Дополнительные материалы к курсу

Расширенное администрирование РЕД ОС

Автор: Груздев П. Ю.

г.Екатеринбург
2024

Оглавление

Глава 1. Работа в командной оболочке bash

1.1. Что такое оболочка?

Командная оболочка (shell) – это программа, взаимодействующая с пользователем с помощью интерфейса командной строки и позволяющая пользователю запускать прикладные программы и выполнять различные команды операционной системы.

Оболочка интерпретирует введенные пользователем команды, и преобразует их в инструкции операционной системы.

Показывая пользователю, что оболочка готова интерпретировать команды, она выводит специальное приглашение командной строки, заканчивающееся обычно символом доллара `$` в сеансе обычного пользователя.

В сеансе суперпользователя оболочка обычно в качестве приглашения использует символ решетки `#`, предупреждая о возможности нарушения работоспособности всей системы вследствие ошибочных действий.

В GNU/Linux может быть использовано множество различных оболочек, однако стандартом de facto является оболочка Bourne Again Shell – bash.

Оболочка запускается при входе пользователя в сеанс.

Какая конкретно оболочка будет запущена определяется учетной записью пользователя. Определить, какая оболочка установлена в учетной записи пользователя, можно путем вывода переменной SHELL:

Пример:

```
$ echo $SHELL
/bin/bash
```

Примечание: Переменная окружения `SHELL` *содержит в себе полное имя исполняемого файла оболочки пользователя, используемой при входе в сеанс этого пользователя. В данном случае – это Bash. Символ доллара* `$` *перед переменной окружения используется для извлечения ее значения.*

Команды пользователя представляют собой строки, вводимые с клавиатуры.

После того, как команда введена и нажата клавиша `Enter`, команда интерпретируется оболочкой и, при удачной интерпретации, исполняется.

Если команда введена синтаксически неверно, то выдается сообщение об ошибке.

Помимо отдельных команд, вводимых последовательно с клавиатуры, в командной оболочке можно также использовать файлы сценариев.

Сценарии позволяют выполнять достаточно сложные задачи с использованием условных переходов, циклов и подпрограмм.

1.2. Наиболее распространенные оболочки в GNU/Linux.

В GNU/Linux можно использовать множество различных оболочек, а также можно написать свою собственную оболочку, если существующие варианты не удовлетворяют имеющихся требований.

Наиболее распространены четыре вида оболочек:

Bash shell (`bash`) – используется по умолчанию;

Public domain Korn shell (`pdksh` или `ksh`);

Enhanced C shell (`tcsh`);

Z Shell (`zsh`).

Примечание: В скобках указаны команды для запуска этих оболочек (имена исполняемых файлов оболочек).

Различные оболочки обладают различным набором функций и даже различными встроенными командами.

Встроенные команды оболочки – это команды, которые реализованы внутри нее, а не в виде внешних программ.

При запуске скриптов необходимо убеждаться, что скрипт предназначен для выполнения в данной оболочке.

Bash shell является лидером по популярности и большинство пользователей GNU/Linux используют именно эту оболочку.

Bash устанавливается по умолчанию для пользователей GNU/Linux и именно она обычно загружается после входа в сеанс.

Оболочка Bash позволяет настраивать вид приглашения, однако чаще всего, обычные пользователи видят приглашение, заканчивающееся символом `$`, а для суперпользователя (root) приглашение заканчивается символом `#`.

Примечание: Это сделано для лишнего напоминания суперпользователю о требуемой при его работе осторожности.

В системе может быть установлено множество оболочек. Для получения их списка можно воспользоваться командой `chsh -l` , если она установлена в системе и настройки безопасности позволяют ей воспользоваться.

Можно также просто просмотреть содержимое файла `/etc/shells`,

содержащего в себе список установленных в системе оболочек.

Пример: Ниже приведено типичное содержимое этого файла:

```
# cat /etc/shells
/bin/sh
/bin/bash
/bin/csh
/bin/tcsh
/bin/ksh
/bin/zsh
```

Примечание: В этом примере с помощью команды `cat` *получено содержимое файла* `/etc/shells`*, содержащего список оболочек в системе.*

Примечание: Следует еще раз отметить, что в GNU/Linux, вовсе не обязательно, чтобы оболочка, указанная в файле `/etc/shells` *была на самом деле установлена в системе. Этот файл более необходим для программ, обеспечивающих удаленный доступ к системе. Например, большинство программ – серверов FTP запрещают входить в сеанс пользователям, оболочка по умолчанию которых не указана в данном файле.*

Для временной загрузки другой оболочки необходимо просто набрать имя соответствующего исполняемого файла.

Пример: для запуска оболочки Enhanced C shell выполняем команду:

```
$ tcsh
$ ps
  PID TTY          TIME CMD
 2349 pts/0    00:00:00 bash
10295 pts/0    00:00:00 tcsh
10319 pts/0    00:00:00 ps
```

Примечание: Эта команда запустит оболочку Enhanced C shell. Далее с помощью команды `ps` демонстрируется, что оболочка `tcsh` была запущена из `bash`.

Временно загружать оболочку иногда требуется для того, чтобы выполнить сценарий командной строки, предназначенный для этой оболочки.

Для выхода из временно загруженной оболочки достаточно набрать команду `exit` .

Если системная политика это позволяет, то пользователь может изменить для себя оболочку, которая будет запускаться при его входе в сеанс. Это можно сделать с помощью команды `chsh -s <shell>` , где `<shell>` - имя исполняемого файла оболочки

Пример:

```
$ chsh –s /bin/csh
```

Примечание: Здесь установлена оболочка, которая будет загружаться по умолчанию, C shell, исполняемый файл которой `/bin/csh`. *При этом, всякий раз, когда пользователь будет входить в сеанс, будет загружена именно эта оболочка.*

1.3. Встроенные и системные команды.

Все команды GNU/Linux делятся на два больших класса: встроенные и системные.

Встроенные команды интерпретируются и выполняются самой оболочкой.

Системные команды представляют собой исполняемые файлы, находящиеся в специальных каталогах.

Встроенные команды представляют собой процедуры оболочки и, следовательно, выполняются быстрее системных команд.

В силу того, что они привязаны к конкретной оболочке, необходимо понимать, что одинаковые команды в разных оболочках могут вести себя по-разному.

Примечание: например, команда `set` *в Bash и C shell используется совершенно по-разному.*

Основная масса встроенных команд, все – таки, одинакова в различных оболочках.

Командная оболочка `bash`, например, предоставляет пользователю такие встроенные команды, как `cd`, `alias`, `bg`, `kill`, `pwd` и `echo`.

Исполняемые файлы системных команд обычно находятся в одном из каталогов, указанных ниже:

```
/bin
/sbin
/usr/bin
/usr/sbin
/usr/local/bin
/usr/local/sbin
```

Пользователь может написать свои собственные системные команды и использовать их.

Обычно для размещения таких команд используется домашний каталог

пользователя или его подкаталог `bin`.

Если в системе имеется одновременно и встроенная и системная версия какой-либо команды, то если команда вызвана без указания пути к ней, то выполняется встроенная команда.

Если при вызове команды указан путь, то выполняется системная команда.

1.4. Переменные оболочки и переменные окружения.

Bash предоставляет возможность временного сохранения данных в переменных, называемых переменными оболочки.

Переменные оболочки размешаются в памяти автоматически при присвоении им значения.

Для создания переменной необходимо указать имя переменной, знак равно и строку – значение переменной (без пробелов).

Для извлечения значения переменной в командную строку необходимо перед именем переменной поставить знак $.

Пример: Переменной `VAR1` присваивается значение `Privet!`.

```
$ VAR1=Privet!
$ echo $VAR1
Privet!
```

Примечание: Команда `echo` здесь получает в качестве аргумента значение, сохраняемое в переменной `VAR1`, и выводит его на экран.

В оболочках переменные могут содержать лишь строковые данные.

Примечание: Даже если переменная содержит в себе цифру, то все равно переменные Bash хранятся в виде строк.

Имя переменной должно состоять только из букв и цифр или подчеркивания.

В качестве первого символа в имени переменной должна использоваться либо буква, либо символ подчеркивания.

Желательно (но не обязательно) использовать в именах переменных только большие буквы для того, чтобы не путать имена команд и имена переменных.

Пример:

```
$ hostname
work.doma.ru
$ echo $HOSTNAME
work.doma.ru
```

Примечание: Обратите внимание на то, что в этом примере команда `hostname` *выводит ту же информацию, которая содержится в переменной* `HOSTNAME`*. Тем не менее, это совершенно разные объекты.*

Если переменная должна содержать, в качестве значения, строку с пробелами, то строку следует экранировать с помощью одиночных или двойных кавычек.

Пример:

```
$ VAR1='Bolshoy Privet!'
$ echo $VAR1
Bolshoy Privet!
```

Примечание: В этом примере переменная `VAR1` *содержит строку с пробелом.*

Переменным можно присваивать значения других переменных или же их собственные значения, возможно, несколько изменяя их.

