Отчёт лабораторной работы №7

Дисциплина: Операционные системы

Касьянов Даниил Владимирович

Содержание

1 Цель работы **5**

2 Выполнение лабораторной работы **6**

3 Контрольные вопросы **13**

4 Выводы **18**

5 Библиография **19**

List of Tables

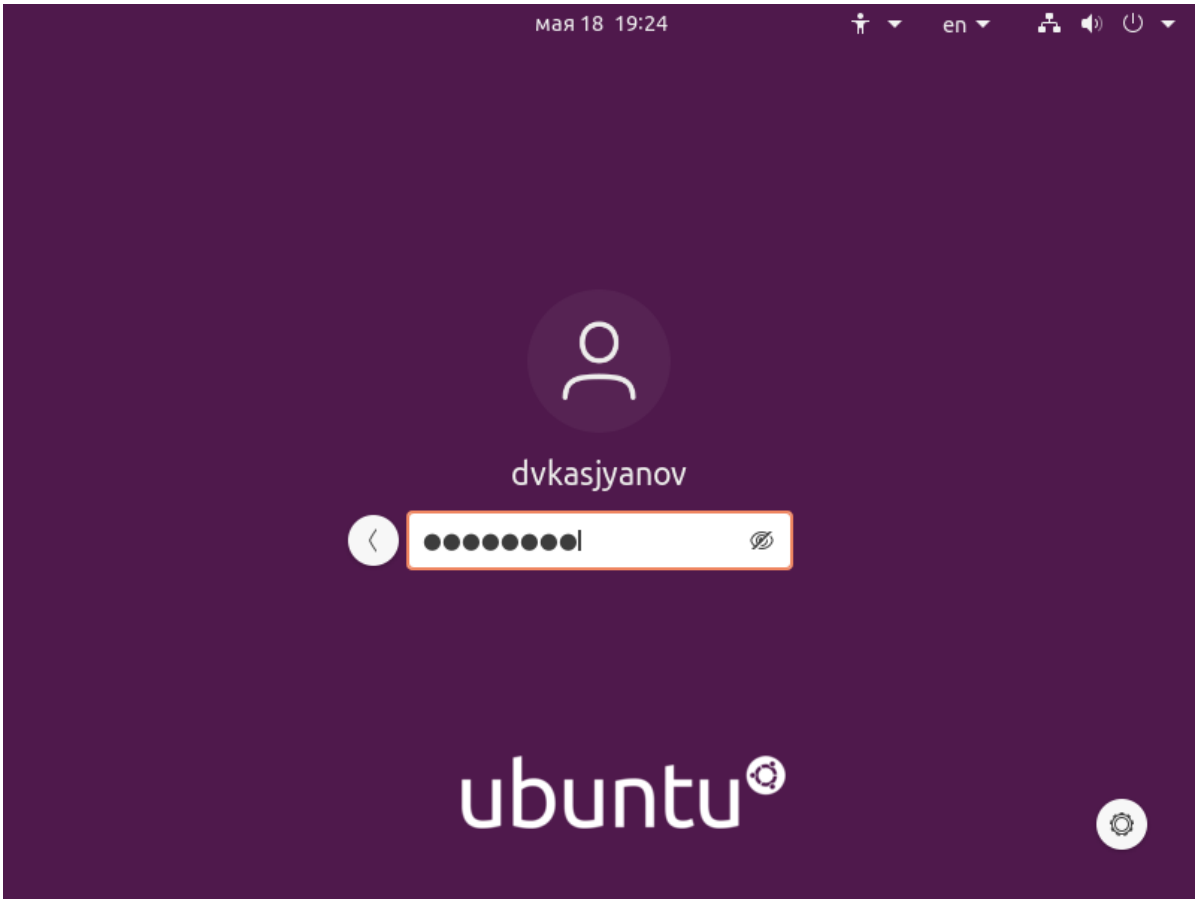
List of Figures

1 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.

2 Выполнение лабораторной работы

1. Осуществляю вход в систему (Рисунок 1).



