использования диска и обслуживанию файловых систем.

Выполнение лабораторной работы

1. Осуществила вход в систему, используя соответствующее имя пользователя
2. Записала в файл file.txt названия файлов, содержащихся в каталоге /etc. Дописала в этот же файл названия файлов, содержащихся в моем домашнем каталоге (рис.2, 2.1)



```
aorepina@dk3n52 ~ $ cat file.txt
:
:
a2ps
acpi
adjtime
afs.keytab
alsa
apache2
apparmor.d
appstream.conf
ati
audit
autofs
avahi
bash
bash_completion.d
bindresvport.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
chrony
cifs-utils
common-lisp
conf.d
cpufreq-bench.conf
cron.d
```

```
Новая вкладка  Разделить окно по вертикали  Разде
aorepina@dk3n52 ~ $ ls -a /etc >file.txt
aorepina@dk3n52 ~ $ ls -a ~ >>file.txt
aorepina@dk3n52 ~ $ cat file.txt
```

3. Вывела имена всех файлов из file.txt, имеющих расширение .conf, после чего записала их в новый текстовый файл conf.txt. (рис.3, 3.1)

```
aorepina@dk3n52 ~ $ grep -e '\.conf$' file.txt >conf.txt
aorepina@dk3n52 ~ $ cat conf.txt
appstream.conf
brlty.conf
ca-certificates.conf
cachefilesd.conf
cfg-update.conf
cpufreq-bench.conf
dhcpcd.conf
dispatch-conf.conf
dleyna-server-service.conf
dnsmasq.conf
dracut.conf
e2fsck.conf
e2scrub.conf
etc-update.conf
fluidsynth.conf
fuse.conf
gai.conf
genkernel.conf
gssapi_mech.conf
host.conf
idmapd.conf
idn2.conf
idnalias.conf
krb5.conf
ldap.conf
ld.so.conf
libaudit.conf
lightdm.conf
locale.conf
logrotate.conf
mailutils.conf

aorepina@dk3n52 ~ $ cat logfile
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/puppet/modules/pu
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/puppet/modules/pu
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/.git/modules/puppe
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/.git/modules/puppe
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/.git/modules/puppe
/afs/dk.sci.pfu.edu.ru/common/files/drupal/lamp/.git/logs
/afs/dk.sci.pfu.edu.ru/common/files/drupal/vagrant-proxyconf/.git
/afs/dk.sci.pfu.edu.ru/common/files/drupal/vagrant-proxyconf/spec
/afs/dk.sci.pfu.edu.ru/common/files/drupal/vagrant-proxyconf/lib/
aorepina@dk3n52 ~ $
```

4. Определила, какие файлы в вашем домашнем каталоге имеют имена, начинавшиеся с символа с. Предложила несколько вариантов, как это сделать. (варианты показаны на скриншотах) (рис.4, 5, 6)

```
aorepina@dk3n52 ~ $ find ~ -maxdepth 1 -name "c*" -print
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/pulse/cookie
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/gtk-3.0/assets/close-normal.svg
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/gtk-3.0/assets/close-active.svg
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/gtk-3.0/assets/close-hover.svg
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/gtk-3.0/assets/close-backdrop-normal.svg
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/gtk-3.0/assets/close-backdrop-active.svg
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/gtk-3.0/assets/close-backdrop-hover.svg
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/gtk-3.0/colors.css
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/kdeconnect/certificate.pem
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/kdeconnect/config
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/libreoffice/4/user/config
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/libreoffice/4/user/extensions/shared/registry/com.sun.star.comp.deployment.component.PackageRegistryBackend
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/libreoffice/4/user/extensions/shared/registry/com.sun.star.comp.deployment.configuration.PackageRegistryBackend
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/libreoffice/4/user/extensions/shared/registry/com.sun.star.comp.deployment.executable.PackageRegistryBackend
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/libreoffice/4/user/extensions/shared/registry/com.sun.star.comp.deployment.help.PackageRegistryBackend
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/libreoffice/4/user/extensions/shared/registry/com.sun.star.comp.deployment.script.PackageRegistryBackend
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/libreoffice/4/user/extensions/shared/registry/com.sun.star.comp.deployment.sfwk.PackageRegistryBackend
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/libreoffice/4/user/extensions/shared/registry/com.sun.star.comp.deployment.bundle.PackageRegistryBackend
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/.config/libreoffice/4/user/extensions/bundled/registry/com.sun.s
```

```
aorepina@dk3n52 ~ $ ls ~/c*
/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/conf.txt

/afs/.dk.sci.pfu.edu.ru/home/a/o/aorepina/course-directory-student-template:
config LICENSE Makefile package.json README.en.md README.git-flow.md README.md template
aorepina@dk3n52 ~ $
```