Пример:

```
$ VAR1='Vam vsem '$VAR1
$ echo $VAR1
Vam vsem Bolshoy Privet!
```

Примечание: Здесь переменной `VAR1` *было присвоено значение, составленное из строки* `Vam vsem` *и ее собственного старого значения. Следует отметить, что в данном случае пробела между строкой и символом доллара нет!*

При необходимости добавления строки к значению переменной имя переменной следует взять в фигурные скобки для отделения имени переменной от последующей строки.

Пример:

```
$ VAR1=${VAR1}'?'
$ echo $VAR1
Vam vsem Bolshoy Privet!?
```

Примечание: Пример, приведенный выше демонстрирует, как к значению переменной можно добавить строку. Если бы имя переменной не было бы экранировано с помощью фигурных скобок, то конструкция `$VAR1?` *Была бы интерпретирована Bash неверно.*

Для получения списка всех переменных, определенных в текущей оболочке, следует использовать команду `set`.

Пример:

```
$ set | less
```

Примечание: Команда выведет список всех переменных, определенных в данной оболочке, и их значения, отобразив его с помощью программы постраничного просмотра `less`.

Если вы хотите уничтожить переменную, то для этого достаточно выполнить команду `unset`, указав в качестве ее аргумента имя переменной.

Пример:

```
$ VAR1="It's a variable"
$ echo $VAR1
It's a variable
$ unset VAR1
$ echo $VAR1
```

Примечание: После того, как переменная `VAR1` *была удалена командой* `unset` *, команда* `echo $VAR1` *ничего не выводит на экран.*

Переменные оболочки доступны только в той оболочке, в которой они были описаны.

Примечание: Нельзя извлечь значение переменной из порожденной оболочки или из программы, запущенной из этой оболочки.

Однако существует способ превратить переменную оболочки в так называемую переменную окружения.

Для этого необходимо произвести операцию экспортирования переменной, после чего она станет доступной для любых программ, запущенных из оболочки.

Экспортирование переменной осуществляется с помощью команды `export`.

Пример:

```
$ VAR1=01/01/1970
$ echo $VAR1
01/01/1970
$ bash
$ echo $VAR1
$ exit
exit
$ export VAR1
$ bash
$ echo $VAR1
01/01/1970
$ exit
exit
```

Примечание: Этот пример демонстрирует, что переменная `VAR1`*, описанная в текущей оболочке, при запуске новой оболочки командой* `bash`*, не известна в порожденной оболочке. Однако, после экспортирования ее она превращается в переменную окружения и становится доступной в порожденной оболочке.*

Весь набор переменных окружения содержится в массиве, называемом environ.

Все переменные окружения и их значения могут быть получены с помощью команды `env`.

Окружение – это один из способов передачи информации процессам.

Очень часто изменение значения какой-либо переменной окружения приводит к изменению поведения запускаемой программы.

Ниже приведена таблица часто используемых переменных окружения.

Переменная	Сущность
`HOME`	Путь к домашнему каталогу пользователя.
`LOGNAME`	Имя пользователя.
`MAIL`	Путь к почтовому ящику пользователя.
`PATH`	Путь поиска исполняемых файлов.
`PS1`	Вид приглашения оболочки.
`PWD`	Имя текущего каталога.
`SHELL`	Текущая оболочка.
`USER`	Имя пользователя.

Переменная окружения `PS1`, определяющая вид приглашения оболочки, кодируется с помощью специальных символов.

Пример:

```
[user1@work tmp]$ echo $PS1
[\u@\h \W]\$
```

Примечание: Символ `\u` *устанавливает вывод имени пользователя, затем идет двоеточие, затем символ* `\W` *устанавливает вывод имени текущего каталога, а символ* `\h` *заставляет отображать в строке приглашения Bash имя хоста.*

Выходя из сеанса, пользователь завершает работу оболочки, поэтому установленные им переменные уничтожаются.

Для того чтобы необходимые значения переменных окружения автоматически устанавливались при входе в сеанс, их необходимо инициализировать в файлах профиля.

Переменные окружения, общие для всех пользователей, хранятся в файле `/etc/profile`

Настройки специфичные для конкретных пользователей хранятся в одном из файлов домашнего каталога пользователя: либо в `~/.bash_profile`, либо `~/.bash_login`, либо `~/.profile`.

Эти файлы выполняются при каждом входе в сеанс.

При каждом запуске оболочки bash выполняется файл профиля `~/.bashrc`.

1.5. Перенаправление потоков ввода-вывода.

Если программа требует чтения или записи в файл, то этот файл должен быть сначала открыт, т.е. создан поток (stream) данных.

Процедура открытия файла создает структуру в ядре, называемую файловым дескриптором.

Все файловые дескрипторы пронумерованы.

С любым пользовательским процессом при его создании связываются три файловых дескриптора (потока):

стандартный поток ввода (`stdin`), файловый дескриптор которого – `0`;

стандартный поток вывода (`stdout`), файловый дескриптор – `1`;

стандартный поток вывода ошибок (`stderr`), файловый дескриптор – `2`.

Примечание: Поток ввода открыт на чтение, а потоки вывода и ошибок – на запись. Обычно по умолчанию стандартный поток ввода связан с клавиатурой, а стандартные потоки вывода и ошибок связаны с дисплеем.

Оболочка Bash позволяет перенаправить стандартные потоки в файлы,

для чего используются следующие операторы:

< имя_файла или 0< `имя_файла` – перенаправление стандартного потока ввода;

> имя_файла или `1> имя_файла` – перенаправление стандартного потока вывода;

`2> имя_файла` – перенаправление стандартного потока вывода ошибок

Если файла, в который производится запись не существует, то такой файл создается

Пример: с помощью такой команды можно отправить электронное письмо с текстом, содержащемся в файле `letter` :

```
$ mail -s 'Pismo' user1 < letter
```

Примечание: Эта команда направит электронное письмо пользователю `user1`. *Тема письма (Subject), указана после опции* `-s`, *а текст письма передан через стандартный поток ввода из файла* `letter`.

Пример: Следующая команда использует перенаправление стандартного потока вывода в файл `ls.txt` :

```
$ ls > ls.txt
```

Примечание: В файле `ls.txt` *в результате работы такой команды окажется список файлов в текущем каталоге, выведенный командой* `ls` *в стандартный поток вывода.*

Пример: Аналогично в файл можно перенаправить ошибки:

```
$ ls -ld /etc /ctc 2> ls.err
drwxr-xr-x   87 root     root         6064 Окт 15 18:57 /etc
$ cat ls.err
ls: /ctc: No such file or directory
```

Примечание: В этом примере поток вывода ошибок был перенаправлен в файл `ls.err`. *В силу того, что в качестве аргумента команды* `ls -ld` *был задан несуществующий каталог* `/ctc`, *то команда вывела соответствующее сообщение об ошибке, которое и было перенаправлено в файл* `ls.err`. *Содержимое файла было выведено с помощью команды* `cat`.

Пример: Поток вывода перенаправлен в файл `ls.txt` одновременно с перенаправлением потока ошибок в файл `ls.err`.

```
$ ls -ld /etc /ctc > ls.txt 2> ls.err
```

Чтобы перенаправить оба потока вывода в один и тот же файл следует использовать оператор сцепления потоков &

Оболочка bash позволяет следующие эквивалентные формы записи перенаправления потоков вывода в один и тот же файл:

```
 > имя_файла 2>&1
2> имя_файла 1>&2
>&  имя_файла
&> имя_файла
```

```
Примеры:
$ ls -ld /etc /ctc > ls.txt 2>&1
$ ls -ld /etc /ctc 2> ls.txt >&2
$ ls -ld /etc /ctc &> ls.txt
```

Операции перенаправления потоков вывода и вывода ошибок в файл стирают его содержимое, записывая новое содержимое взамен старого.

Операцию перенаправления можно использовать для стирания содержимого файлов и создания новых пустых файлов.

Пример: для стирания содержимого файла `ls.txt` можно использовать команду

```
$  > ls.txt
```

Оболочка Bash позволяет исключить стирание содержимого файлов при перенаправлении в них потоков вывода или ошибок с помощью установки флага `noclobber` командой set -o.

Пример:

```
$ set -o noclobber
```

Значение всех флагов оболочки можно с помощью команды `set -o`.

Пример:

```
$ set -o
allexport       off
braceexpand     on
emacs           on
```

```
errexit         off
hashall         on
histexpand      on
history         on
ignoreeof       off
interactive-comments    on
keyword         off
monitor         on
noclobber       on
noexec          off
noglob          off
nolog           off
notify          on
nounset         off
onecmd          off
physical        off
posix           off
privileged      off
verbose         off
vi              off
xtrace          off
```

Примечание: Листинг, выведенный командой `set -o`*, демонстрирует, что после выполнения пользователем команды* `set -o noclobber`*, данная опция была активизирована (состояние* `on`*). Установка этой опции оболочки предотвращает перезапись существующих файлов с помощью операций вывода.*

Пример: Попробуем очистить существующий файл `dir.txt`**.**

```
$ > dir.txt
bash: dir.txt: cannot overwrite existing file
```

Примечание: Так как установлена опция `noclobber`*, то оболочка не позволила переписать (в данном случае стереть) существующий файл.*

Для перезаписи содержимого файла при установленной опции `noclobber` можно воспользоваться операторами:

`>|` - для перенаправления потока вывода с гарантированной перезаписью файла;

`2>|` - для перенаправления потока вывода ошибок с гарантированной перезаписью файла.