(Рисунок 1)

2. Запишу в файл file.txt названия файлов, содержащихся в каталоге /etc. Допишу в этот же файл названия файлов, содержащихся в домашнем каталоге (Рис. 2, 3, 4).

```
dvkasjyanov@dvkasjyanov:~$ ls -a /etc > file.txt
dvkasjyanov@dvkasjyanov:~$ cat file.txt
.
..
acpi
adduser.conf
alsa
alternatives
anacrontab
apg.conf
apm
apparmor
apparmor.d
apport
appstream.conf
apt
avahi
bash.bashrc
bash_completion
bash_completion.d
bindresvport.blacklist
binfmt.d
bluetooth
brlapi.key
brltty
brltty.conf
ca-certificates
ca-certificates.conf
ca-certificates.conf.dpkg-old
calendar
chatscripts
console-setup
cracklib
cron.d
cron.daily
cron.hourly
cron.monthly
crontab
```

(Рисунок 2)

```
dvkasjyanov@dvkasjyanov:~$ ls -a ~ >> file.txt
dvkasjyanov@dvkasjyanov:~$ cat file.txt
.
..
acpi
adduser.conf
alsa
alternatives
anacrontab
apg.conf
apm
apparmor
apparmor.d
apport
appstream.conf
apt
avahi
bash.bashrc
bash_completion
bash_completion.d
bindresvport.blacklist
binfmt.d
bluetooth
brlapi.key
brltty
brltty.conf
ca-certificates
ca-certificates.conf
ca-certificates.conf.dpkg-old
calendar
chatscripts
console-setup
cracklib
cron.d
cron.daily
cron.hourly
cron.monthly
crontab
```

(Рисунок 3)

```
abc1
australia
.bash_history
.bash_logout
.bashrc
.cache
.config
conf.txt
Desktop
Documents
Downloads
feathers
file.txt
.gnupg
l.log
.local
logfile.txt
may
monthly
.mozilla
Music
my_os
Pictures
play
print
.profile
Public
reports
ski.plases
snap
.sudo_as_admin_successful
Templates
Videos
dvkasjyanov@dvkasjyanov:~$
```

(Рисунок 4)

3. Выведу имена всех файлов из file.txt, имеющих расширение `.conf`, после чего запишу их в новый текстовой файл conf.txt с помощью команды `grep '\.conf$' file.txt > conf.txt` (Рисунок 5).

```
dvkasjyanov@dvkasjyanov:~$ grep '\.conf$' file.txt > conf.txt
dvkasjyanov@dvkasjyanov:~$ cat conf.txt
adduser.conf
apg.conf
appstream.conf
brltty.conf
ca-certificates.conf
debconf.conf
deluser.conf
e2scrub.conf
fprintd.conf
fuse.conf
gai.conf
hdparm.conf
host.conf
kernel-img.conf
kerneloops.conf
ld.so.conf
libao.conf
libaudit.conf
logrotate.conf
ltrace.conf
mke2fs.conf
mtools.conf
nsswitch.conf
pam.conf
pnm2ppa.conf
popularity-contest.conf
resolv.conf
rsyslog.conf
rygel.conf
sensors3.conf
sysctl.conf
ucf.conf
usb_modeswitch.conf
xattr.conf
dvkasjyanov@dvkasjyanov:~$
```

(Рисунок 5)

4. Определяю, какие файлы в моем домашнем каталоге имеют имена, начинающиеся с символа `c`. Это можно сделать несколькими способами:

- Используя команду `find -maxdepth 1 -name 'c*'` (`maxdepth` - глубина присмотра) (Рисунок 6):

```
dvkasjyanov@dvkasjyanov:~$ find ~ -maxdepth 1 -name 'c*'
/home/dvkasjyanov/conf.txt
```

(Рисунок 6)

- Используя команду `ls ~ | grep c*` (Рисунок 7):

```
dvkasjyanov@dvkasjyanov:~$ ls ~ | grep c*
conf.txt
dvkasjyanov@dvkasjyanov:~$
```

(Рисунок 7)

5. Постранично выведу имена файлов из каталога /etc, начинающиеся с символа h (Рис. 8, 9).

```
dvkasjyanov@dvkasjyanov:~$ find /etc -maxdepth 1 -name 'h*' | less
dvkasjyanov@dvkasjyanov:~$
```

(Рисунок 8)

```
/etc/hostid
/etc/hostname
/etc/hosts.allow
/etc/hosts.deny
/etc/hp
/etc/hdparm.conf
/etc/host.conf
/etc/hosts
(END)
```

(Рисунок 9)