Рис. 0.1: 5

```

aorepina@dk3n52 ~ $ ls -a ~ | grep c*

```

Рис. 0.2: 6

5. Вывела на экран (по странично) имена файлов из каталога /etc, начинающиеся с символа h. (рис.7)

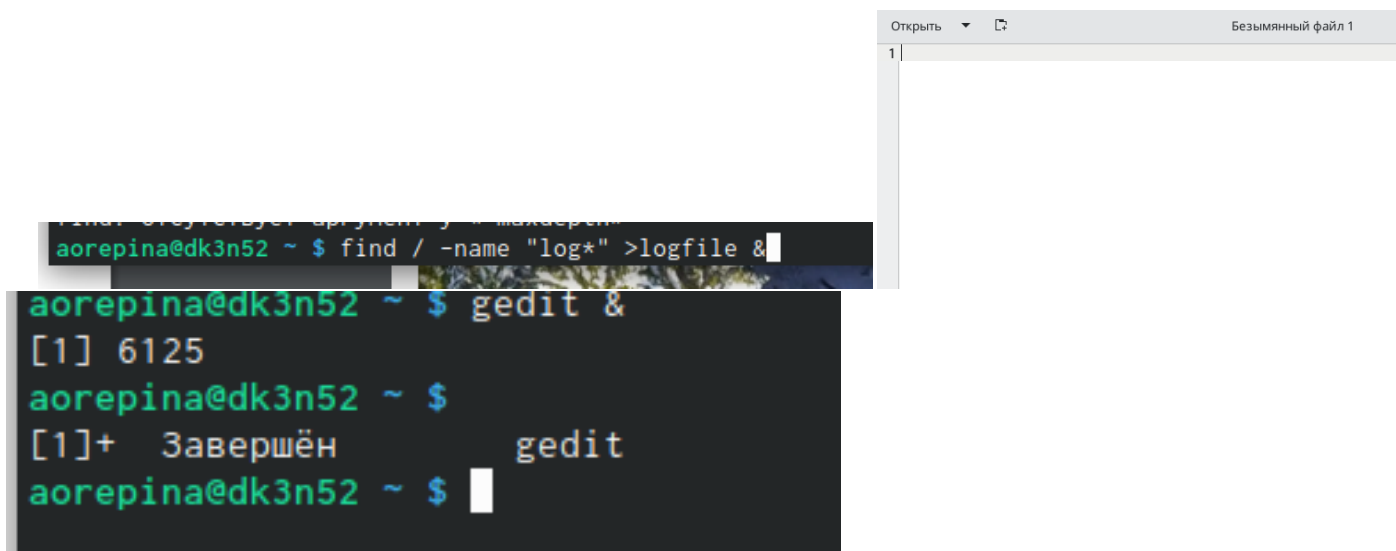
```

aorepina@dk3n52 ~ $ find /etc -maxdepth 1 -name "h*" | less

```

Рис. 0.3: 7

6. Запустила в фоновом режиме процесс, который будет записывать в файл ~/logfile файлы, имена которых начинаются с log. (рис.8, 8.1, 8.2)



The image shows a terminal window and a file manager window. The terminal window displays the following commands and output:

```

aorepina@dk3n52 ~ $ find / -name "log*" >logfile &
aorepina@dk3n52 ~ $ gedit &
[1] 6125
aorepina@dk3n52 ~ $
[1]+  Завершён      gedit
aorepina@dk3n52 ~ $

```

The file manager window shows a file named "Безымянный файл 1" (Unnamed file 1) with the content "1".