Пример:

```
$ ls -l >| dir.txt
```

Примечание: Эта команда запишет подробную информацию о содержимом текущего каталога в существующий файл `dir.txt`*, не обращая внимания на то, что файл уже существует и установлена опция оболочки* `noclobber`.

Отключить опцию `noclobber` (как и другие опции по аналогии) можно с помощью команды: set +o noclobber

Если необходимо добавить в существующий файл информацию из потоков вывода, то можно использовать следующие операторы:

`>>` - для перенаправления потока вывода на добавление к файлу;

`2>>` - для перенаправления потока вывода ошибок на добавление к файлу

Если файла не существует, то создается новый файл.

Пример: следующая команда добавит в файл `dir.txt` имя текущего каталога:

```
$ pwd >> dir.txt
```

Некоторые команды, работающие с текстовыми файлами, позволяют вместо имени файла указать стандартный поток ввода. Для чего используется символ –

Пример: команда `vi -` позволяет считать редактируемый текст из стандартного потока ввода вместо открытия файла.

Для завершения потока ввода, производимого с клавиатуры, следует набрать сочетание клавиш `Ctrl-D`, передающее в поток ввода символ конца файла.

Пример:

```
$ view -
Privet
<Ctrl-D>
```

Примечание: В этом примере команда `view` *позволила считать содержимое потока ввода с клавиатуры, окончание которого было обозначено с помощью сочетания* `Ctrl-D`.

Конструкция here document (документ здесь) позволяет вместо символа конца файла (EOF), который не может быть использован внутри файла, воспользоваться любым другим удобным символом.

Пример: следующая конструкция обеспечивает посылку письма пользователю `user1`, где тело письма передается через поток ввода команды `mail` с помощью here document:

```
$ mail -s HereDoc user1 << .
This is Here Document here!
.
```

Примечание: Обратите внимание, что в предыдущем примере вместо использования символа окончания потока здесь используется символ точки. Этот символ был задан в качестве ограничителя потока ввода в командной строке << . . Это обозначает, что текст, вводимый через стандартный поток ввода, должен быть ограничен символом точка. Символ – ограничитель должен находиться в строке, перед которой чтение стандартного потока ввода прекращается. Символ – ограничитель должен быть единственным символом в строке (не считая символа перевода строки).

1.6. Конвейеры и фильтры.

Конвейер (pipe) позволяет направить стандартный поток вывода (stdout) одного процесса в стандартный поток ввода (stdin) другого процесса.

Для организации конвейера необходимо поставить символ конвейера – вертикальную черту | между командами, потоки которых необходимо объединить.

Пример:

```
$ last | view -
```

Примечание: Выводимый командой `last` список входивших в сеанс пользователей передан через конвейер команде просмотра `view` (один из вариантов вызова `vi`). Знак тире после команды `view` сообщает, что данные должны быть введены из стандартного потока ввода, в который передает данные команда `last` через свой поток вывода.

Команда `tee` позволяет организовать вывод информации, полученной из потока ввода, в файлы, указанные как аргументы, и в стандартный поток вывода одновременно.

Пример: команда `ps` в примере ниже выведет список процессов в файл `ps.txt` и в стандартный поток вывода.

```
$ ps | tee ps.txt
  PID TTY          TIME CMD
 1720 pts/0    00:00:00 bash
 2438 pts/0    00:00:00 ps
```

```
 2439 pts/0    00:00:00 tee
$ cat ps.txt
  PID TTY          TIME CMD
 1720 pts/0    00:00:00 bash
 2438 pts/0    00:00:00 ps
 2439 pts/0    00:00:00 tee
```

Примечание: Содержимое файла `ps.txt` *совершенно идентично выводу команды* `ps`. *Обратите внимание на присутствие в списке процессов команды* `tee`.

Примечание: Команда `tee` *наиболее часто используется для отладки работы сложных конвейеров, состоящих из множества команд. Эту команду удобно устанавливать в месте конвейера, которое вызывает подозрения. Так как в файле, указанном в качестве аргумента* `tee`, *будет находиться та же информация, что была передана в данном месте конвейера, то по ней легко можно будет определить наличие и суть ошибок.*

Фильтр – это не интерактивная команда, способная принимать поток данных через стандартный поток ввода, обрабатывать его, и выводить обработанные данные в стандартный поток вывода.

Пример: команда `wc -l` **– фильтр, подсчитывающий количество строк в потоке ввода и передающий результат подсчета в поток стандартного вывода. Конвейер, показанный ниже, подсчитывает число процессов, работающих в системе от имени пользователя** `user1`**.**

```
$ ps -u user1 | wc -l
      8
```

Команды, находящиеся в конвейере должны удовлетворять следующим требованиям:

Первая команда конвейера должна уметь выводить информацию в стандартный поток вывода.

Команды, находящиеся внутри конвейера, должны быть фильтрами.

Последняя команда в конвейере должна уметь читать стандартный поток ввода.

Пример: для отправки суперпользователю письма с отсортированным списком пользователей, вошедших в сеанс, можно использовать такой конвейер:

```
$ who | sort | mail -s 'Logged users' root
```

Примечание: В этом примере команды `who` *и* `mail` *не являются фильтрами, но*

поскольку команда `who` *осуществляет вывод в стандартный поток вывода, а* `mail` *читает поток ввода, то их можно использовать, соответственно, в начале и конце конвейера. Команда* `sort`*, осуществляющая сортировку, является фильтром, то есть она читает из стандартного потока ввода и выводит отсортированный текст в стандартный поток вывода, поэтому она вполне может находиться где-либо в середине конвейера.*

1.7. Ввод, редактирование и исполнение команд.

При работе в командной строке можно использовать привычные клавиши управления курсором и клавиши редактирования.

Однако, во-первых, не все виды клавиатур обеспечивают такие клавиши, как `Home`, а, во-вторых, во многих случаях привычные клавиши управления курсором не работают.

Ниже приведена таблица клавиатурных сочетаний, которые могут быть использованы при работе в командной строке.

Клавиши	Действие
`Ctrl-B`	Курсор влево
`Ctrl-F`	Курсор вправо
`Alt-B`	Курсор на слово влево
`Alt-F`	Курсор на слово вправо
`Ctrl-A`	Курсор в начало строки
`Ctrl-E`	Курсор в конец строки
`Ctrl-H`	Удаление символа перед курсором
`Ctrl-D`	Удаление символа в позиции курсора
`Alt-D`	Удаление слова
`Ctrl-U`	Удаление части строки слева от курсора
`Ctrl-K`	Удаление части строки справа от курсора
`Ctrl-M или Ctrl-J`	Ввод
`Ctrl-V`	Отмена специального значения символа
`Ctrl-L`	Очистка экрана
`Alt-T`	Перемена мест аргументов
`Alt-L`	Перевод слова в нижний регистр
`Alt-U`	Перевод слова в верхний регистр
`Ctrl-C`	Остановка выполнения задания
`Ctrl-Z`	Приостановка выполнения задания

Иногда бывает необходимо ввести в командную строку символический код клавиатурного сочетания вместо выполнения команды, связанной с этим сочетанием. В этом случае перед вводом клавиатурного сочетания, имеющего специальное значение, необходимо ввести `Ctrl-V`

Пример:

```
$ echo "1   2       3"
1       2       3
```

Примечание: Обратите внимание на то, что в предыдущем примере в качестве аргумента команды `echo` *использована строка, в которой имеются символы табуляции. Для ввода такой строки необходимо перед каждым символом табуляции нажать сочетание* `Ctrl-V`*, так как в противном случае Bash ошибочно воспринимает символ табуляции как специальный символ.*

Нажатие `Ctrl-C` приводит к передаче процессу сигнала `INT`, что эквивалентно команде `kill -2` . Однако не все задания могут быть остановлены так.

Нажатие `Ctrl-Z` приводит к приостановке активного задания, которое затем может быть переведено в фоновый режим командой `bg`, либо это задание может быть уничтожено командой `kill`.

Оболочка Bash предоставляет специальную возможность исправить ошибку, допущенную при вводе команды. Исправление достигается изменением цепочки символов в выполненной ранее команде и выполнением новой измененной команды. Для этого надо набрать: `^заменяемое^замена` и нажать `Enter` при этом будет выполнена команда с замененной подстрокой

Пример:

```
$ ls /dmb
ls: /dmb: No such file or directory
$ ^dmb^tmp
ls /tmp
0103740143  0793455464  1645210352  AcromIO5pU
```

Примечание: Команде `ls` *в этом примере был указан в качестве аргумента несуществующий каталог, поэтому было получено сообщение об ошибке. Далее, пользуясь механизмом замены символов, подстрока* `dmb` *была заменена на* `tmp`*, команда была исполнена автоматически и на экран было выведено содержимое каталога* `/tmp`.