6. Запущу в фоновом режиме процесс, который будет записывать в файл ~/logfile файлы, имена которых начинаются с log (Рисунок 10).

```
dvkasjyanov@dvkasjyanov:~$ find -name 'log*' > logfile &
[1] 8480
dvkasjyanov@dvkasjyanov:~$ cat logfile
./logfile
./.local/share/keyrings/login.keyring
./.mozilla/firefox/ezhomkr9.default-release/logins-backup.json
./.mozilla/firefox/ezhomkr9.default-release/logins.json
./logfile.txt
[1]+  Done                    find -name 'log*' > logfile
dvkasjyanov@dvkasjyanov:~$
```

(Рисунок 10)

7. Удалю ~/logfile (Рисунок 11);

```
dvkasjyanov@dvkasjyanov:~$ rm ~/logfile
[1]+  Done                    find ~ -maxdepth 1 -name 'log*' > ~/logfile
dvkasjyanov@dvkasjyanov:~$
```

(Рисунок 11)

8. Запускаю из консоли в фоновом режиме редактор gedit (Рисунок 12):

(Рисунок 12)

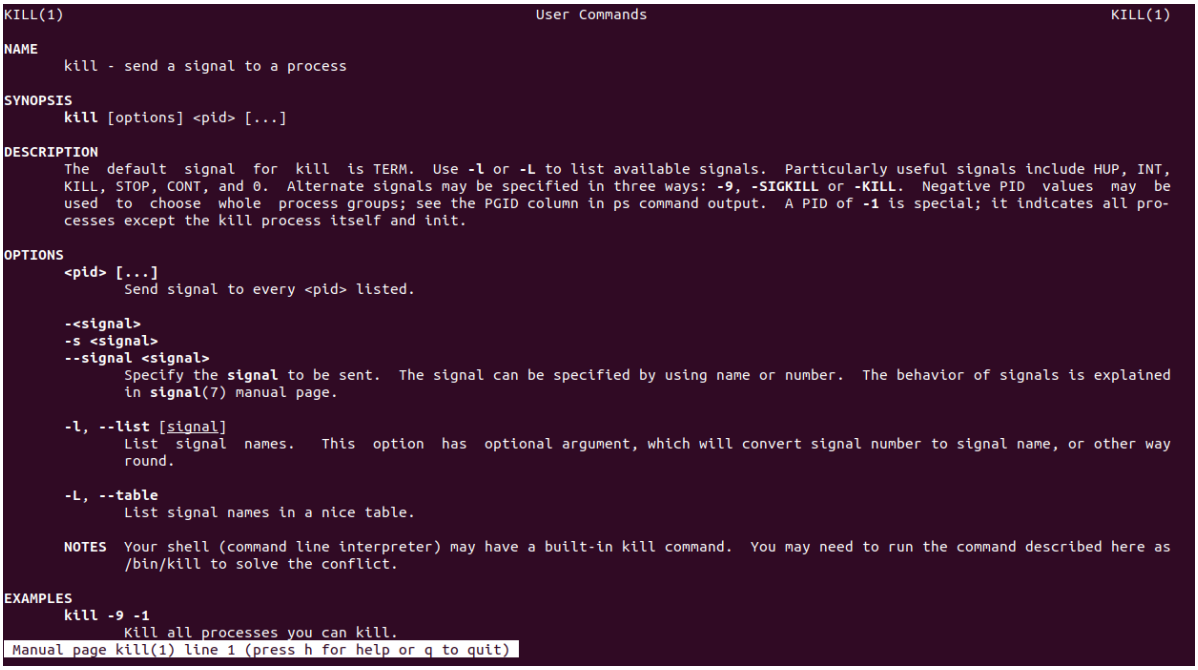
9. Определяю идентификатор процесса gedit, используя команду `ps`, конвейер и фильтр `grep` (Рисунок 13).

(Рисунок 13)

Определить этот идентификатор можно более простыми способами - с помощью команд `pgrep` и `pidof` (Рисунок 14).

(Рисунок 14)

10. Прочитаю справку `man` команды `kill` (Рисунок 15), после чего использую её для завершения процесса `gedit` (Рисунок 16).