7. Удалила файл ~/logfile. (рис.9)

```

aorepina@dk3n52 ~ $ rm logfile

```

Рис. 0.4: 9

8. Запустила из консоли в фоновом режиме редактор gedit. (рис. 10)

```
aorepina@dk3n52 ~ $ gedit &
[1] 6125
aorepina@dk3n52 ~ $
```

Рис. 0.5: 10

9. Определила идентификатор процесса gedit, используя команду ps, конвейер и фильтр grep.
10. Прочитала справку (man) команды kill, после чего использовала её для завершения процесса gedit. (рис 11, 11.1)

```
aorepina@dk3n52 ~ $ man kill
```

```
KILL(1) User Commands
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 include HUP, INT, KILL, STOP, CONT, and 0. Alternatively, three ways: -9, -SIGKILL or -KILL. Negative PID values refer to process groups; see the PGID column in ps command output. A PID of -1 kills 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 name or number. The behavior of signals is explained in signal(7) manual page.

    -q, --queue <value>
        Use sigqueue(3) rather than kill(2) and the value argument to be sent with the signal. If the receiving process is not a member of the signal group, the signal is sent using the SA_SIGINFO flag to sigaction(2), then the si_value field of the siginfo_t structure.

    -l, --list [<signal>]
        List signal names. This option has optional argument to signal name, or other way round.

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

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

11. Выполнила команды df и du, предварительно получив более подробную информацию об этих командах, с помощью команды man. (рис.12, 12.1, 12.2, 12.3, 12.4)

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

DU(1)                                User Commands                                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 size of some files is known exactly, it may be larger due to holes in ('sparse') files, 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
Manual page du(1) line 1 (press h for help or q to quit)

aorepina@dk3n52 ~ $ df
df: /run/user/4268/doc: Операция не позволена
Файловая система      1К-блоков  Использовано  Доступно  Использовано%  Смонтировано
/                     3999736      20268        3979468         1% /run
udev                  10240         0           10240          0% /dev
tmpfs                 3999736      20484        3979252         1% /dev/shm
/dev/sda8             490892692    98076588     367810360       22% /
tmpfs                 3999740      38772        3960968         1% /tmp
/dev/sda6             91164400     215068       86275388         1% /var/cache/o
penafs
mark.sci.pfu.edu.ru:/com/lib/portage 1048320000 182834688 865485312 18% /com/lib/portage
mark.sci.pfu.edu.ru:/usr/local/share/portage 18350080 5778944 10560256 36% /usr/local/share/portage
mark.sci.pfu.edu.ru:/usr/portage 18350080 5778944 10560256 36% /usr/portage
AFS                   2147483647 0 2147483647 0% /afs
tmpfs                 799944       204         799740         1% /run/user/4268
mark.sci.pfu.edu.ru:/usr/portage 18350080 5778944 10560256 36% /usr/portage
aorepina@dk3n52 ~ $
aorepina@dk3n52 ~ $ man df
aorepina@dk3n52 ~ $ man du
aorepina@dk3n52 ~ $
```

12. Воспользовавшись справкой команды find, вывела имена всех директорий, имеющих в моем домашнем каталоге. (рис.13, 13.1, 13.2)

[illegible]

9

Выводы

В результате данной лабораторной работы я ознакомилась с инструментами поиска файлов и фильтрации текстовых данных, приобрела практические навыки по управлению процессами, по проверке использования дисков и обслуживания файловых систем.

Ответы на контрольные вопросы: 1). В системе по умолчанию открыто три специальных потока: `-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). `'>'` Перенаправление вывода в файл `'>'` 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: Ответы на контрольные вопросы: 1). В системе по умолчанию открыто три специальных потока: `-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). `'>'` Перенаправление

вывода в файл ‘»’ 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: команда1|команда2 (это означает, что вывод команды 1 передастся 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 stop, 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 дамп 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* (в канале) и будет выполнять поиск по результатам команды *ps*. Утилита *pkill* – 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. команда1|команда2 (это означает, что вывод команды 1 передается 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 `stop`, 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 `-typed` - 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 дамп 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 (в канале) и будет выполнять поиск по результатам команды ps. Утилита pkill – 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.