В случае если необходимо ввести длинную команду, которая не помещается в одну строку, необходимо воспользоваться символом обратной косой черты `\` и продолжить ввод на следующей строке

Пример:

```
$ find . \
```

```
> -name "*prim*" \
> -user 501 \
> -type f \
> -ls
1352836     20  -rw-r--r--     1 user     user          16408  Окт  14
1999 ./Documents/DocBook/html/primaryie.html
1353135     16  -rw-r--r--     1 user     user          16093  Окт  14
1999 ./Documents/DocBook/html/primary.html
```

Примечание: Команда `find . -name "*prim*" -user 501 -type f -ls`, *выполненная здесь, довольно длинная. Она ищет все обычные файлы в текущем каталоге, в имени которых встречается строка* `prim`, *принадлежащие пользователю с UID=501, и печатает найденные имена файлов в формате, подобном* `ls -l`. *Так как команда длинная, то ее удобно ввести по строкам, разделяя каждую строку, входящую в команду, с помощью символа обратной косой черты. Знаки больше, показанные в листинге, являются вторичным приглашением командной строки. Они могут быть установлены с помощью переменной окружения* `PS2`.

Можно вводить несколько команд в одной строке, разделяя их символом точка с запятой `;`

Пример:

```
$ cd /opt; ls -l; cd; pwd
итого 8
drwxr-xr-x    7 root     root         4096 Июл 23 08:21 drweb
drwxr-xr-x    7 oracle   oinstall     4096 Июл 23 07:20 oracle
/home/user
```

Примечание: В командной строке, приведенной выше, объединено сразу четыре команды, разделенные точкой с запятой.

Если команды отделены друг от друга с помощью двух амперсандов `&&`, то вторая команда будет выполнена только в случае успешного выполнения первой. То есть, в случае, когда первая команда вернула нулевой код возврата.

Пример:

```
$ ls nofile && cat nofile
ls: nofile: No such file or directory
$ ls /etc/motd && cat /etc/motd
/etc/motd
It's time to work!
```

Примечание: Первая команда `ls nofile` *возвратила ненулевой код возврата, так как такого файла не оказалось, поэтому вторая команда не была выполнена. Во втором*

случае первая команда закончилась удачно, поэтому была выполнена вторая команда, отобразившая на экране содержимое файла `/etc/motd`.

При необходимости выполнять вторую команду только в случае неудачи первой следует использовать две вертикальные черты ||

Пример:

```
$ ls nofile || echo "Net EGO"
ls: nofile: No such file or directory
Net EGO
```

Примечание: Данный пример демонстрирует, что вторая команда в цепочке `echo "Net EGO"` *была выполнена, так как первая команда* `ls nofile` *такой файл не обнаружила и закончилась с ошибкой.*

1.8. Сценарии оболочки.

Сценарий оболочки представляет собой текстовый файл, который состоит из системных и встроенных команд и синтаксических конструкций встроенного в оболочку языка программирования.

Сценарии предназначены для автоматизации выполнения рутинных задач.

Оболочка последовательно интерпретирует и выполняет команды, заданные в сценарии.

Примечание: Эти же команды могут быть выполнены простым последовательным вызовом их в командной строке оболочки.

Для файлов сценариев оболочки Bash принято использовать суффикс `.sh`.

Сценарии оболочки могут быть исполнены двумя различными путями:

Имя файла сценария можно указать в качестве аргумента командной строки при запуске оболочки, на языке которой написан данный скрипт. В этом случае файл сценария должен быть доступен для чтения.

Сценарий можно запустить так же, как и обычные системные команды. Файл сценария в этом случае должен быть доступен для чтения и исполнения.

Пример:

Запуск сценария путем явного вызова оболочки и указания имени сценария в качестве аргумента:

```
$ bash myscr1.sh
Privet!
```

При отладке сценариев полезны опции `-v` и `-x` bash.

Опция `-v` переводит оболочку в режим подробного информирования о работе. В этом режиме отображаются команды сценария перед их интерпретацией.

Опция `-x` отображает результаты интерпретации команд.

Для автоматического запуска требуемой оболочки в случае, если сценарий запускается как команда, в первой строке сценария должна находиться строка с полным именем оболочки:

Пример:

```
$ cat myscr1.sh
#!/bin/bash
echo 'Privet!'
```

Примечание: Строка `#!/bin/bash` *указывает с помощью какой оболочки должен быть интерпретирован и выполнен сценарий. Эта директива должна находиться в файле сценария в первой строке. Запуск сценария при неявном вызове оболочки возможен, если на файл сценария имеются разрешения для чтения и исполнения:*

```
$ chmod a+rx myscr1.sh
$ ./myscr1.sh
Privet!
```

Примечание: Обратите внимание на то, что нахождение исполняемого файла в текущем каталоге не означает возможности его запуска без указания пути к нему.

При необходимости выполнения сценария в контексте вызывающего сценария (оболочки) необходимо использовать команду точка (`.`), которая называется inline подстановкой.

Пример:

```
$ . .bashrc
```

Примечание: Если просто вызвать сценарий из другого сценария, то вызываемый сценарий будет исполнен в собственной оболочке. То есть переменные, значения которых были установлены в вызываемом сценарии, не будут известны в вызывающем сценарии. Inline подстановка часто используется для считывания в теле сценария переменных, заданных в другом файле.

Пример:

```
$ cat myscr2rc
VAR1='Snova Privet!'
$ cat myscr2.sh
#!/bin/bash
. myscr2rc
echo $VAR1
$ ./myscr2.sh
Snova Privet!
```

Примечание: В этом примере в коде сценария `myscr2.sh` *была выполнена inline подстановка содержимого файла* `myscr2rc`*, в котором была установлена переменная* `VAR1`.

1.9. Командная подстановка.

Командная подстановка (command substitution) позволяет результат выполнения одной команды поместить в указанном месте командной строки другой команды.

Механизм подстановки команд позволяет передавать в командную строку результаты выполнения заключенной в скобки команды, перед которыми должен стоять символ `$`.

Вместо доллара и скобок можно заключить выражение в обратные кавычки ``.

Пример: Предположим, что необходимо вывести на экран такую строку с информацией о первичной группе текущего пользователя: “My UID is ...”, где вместо многоточия должен быть выведен UID пользователя. Одним из способов решения этой задачи является получение UID пользователя с помощью команды `id -u`**, и подстановки этого значения вручную в качестве аргумента команды** `echo`**. Однако, ясно, что это весьма несовершенный способ. Можно объединить эти две команды в одну с помощью командной подстановки.**

```
$ echo "My UID is `id -u`"
My UID is 501
```

Примечание: Взгляните, команда `echo` *имеет единственный аргумент – строку, заданную в двойных кавычках. Внутри кавычек находится команда* `id -u`*, заключенная в обратные кавычки, что интерпретируется Bash как командная подстановка.*

Пример: Ниже приведен другой пример использования командной подстановки с использованием доллара и круглых скобок.

```
$ ls $(cat /etc/shells)
ls: /bin/ksh: No such file or directory
ls: /bin/zsh: No such file or directory
/bin/bash  /bin/csh  /bin/sh  /bin/tcsh
```

Примечание: Приведенная выше команда позволила определить, имеются ли в системе все оболочки, зарегистрированные в файле `/etc/shells`. *Для этого был использован механизм командной подстановки, где строки, выведенные командой* `cat` *из файла* `/etc/shells`, *содержащие пути к зарегистрированным оболочкам, были подставлены как аргументы команды* `ls`. *По результату работы команды видно, что оболочек* `ksh` *и* `zsh` *в системе нет, хотя в файле* `/etc/shells` *эти оболочки указаны.*

Очень удобно использовать результат командной подстановки для назначения его в качестве значения переменной.

Пример:

```
$ SYSTEM_START=`who -b`
$ echo $SYSTEM_START
system boot Oct 3 20:06
```

Примечание: В этом случае была определена переменная `SYSTEM_START`, *которая в результате подстановки результата работы команды* `who -b`, *получила в качестве значения строку с датой загрузки системы.*

1.10. Вычисление арифметических выражений в командной строке.

В командной строке можно вычислять выражения, помещая выражения либо в квадратные скобки, либо в двойные круглые, перед которыми должен стоять символ `$`,

Результаты выражений можно передавать как аргумент какой-либо команде или назначать переменной.

Синтаксис с квадратными скобками считается устаревшим.

Пример:

```
$ echo $((192*512))
98304
```

Примечание: В этом примере вычислено значение выражения `192*512`. *Команда* `echo` *выводит в стандартный поток вывода строку, указанную в качестве аргумента.*

Пример: Ниже приведен пример назначения результата арифметического выражения переменной.

```
$ V1=5
$ V2=$(($V1/2))
$ echo $V2
2
```

Примечание: Переменной `V1` *присвоено значение* `5`*. Далее производится присвоение переменной* `V2` *результата деления значения* `V1` *на два. В силу того, что в Bash возможно вычисление только целочисленных выражений, то в результате значением стало* `V2` *число* `2`.

1.11. Шаблоны подстановки и перечисление.

Для обработки одновременно нескольких файлов в качестве аргумента команды можно указать требуемую группу файлов, используя символы подстановки (шаблона).

Символы подстановки позволяют оболочке произвести поиск файлов, имена которых удовлетворяют шаблону, и подставить их в качестве аргументов файловой команде.

В качестве команды для испытания шаблонов подстановки можно использовать команду `echo`, которая просто выводит на экран то, что ей задано в качестве аргумента.