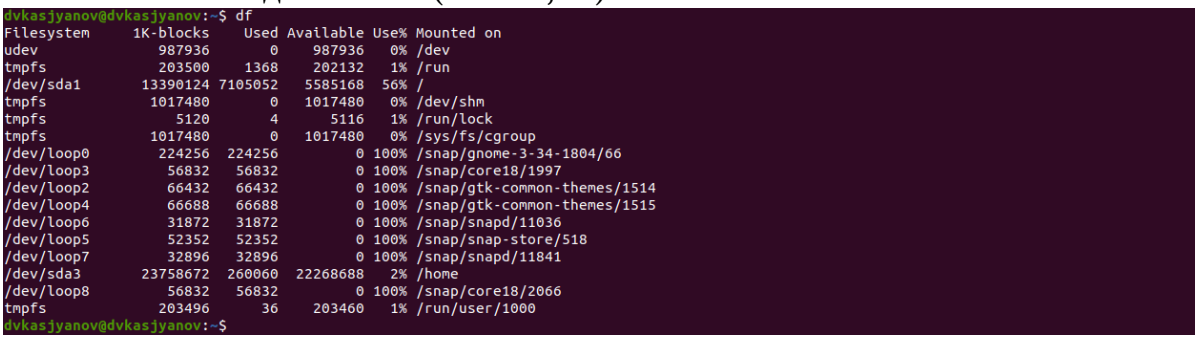
(Рисунок 15)

(Рисунок 16)

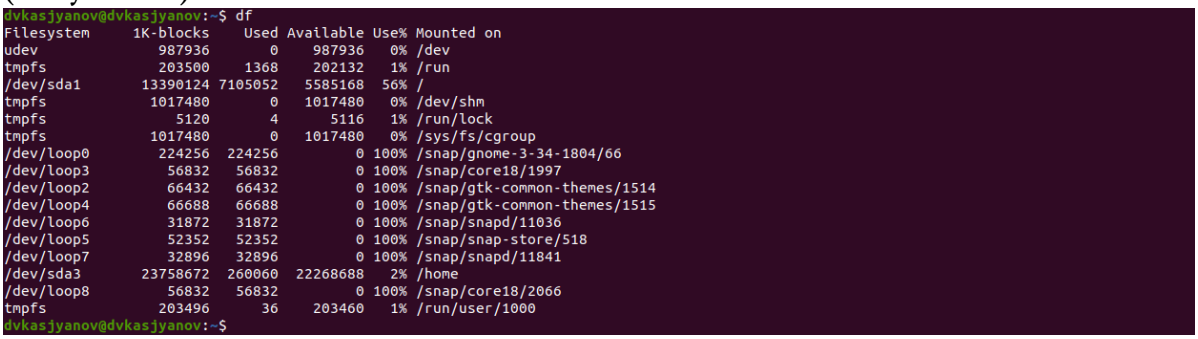
11. Получаю более подробную информацию о командах `df` и `du` с помощью команды `man` (Рисунок 17).

(Рисунок 17)

Выполняю команды `df` и `du` (Рис. 18, 19).

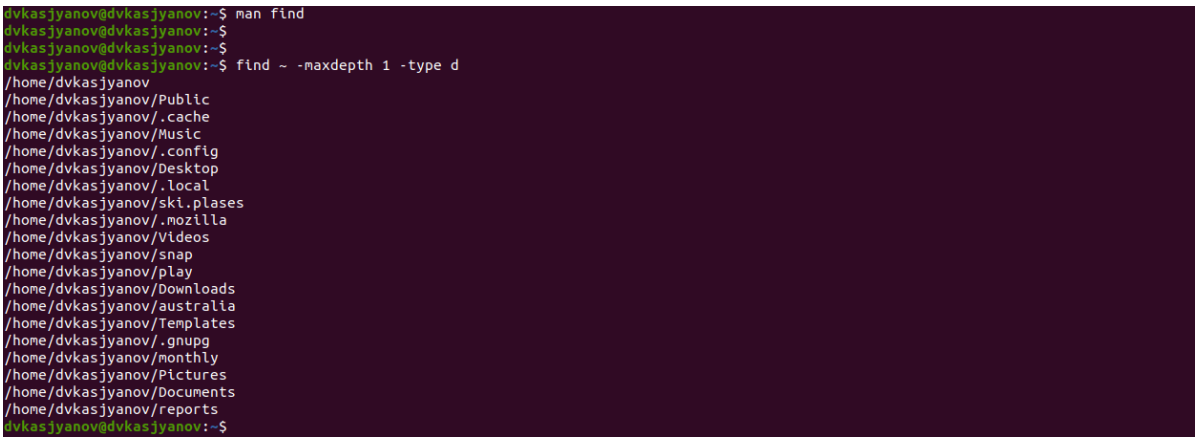


(Рисунок 18)