Символ звездочка `*` является шаблоном для любого количества любых символов в именах файлов и даже для их отсутствия. Единственный символ, который не удовлетворяет этому шаблону – лидирующая точка в именах скрытых файлов.

Примечание: Таким образом, подставив звездочку в качестве аргумента команде `echo`*, мы увидим в результате либо саму звездочку, если в каталоге не скрытых файлов нет, либо оболочка подставит команде echo имена всех файлов в каталоге в командную строку в качестве аргументов и можно будет увидеть список этих файлов.*

Пример:

```
$ echo *
f1 f2 f3 f4
$ rm *
$ echo *
*
```

Примечание: В данном примере приводится некоторый каталог, в котором есть файлы, имена которых подставляются Bash в качестве аргументов команде `echo`*, так как они удовлетворяют шаблону* `*`*. Затем, с помощью того же шаблона все эти файлы из каталога удаляются. После чего эксперимент с командой* `echo` *повторяется. На этот раз Bash не может подставить вместо звездочки имена файлов, поэтому команда* `echo` *выводит звездочку.Также существуют скрытые файлы. Имена этих файлов начинаются с точки. Шаблоном для имен таких файлов будет являться конструкция* `.*` .

Звездочку можно использовать в любом месте строки – части имен файлов, для подстановки которых и строится шаблон.

Пример:

```
$ echo *
f1 f2 s s1 s2
$ echo s*
s s1 s2
```

Примечание: Обратите внимание, что в каталоге находится пять файлов с именами `f1`*,* `f2`*,* `s`*,* `s1` *и* `s2`*. Команда* `echo s*` *выводит на экран имена только тех файлов, которые начинаются с буквы* `s` *и файл* `s`*.*

Символ `?` заменяет один символ в имени файла, который должен находиться в той позиции, где находится знак вопроса.

Пример:

```
$ echo s?
s1 s2
```

Примечание: Легко заметить, что файл `s` *не удовлетворяет заданному шаблону, поэтому Bash не подставляет его как аргумент команде* `echo`*.*

Можно еще более сузить диапазон поиска имен файлов, используя набор символов, заключенных в квадратные скобки. Применение такого шаблона обозначает, что в указанном месте должен находиться один любой символ из заданного множества.

Диапазон символов в наборе указывается через тире.

Примечание: `[0-9]` *– шаблон подходит для любых цифр, а* `[a-zA-Z]` *– шаблон для букв английского алфавита в верхнем и нижнем регистрах.*

Пример: Предположим, например, что требуется подставить в качестве аргументов имена всех файлов, которые начинаются либо с буквы `f`, либо с `s`, и имеют длину два символа. Для этого подойдет шаблон `[fs]?`.

```
$ echo [fs]?
f1 f2 s1 s2
```

Пример: Шаблону `?a[a-h]*` удовлетворяют имена файлов, начинающихся с любого символа, второй символ которых должен быть `a`, а третий символ должен находиться в диапазоне от `a` до `h`. Далее могут следовать любые символы.

С помощью шаблона `[! ]` можно указать множество символов (внутри квадратных скобок), которые не должны встречаться в именах файлов.

Пример:

```
$ echo *[!0-9]
s
```

Примечание: Такому шаблону здесь удовлетворяет единственный файл с именем `s`, так как шаблон требует, чтобы в конце имени файла не находилась цифра.

Очень удобен, хотя и не относится к шаблонам, механизм перечислений Bash.

Он позволяет задать с помощью фигурных скобок множество вариантов, которое должна перебрать оболочка, составляя последовательно все возможные варианты строк с использованием подстрок, находящихся в фигурных скобках.

Перечисление может быть с помощью запятых, например {a,b,c,d,e} или диапазона значений, например: {a..e}.

При использовании перечислений в командную строку вставляются все варианты перечисления, в то время как при использовании шаблонов вставляются все соответствующие шаблону имена файлов.

Пример:

```
$ echo s{1,2}
s1 s2
$ echo s1 s2
s1 s2
$ echo s{1..9}
s1 s2 s3 s4 s5 s6 s7 s8 s9
```

```
$ echo s{1..9..2}
s1 s3 s5 s7 s9
```

Примечание: Первые две команды совершенно эквивалентны. Bash подставляет все варианты, перечисленные в фигурных скобках, к строке, находящейся вне скобок и подставляет получившиеся строки, как аргументы командной строки. Следующие две команды показывают возможные переборы последовательных значений. В последней команде задан шаг перебора равный 2.

Можно использовать несколько перечислений одновременно.

Пример:

```
$ echo {s,f}{1,2}
s1 s2 f1 f2
```

Можно использовать перечисления в виде вложенных конструкций.

Пример:

```
$ echo {s,{f1,f2}}
s f1 f2
```

В фигурных скобках допускается использование шаблонов.

Пример:

```
$ echo {s,f?}
s f1 f2
```

1.12. Условное исполнение команд.

Операторы условного исполнения команд `&&` и `||` позволяют исполнять команды в зависимости от успешности выполнения предыдущих команд.

Если оператор `&&` установлен между двумя командами, то вторая из них будет исполнена только в случае успеха предыдущей.

Пример:

```
$ ls /tmp &> /dev/null && cd /tmp
$ pwd
/tmp
```

Примечание: В этом примере первая команда (`$ ls /tmp &> /dev/null`) используется лишь для проверки существования каталога, переход в который осуществляет вторая команда (`cd /tmp`) при условии успешного исполнения первой.

Пример: Более изящный вариант выполнения этой же задачи заключается в использовании команды `test` для проверки существования целевого каталога:

```
$ [ -d /tmp ] && cd /tmp
$ pwd
/tmp
```

Примечание: Команда `test` в этом примере проверяет существование каталога. В случае существования этого каталога осуществляется переход в него.

Если между командами установлен оператор `||`, то вторая команда будет выполнена в случае неудачного завершения работы первой команды.

Пример: требуется перейти в каталог `d1`. В случае его отсутствия этот каталог необходимо создать и перейти в него. Эту задачу можно решить следующим образом:

```
$ [ -d d1 ] || mkdir d1 ; cd d1
$ pwd
/tmp/d1
```

Примечание: Команда `test` проверила наличие каталога. Если он отсутствует, то этот каталог создается. Далее осуществляется переход в заданный каталог.

Оболочка предоставляет также специальный оператор условного выполнения `if`

Пример: предположим, что требуется обеспечить выход из некоторого сценария с ошибкой если в командной строке установлено отличное от единицы число аргументов командной строки. При этом команда должна сообщать об ошибке и выдавать подсказку о правильном варианте ее использования. Ниже приведен текст программы:

```
#!/bin/bash
if [ $# -ne 1 ]
then
        cat <<- ERR
                Не хватает аргументов.
                Использование:
                if.sh file
                Аргумент file должен быть обычным файлом.
        ERR
        exit 1
fi
```

```
ls -l $1
```

Примечание: Команда `if` *исполняет блок команд после команды* `then` *только в случае получения нулевого кода возврата команды, указанной в качестве ее аргумента. В данном случае аргумент команду* `if` *- это команда* `test`*, которая в данном случае проверяет неравенство количества аргументов, переданных сценарию, единице. Если количество аргументов не равно единице, то выполняется блок операторов после* `then` *до команды* `fi`*, заканчивающей* `if`*. Конструкция* `cat <<- ERR` *- это "here document". Она позволяет указывать целый блок данных непосредственно в теле скрипта, передавая его в данном случае в стандартный поток ввода команде* `cat` *для вывода на экран. Блок данных ограничен строкой* `ERR`*. В этом примере используется оператор* `<<-` *вместо* `<<` *для игнорирования табуляций перед блоком данных.*

Ниже приведены результаты испытания программы:

```
$ ./if.sh
Не хватает аргументов.
Использование:
if.sh file
Аргумент file должен быть обычным файлом.
$ ./if.sh if.sh
-rwxr--r--    1 aberes   aberes        172 Окт  8 19:34 if.sh
```

Команда `if` допускает использование команды `elif` для выполнения дополнительной проверки.

Пример:

```
#!/bin/bash
if [ $# -ne 1 ]
then
        cat <<- ERR
                Не хватает аргументов.
                Использование:
                if.sh file
                Аргумент file должен быть обычным файлом.
        ERR
        exit 1
elif [ ! -f $1 ]
then
        echo -n 'Тип файла '
        file $1
        exit 1
fi
ls -l $1
```

Примечание: Если в качестве аргумента задан не обычный файл, то выводится тип этого файла.

```
$ ./if.sh .
Тип файла .: directory
```

В команде `if` можно также использовать `else` для указания блока команд, которые должны исполняться в случае, если предыдущие проверки не закончились успехом.

Пример: программу можно модифицировать следующим образом:

```
#!/bin/bash
if [ $# -ne 1 ]
then
        cat <<- ERR
                Не хватает аргументов.
                Использование:
                if.sh file
                Аргумент file должен быть обычным файлом.
        ERR
        exit 1
elif [ ! -f $1 ]
then
        echo -n 'Тип файла '
        file $1
        exit 1
else
    ls -l $1
fi
```

1.13. Оператор case.

Оператор `case позволяет` проверить значение, содержащееся в переменной, на совпадение с заданными шаблонами.

Метасимволы шаблонов сравнения для `case` такие же, как метасимволы шаблонов для имен файлов.

Синтаксис команды `case` в общем виде таков:

```
case слово in
    шаблон1 )
      команды
      ;;
    шаблон2 )
      команды
      ;;
esac
```

Сравнение слова с шаблонами производится последовательно. Как только найдется совпадение выполняются соответствующие команды и дальнейших проверок не выполняется.

Пример: Допустим, что в текущем каталоге располагаются символические ссылки на сценарии запуска служб GNU/Linux. Требуется написать сценарий, который будет запускать службы, передавая сценарию запуска службы аргумент `start`, если в качестве аргумента будет использована символическая ссылка, начинающаяся с букв `S` или `s`. Если же ссылка будет начинаться с букв `K` или `k`, то служба должны останавливаться и, следовательно, их сценариям должен передаваться аргумент `stop`.

Ниже приведен текст сценария:

```
$ cat case.sh
#!/bin/bash
[ $# -ne 1 ] && exit 1;
FIRST=`echo $1 | cut -c1`
case $FIRST in
        [Ss] )
                echo "Стартую $1"
                ./$1 start
                ;;
        K|k )
                echo "Останавливаю $1"
                ./$1 stop
                ;;
        * )
                echo "Статус $1"
                ./$1 status
                ;;
esac
```

Примечание: В этом сценарии в начале проверяется количество аргументов. Если оно не равно 1, то осуществляется выход с ошибкой. Далее в переменную `FIRST` *помещается первая буква аргумента командной строки (предполагается, что это имя символческой ссылки).*

Команда `case` *проверяет соответствие содержащейся в переменной* `FIRST` *буквы шаблону* `[Ss]`*. Этот шаблон обозначает множество, аналогично с обычными файловыми шаблонами. То есть, этому шаблону удовлетворяют либо* `S`*, либо* `s`*.*

Если буква, содержащаяся в `FIRST` *удовлетворяет этому шаблону, то выводится сообщение о запуске службы и она запускается.*

Для остановки службы (в качестве примера) используется иной шаблон, имеющий в данном случае тот же смысл: либо `K`*, либо* `k`*. Вертикальная черта обозначает "или". Однако, ее можно применять даже для целых строк или шаблонов.*

Если содержимое переменной `FIRST` *не совпадает и с этим шаблоном, то выводится информация о текущем статусе службы.*

Ниже приведен пример работы этого сценария:

```
# ls *postfix
K11postfix  postfix  S89postfix
# ./case.sh S89postfix
Стартую S89postfix
 * Starting postfix...                  [ ok ]
# ./case.sh postfix
Статус postfix
 * status:  started
# ./case.sh K11postfix
Останавливаю K11postfix
 * Stopping postfix...                  [ ok ]
```

1.14. Циклы.

Для программирования в оболочке доступны три вида циклов:

`for` - этот оператор позволяет создавать циклы типа перебора значений;

`while` - цикл, выполняющийся до тех пор, пока истинно некоторое условие;

`until` - цикл, выполняющийся до тех пор, пока некоторое условие ложно.

Пример: Допустим, что имеется некоторый набор значений, которые должны быть последовательно присвоены некоторой переменной, требующейся для произведения каких-либо операций. В этом случае удобно использовать команду `for`**.**

Пусть, например, требуется в цикле вывести права доступа к нескольким каталогам:

```
#!/bin/bash
for DIR in /etc /tmp /var
do
        echo -n "Права доступа к $DIR "
        ls -ld $DIR | cut -c2-11
done
```

Примечание: В этом сценарии переменная `DIR` *последовательно принимает три значения:* `/etc`*,* `/tmp` *и* `/var`*. Это достигается с помощью команды* `for`*, в которой*

список значений указан после `in` . *Тело цикла начинается после команды* `do` *и ограничивается* `done` .

Ниже показаны результаты работы сценария:

```
$ ./for.sh
Права доступа к /etc rwxr-xr-x
Права доступа к /tmp rwxrwxrwt
Права доступа к /var rwxr-xr-x
```

Если в команде `for` не указан список после директивы `in` , то переменная цикла будет последовательно принимать значения, соответствующие аргументам командной строки.

Пример: Изменим предыдущий сценарий так, чтобы имена каталогов можно было бы задавать в качестве аргументов в командной строке:

```
#!/bin/bash
[ $# -lt 1 ] && exit 1
for DIR
do
        if [ -d $DIR ]; then
                echo -n "Права доступа к $DIR "
                ls -ld $DIR | cut -c2-11
        elif [ ! -e $DIR ]; then
                echo "$DIR не существует"
        fi
done
```

Примечание: В этом сценарии переменная `DIR` *последовательно принимает значения аргументов командной строки, так как в команде* `for` *не задан список значений. Ниже приведен пример работы такого сценария:*

```
$ ./for.sh /usr /rsu /root
Права доступа к /usr rwxr-xr-x
/rsu не существует
Права доступа к /root rwx------
```

Операторы `while` и `until` удобно использовать для организации итерационных (со счетчиком) циклов.

Пример: для вывода в цикле списка значений от `1` до `10` можно написать такой сценарий:

```
#!/bin/bash
i=1
```

```
while [ $i -le 10 ]; do
        echo $i
        i=$((i+1))
done
```

Этот скрипт последовательно выводит значения:

```
$ ./while.sh
1
2
3
4
5
6
7
8
9
10
```

Цикл `while` работает до тех пор, пока команда, указанная в качестве ее аргумента, возвращает успешный код завершения.

`Цикл until` работает пока команда - аргумент заканчивается неудачей.

Пример: последовательно удалим из полного доменного имени узла (FQDN) подстроки до первой точки. Сначала должно быть выведено полное имя узла, затем домен, затем родительский домен, и так далее, пока строка не станет пустой.

```
#!/bin/bash
HN=`hostname`
until [ -z $HN ]; do
        echo $HN
        HN=`echo -n $HN | tr '.' '\n' | sed '1d' | tr '\n' '.'`
done
```

Примечание: В сценарии используется переменная `HN`*, которой в начале присваивается значение - доменное имя узла. Затем из него в цикле удаляется подстрока от начала строки до первой встретившейся точки. Для этого в строке - имени узла точки сначала заменяются на переводы строки, затем удаляется первая строка, и в заключение переводы строки снова заменяются на точки. Цикл* `until` *работает в этом примере до тех пор, пока содержимое переменной* `HN` *не станет пустой строкой.*

Результаты работы сценария:

```
$ hostname
nechto.zamislovatoe.tmn.ru
$ ./until.sh
nechto.zamislovatoe.tmn.ru
```

```
zamislovatoe.tmn.ru
tmn.ru
ru
```

1.15. Функции.

Функции, представляют собой именованные блоки кода, написанные на языке оболочки.

Синтаксис описания функции:

```
function имя()
{
    команды
}
```

Код функции описывают в начале сценария до первого вызова функции.

Для вызова функции достаточно просто указать ее имя.

Пример: Для демонстрации использования функций оболочки модифицируем программу `if.sh` из параграфа об условном выполнении команд. В этой программе можно, например, вынести часть кода, связанную с выводом сообщения об ошибке, в функцию.

```
$ cat if.sh
#!/bin/bash
function err_msg()
{
        cat <<- ERR
                Не хватает аргументов.
                Использование:
                if.sh file
                Аргумент file должен быть обычным файлом.
        ERR
}
if [ $# -ne 1 ]
then
        err_msg
        exit 1
elif [ ! -f $1 ]
then
        echo -n 'Тип файла '
        file $1
        exit 1
fi
ls -l $1
```

Пример работы сценария `if.sh` :

```
$ ./if.sh
Не хватает аргументов.
Использование:
if.sh file
Аргумент file должен быть обычным файлом.
$ ./if.sh /etc/passwd
-rw-r--r--      1  root        root              2033  Авг  19  23:24
/etc/passwd
```

Для передачи аргументов в функцию их указывают после имени функции, разделяя пробелами.

В теле функции обращение к переданным аргументам производится с помощью позиционных параметров.

Пример: модификация `if.sh` , демонстрирующая передачу аргументов в функцию.

```
$ cat if.sh
#!/bin/bash
function err_msg()
{
        echo "Ошибка: $1"
        cat <<- ERR
                Не хватает аргументов.
                Использование:
                if.sh file
                Аргумент file должен быть обычным файлом.
        ERR
}
if [ $# -ne 1 ]
then
        err_msg '001'
        exit 1
elif [ ! -f $1 ]
then
        echo -n 'Тип файла '
        file $1
        exit 1
fi
ls -l $1
```

Примечание: Здесь в функцию передается строка - код ошибки. В теле функции этот код считывается из позиционного параметра.

Пример работы программы:

```
$ ./if.sh
Ошибка: 001
Не хватает аргументов.
Использование:
if.sh file
Аргумент file должен быть обычным файлом.
```

Функции, описанные в отдельных файлах, считываются с помощью inline подстановки.