(Рисунок 19)

12. Воспользовавшись справкой команды `find`, вывожу имена всех директорий, имеющихся в моем домашнем каталоге. Для этого использую команду `find ~ -maxdepth 1 -type d` (Рисунок 20).


dvkasjyanov@dvkasjyanov:~$ man find
dvkasjyanov@dvkasjyanov:~$
dvkasjyanov@dvkasjyanov:~$
dvkasjyanov@dvkasjyanov:~$ find ~ -maxdepth 1 -type d
/home/dvkasjyanov
/home/dvkasjyanov/Public
/home/dvkasjyanov/.cache
/home/dvkasjyanov/Music
/home/dvkasjyanov/.config
/home/dvkasjyanov/Desktop
/home/dvkasjyanov/.local
/home/dvkasjyanov/ski.plases
/home/dvkasjyanov/.mozilla
/home/dvkasjyanov/Videos
/home/dvkasjyanov/snap
/home/dvkasjyanov/play
/home/dvkasjyanov/Downloads
/home/dvkasjyanov/australia
/home/dvkasjyanov/Templates
/home/dvkasjyanov/.gnupg
/home/dvkasjyanov/monthly
/home/dvkasjyanov/Pictures
/home/dvkasjyanov/Documents
/home/dvkasjyanov/reports
dvkasjyanov@dvkasjyanov:~$

(Рисунок 20)

3 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.

2. Операция > позволяет перенаправить поток вывода в какой-либо файл, при этом файл будет создан, если его не существует, или перезаписан, если такой файл уже есть.

   Операция >> имеет те же функции, но если файл существует, то она не перезаписывает его, а добавляет новые данные после старых.

3. Конвейер (`pipe`) служит для объединения простых команд или утилит в цепочки, в которых результат работы предыдущей команды передаётся последующей. Используются для автоматизации рутинных операций в консоли.

   Синтаксис следующий: `команда 1 | команда 2`, при этом 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 по-

следний 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.

Основное различие между программой и процессом заключается в том, что программа представляет собой группу инструкций для выполнения определенной задачи, тогда как процесс представляет собой программу в процессе выполнения. Хотя процесс является активной сущностью, программа считается пассивной.

Между процессом и программой существует отношение многие-к-одному, что означает, что одна программа может вызывать несколько процессов или, другими словами, несколько процессов могут быть частью одной и той же программы.

5. `PID` - идентификатор процесса.

   `GID` - идентификатор группы UNIX, под которым работает программа.

6. 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` – 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.

   Синтаксис: `find [папка] [параметры] критерий шаблон [действие]`. `Папка` – каталог в котором производится поиск. `Параметры` – дополнительные параметры, например, глубина поиска. `Критерий` – критерий поиска: имя, дата создания, права, владелец и т.д. `Шаблон` – непосредственно значение по которому производится отбор файлов.

   - `-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 необходимо поставить `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 ~ -name 'purpose_of_life*'` - поиск в домашнем каталоге файлов, начинающихся с “purpose_of_life”;
- `find /tmp -maxdepth 3 -type f` - поиск файлов в каталоге /tmp глубины 3.

9. Да, для этого используется команда `grep "содержание файла, который мы ищем"`.
10. Для этого используется команда `df`.
11. Для этого используется команда `du`. Для домашнего каталога: `du ~`.
12. Основные сигналы, 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]`. Сигнал представляет собой один из выше перечисленных сигналов для завершения процесса. Перед тем, как выполнить остановку процесса, нужно определить его PID.

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.

4 Выводы

Я ознакомился с инструментами поиска файлов и фильтрации текстовых данных, приобрел практические навыки: по управлению процессами (и заданиями), по проверке использования диска и обслуживанию файловых систем.

5 Библиография

Linux: перенаправление

Потоки данных

Основы Linux от основателя Gentoo. Часть 2 (4/5): Обработка текста и перенаправления

Команда find в Linux – мощный инструмент сисадмина

Разница между программой и процессом