Пример: Перенесем описание функции `err_msg` в файл `myfunctions`. Ниже приводятся содержимое файла `myfunctions` и измененный текст сценария `if.sh` :

```
$ cat myfunctions
function err_msg()
{
        echo "Ошибка: $1"
        cat <<- ERR
                Не хватает аргументов.
                Использование:
                if.sh file
                Аргумент file должен быть обычным файлом.
        ERR
}
$ cat if.sh
#!/bin/bash
. ./myfunctions
if [ $# -ne 1 ]
then
        err_msg '001'
        exit 1
elif [ ! -f $1 ]
then
        echo -n 'Тип файла '
        file $1
        exit 1
fi
ls -l $1
```

Примечание: В сценарии `if.sh` *inline считывается содержимое файла* `myfunctions` *и функция* `err_msg` *, описанная в нем становится доступной для использования в сценарии.*

Глава 2. Система инициализации в РЕД ОС

2.1. Инициализация системы с помощью демона systemd

В стандартном подходе к инициализации Unix-подобных систем используется демон init, который запускается первым и получает PID=1, затем init запускает различные сервисы, в том числе сценарии инициализации rc

Такой подход имеет ряд ограничений:

Нет обратной связи между сервисом и демоном init. Демон init может только отслеживать работает или не работает запущенный им сервис (скрипт).

Нет возможности просмотреть результат загрузки сервисов после старта системы.

Монтирование, запуск служб, запуск сетевых сокетов, запуск виртуальных машин и т. д. в системах с init инициализацией это отдельные задачи, которые решаются разными средствами.

Не возможно параллельно запустить несколько сервисов, отложить запуск или запустить в фоновом режиме службу.

Systemd предлагает универсальное средство для инициализации, управления и настройки ОС.

Как правило программа init является ссылкой на systemd в таких системах.

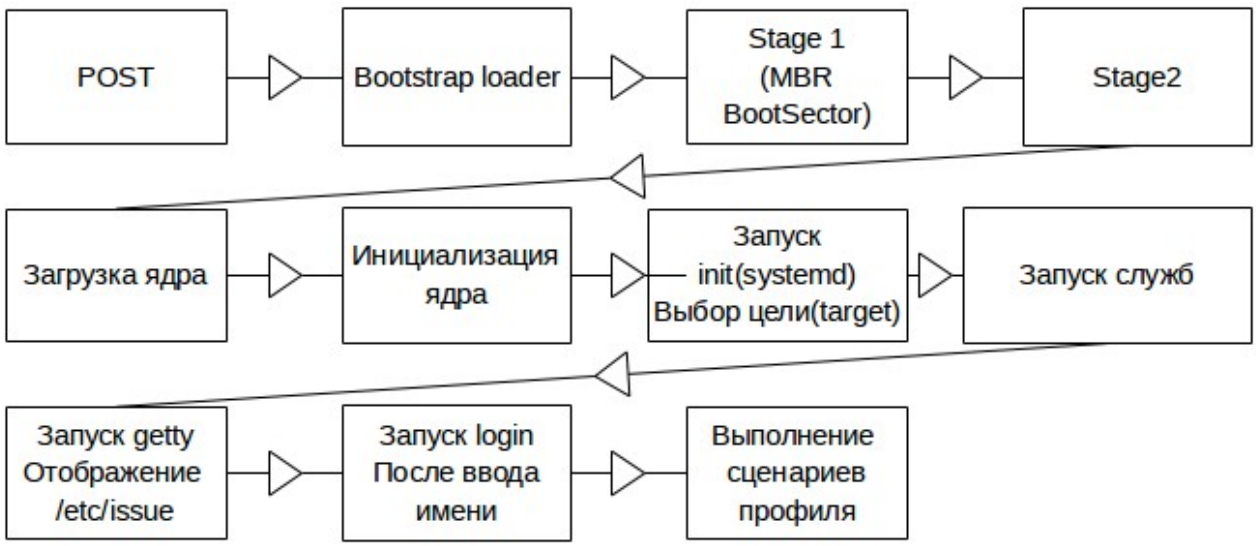


Рисунок 1: Инициализация системы с помощью демона systemd

В systemd отказались от понятия уровень исполнения, вместо этого используется понятие цель (target).

Цель — некоторое состояние в которое система должна прийти после инициализации нужных сервисов.

Не все цели предназначены для конечной загрузки.

Основополагающее понятие systemd — unit.

Unit — отдельный элемент для описания и управления в systemd.

С каждым юнитом может быть связан конфигурационный файл, который должен находиться в одном из каталогов.

/etc/systemd/system

/run/systemd/system

/usr/lib/systemd/system

Каталоги перечислены в порядке убывания приоритета.

Локальные изменения следует проводить в каталоге /etc/systemd/system

Юниты делятся на типы:

target — цель, используется для описания некоторого состояния системы, в которое мы должны попасть;

service — описывает процесс, который контролирует systemd;

socket — сетевой сокет, для каждого сокета должен существовать соответствующий ему сервис;

timer — описывает какой сервис должен запуститься в определенное время;

device — создает файл устройства;

mount — точка монтирования, дополнительная к /etc/fstab;

automout — то же, что mount, но монтирование производится по требованию;

swap — область подкачки;

slice — описание виртуальной машины или контейнера;

snapshot — юнит без юнит-файла, позволяет создать копию конфигурации и затем вернуться к ней;

scope — юнит без юнит-файла, создается программно для группировки процессов.

2.2. Управление юнитами systemd

Команда systemctl управляет работой демона systemd

systemctl list-dependencies — показать от чего зависит юнит.

systemctl list-units — показывает список известных (включенных) юнитов.

systemctl list-unit-files — показывает список всех юнитов, в том числе

выключенных, и их состояние.

systemctl get-default — показывает цель загрузки по умолчанию.

systemctl set-default target — устанавливает цель загрузки по умолчанию.

systemctl {enable|disable} unit — включение (разрешение на запуск) или выключение (запрет) юнита.

Примечание: некоторые сервисы все равно запускаются несмотря на запрет, т. к. это могут потребовать зависимые сервисы.

systemctl {start|stop} unit — запуск или остановка юнита.

Когда запускается systemd он определяет цель по умолчанию (default.target), заданную в настройках системы или во время загрузки ядра (параметр ядра systemd.unit=нужная_цель.target)

```
$ ls -l /etc/systemd/system/default.target
lrwxrwxrwx    1    root    root    40    Nov    27         2016
/etc/systemd/system/default.target                              ->
/usr/lib/systemd/system/graphical.target
$ systemctl get-default
graphical.target
```

После определения цели для загрузки определяется последовательность запуска юнитов:

```
$ systemctl list-dependencies | grep target
default.target
● └─multi-user.target
●   ├─basic.target
●   │ ├─paths.target
●   │ ├─slices.target
●   │ ├─sockets.target
●   │ ├─sysinit.target
●   │ │ ├─cryptsetup.target
●   │ │ ├─local-fs.target
●   │ │ └─swap.target
●   │ └─timers.target
●   ├─getty.target
●   ├─nfs-client.target
●   │ └─remote-fs-pre.target
●   └─remote-fs.target
●     └─nfs-client.target
●       └─remote-fs-pre.target
$ systemctl list-dependencies swap.target
swap.target
●           ├─dev-disk-by\x2duuid-1496f7c8\x2d3bb8\x2d4395\x2d97f3\
x2d31f8a6083d9f.swap
●           ├─dev-disk-by\x2duuid-4cc362d5\x2db46d\x2d4e20\x2d8d75\
```

```
x2dc4225eba0ea9.swap
● └─dev-zram0.swap
```

Подробнее это можно почитать в man 7 bootup.

Когда все нужные юниты будут запущены, то система считается работающей:

```
$ systemctl is-system-running
running
```

Если имеются проблемы, то мы увидим состояние degraded. Чтобы выяснить, что не так можно выполнить команду systemctl --failed:

```
$ systemctl is-system-running
degraded
$ systemctl --failed
  UNIT          LOAD   ACTIVE SUB    DESCRIPTION
● kdump.service loaded failed failed Crash recovery kernel arming
LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization
of SUB.
SUB    = The low-level unit activation state, values depend on
unit type.
1 loaded units listed. Pass --all to see loaded but inactive
units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

Попробуем понять в чем дело:

```
$ systemctl status kdump.service
● kdump.service - Crash recovery kernel arming
   Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled;
vendor preset: enabled)
     Active: failed (Result: exit-code) since Fri 2021-02-05
15:08:54 +05; 1h 57min ago
   Process: 957 ExecStart=/usr/bin/kdumpctl start (code=exited,
status=1/FAILURE)
 Main PID: 957 (code=exited, status=1/FAILURE)
Feb 05 15:08:54 cent8-stream systemd[1]: Starting Crash recovery
kernel arming...
Feb 05 15:08:54 cent8-stream kdumpctl[957]: kdump: No memory
reserved for crash kernel
Feb 05 15:08:54 cent8-stream kdumpctl[957]: kdump: Starting kdump:
[FAILED]
Feb 05 15:08:54 cent8-stream systemd[1]: kdump.service: Main
process exited, code=exited, status=1/FAILURE
Feb 05 15:08:54 cent8-stream systemd[1]: kdump.service: Failed
with result 'exit-code'.
Feb 05 15:08:54 cent8-stream systemd[1]: Failed to start Crash
```

```
recovery kernel arming.
```

Теперь у нас есть причина: No memory reserved for crash kernel. Чтобы исправить мы можем или установить параметр ядра `crashkernel=`auto или просто запретить запуск службы при старте системы:

```
$ sudo systemctl disable kdump
Removed /etc/systemd/system/multi-user.target.wants/kdump.service.
$ sudo systemctl stop kdump
```

В systemd есть три состояния у юнита:

enabled – стартовать зависимость автоматически

disabled – не запускать автоматически, но вручную можно

masked – никогда не запускать юнит.

Systemd имеет свою особую систему журналов работы, как, собственно, systemd, так и всех служб им запущенных.

Просмотр журналов systemd производится командой journalctl.

Примечание: Описание команды будет в главе посвященной системным журналам.

2.3. Создание и использование собственных юнитов

Юниты можно создавать самостоятельно для дальнейшего использования в системе. Для этого вам могут понадобиться следующие команды:

systemctl cat имя_юнита - просмотр юнита;

systemctl --full edit имя_юнита - редактирование юнита;

systemctl --full --force edit имя_юнита - создание юнита.

Пример: Рассмотрим содержимое юнита на примере cups.service

```
sudo systemctl cat cups.service
# /etc/systemd/system/cups.service
[Unit]
Description=CUPS Scheduler
Documentation=man:cupsd(8)
After=network.target sssd.service ypbind.service nslcd.service
Requires=cups.socket

[Service]
ExecStart=/usr/sbin/cupsd -l
Type=notify
Restart=on-failure

[Install]
Also=cups.socket cups.path
```

```
WantedBy=printer.target
```

Юниты состоят из нескольких секций. Количество и именования секций зависят от типа юнита. Внутри секций находятся параметры юнита и их значения.

*Примечание: Чтобы узнать какие секции есть у конкретного типа юнита используйте команду **man systemd.unit**, где вместо unit укажите тип юнита.*

Пример:

```
man systemd.service
```

Разберем типичное содержимое юнита типа service.

Раздел [Unit]:

Description - описание юнита.

Wants - слабые зависимости от других юнитов. Перечисленные юниты при их наличии будут запущены вместе с текущим. Их отсутствие не влияет на запуск исходного юнита.

Requires - сильная зависимость от других юнитов. Перечисленные юниты при их наличии будут запущены вместе с текущим. При их отсутствии исходный юнит не запустится.

After - юнит будет запущен после запуска указанных в этом параметре юнитов.

Before - юнит будет запущен перед запуском указанных в этом параметре юнитов.

Conflicts - перечень юнитов которые нельзя запускать с данным юнитом. Если запустить юнит указанный в списке - будет остановлен текущий.

Раздел [Service]:

Type - тип запуска службы

ExecStart - указывает полный путь до программы/сценария с параметрами для запуска данной службы

ExecStop - указывает полный путь до программы/сценария с параметрами для останова данной службы

ExecReload - указывает команду, необходимую для того, чтобы служба перечитала конфигурацию

Restart - будет ли служба перезапущена если процесс службы завершится

Раздел [Install]:

WantedBy - создает символическую ссылку в каталоге .wants каждого из

перечисленных юнитов. Используется при выполнении команды systemctl enable. В итоге юнит будет запущен, когда будет запущен указанный

RequiredBy - тоже самое что и WantedBy, но для каталога .requires

Alias - дополнительные имена для данного юнита. Такие имена можно использовать в командах systemctl (кроме enable)

Also - дополнительные юниты для установки или удаления данного юнита из автозапуска. Если для текущего юнита будет использована команда enable/disable - то она будет выполнена и для указанных в данном параметре юнитов.

После написания юнита типа service обычно стоит его запустить при помощи команды systemctl start имя_юнита и, при необходимости, добавить в автозагрузку при помощи команды systemctl enable имя_юнита

Глава 3. Работа с дисками в РЕД ОС

3.1. Таблица разделов.

Файловые системы можно классифицировать по различным признакам, далее будет рассмотрен вариант классификации на журналируемые и нежурналируемые файловые системы.

К журналируемым файловым системам относятся те, которые сохраняют список изменений, осуществляемых с файловой системой, перед фактическим их применением. Такие записи хранятся в отдельной части файловой системы, называемой журналом. Как только изменения файловой системы внесены в журнал, она применяет изменения к файлам или метаданным, а затем удаляет эти записи из журнала.

К нежурналируемым файловым системам относятся ext2 и FAT (*FAT12, FAT16, FAT32*). Ядро ОС их также поддерживает.

При перезагрузке компьютера программа монтирования может гарантировать целостность журналируемой файловой системы простой проверкой журнала на наличие ожидаемых, но не произведённых изменений и последующей записью их в файловую систему. Т.е. при наличии журнала в большинстве случаев системе не нужно проводить проверку целостности файловой системы. Соответственно, шансы потери данных в связи с проблемами в файловой системе значительно снижаются.

Для рабочих станций в РЕД ОС используется журналируемая файловая система ext4 — логическое продолжение ext3, которая является журналируемым вариантом системы ext2 — «материнской» файловой системы Linux.

В зависимости от указанных администратором настроек файловая система ext4 может работать в любом из трёх известных для журналируемых систем режиме:

- в режиме обратной связи — журналируются только метаданные;
- упорядочивание — журналируются только метаданные синхронно относительно данных;
- в режиме данных — журналируются как метаданные, так и данные.

Для серверного варианта РЕД ОС рекомендуется использовать файловую систему XFS, которая также является журналируемой.

Обе файловые системы — ext4 и XFS — являются не только

журналируемыми, но и:

•поддерживают большие объёмы дисковых накопителей (до 260 байт);

•поддерживают файлы большого размера (до 244 байт);

•используют механизм пространственной записи файлов, уменьшающий фрагментацию и повышающий производительность. Суть механизма заключается в том, что новая информация добавляется в конец области диска, выделенной заранее рядом с областью, занятой содержимым файла.

Важными отличиями представленных файловых систем является то, что XFS:

•может производить дефргаментацию «на лету»;

•имеет API ввода-вывода реального времени (для приложений жёсткого или мягкого реального времени, например, для работы с потоковым видео);

•имеет малые «накладные расходы» — размер служебных структур данных.

Ext4 в отличие от XFS менее агрессивно использует оперативную память при работе с файлами на носителе, поэтому её оптимально применять на вычислительных системах с небольшим объёмом ОЗУ.

При выборе файловой системы при создании/форматировании раздела рекомендуется определить, для чего будет использоваться этот раздел. Также следует обратить внимание на параметры процессора и оперативной памяти компьютера, и на основании этого принимать решение о необходимости выбора той или иной ФС.

РЕД ОС поддерживает и другие файловые системы, такие как FAT, NTFS, а также ISO-9660 для оптических накопителей, сетевые файловые системы.

3.2. Таблица разделов.

Большинство сценариев использования диска в Linux предполагает создание разделов.

Разделы на дисках создаются для логического разделения информации.

Linux может работать с диском и напрямую без создания разделов. Например при использовании в LVM (Logical Volume Manager).

В Linux в основном используются две схемы разбиения диска на разделы:

MBR (Master Boot Record) – используется для загрузки компьютера и описания разделов на диске.

GPT (GUID Partition Table) – новая схема разбиения дисков. Позволяет создать больше разделов на бóльших дисках.

Linux может создавать и использовать и другие типы разбиения диска, но они на практике почти не применяются: SGI и Sun.

Количество основных разделов в MBR от 1 до 4.

Обычно каждый раздел предназначен для размещения одной операционной системы, отдельной файловой системы, или, например, области размещения страниц подкачки.

MBR находится в нулевом секторе диска, а таблица разделов в нем указывает на начало, конец, размер и тип каждого раздела.

Тип раздела позволяет ОС, в которую он подключен, определить как работать с разделом. Или, другими словами, автоматизировать операции по подключению разделов на этапе загрузки.

Если вы ссылаетесь на разделы в каких-нибудь командах или утилитах, то тип раздела, как правило, игнорируется.

Примечание: Например если вы подключаете диск с «неизвестным» разделом в Windows, то вы даже удалить этот раздел не сможете стандартными средствами Windows. Linux при работе с разделами на тип реагирует «либерально», т. е. Вы можете вручную его всегда подключить или удалить. Но если тип будет не правильный, то во время загрузки нужный вам раздел будет проигнорирован. Например раздел, который не имеет тип fd не будет рассматриваться как часть RAID массива.

Для обеспечения возможности создания большого количества разных файловых систем на жестком диске Linux (как и Windows) поддерживает концепцию логических разделов.

Один из основных разделов (primary partition) может быть объявлен, как расширенный (`extended`). В расширенном разделе может быть создано неограниченное количество логических разделов (`logical partitions`).

Разбиение диска с помощью MBR имеет существенное ограничение (2 TB) на размер диска, который можно разбить. Это одна из причин почему был предложен новый метод деления диска на разделы — GPT (GUID Partition Table).

GPT — часть стандарта EFI.

В GPT для обратной совместимости сохраняется MBR, в котором создается один раздел на весь диск с типом 0xee.

После MBR в первом секторе содержится оглавление таблицы разделов. В оглавлении содержится информация о положении таблицы разделов, а также о количестве и размере записей.

Всего в GPT резервируется 128 записей.

GPT обеспечивает дублирование — оглавление и таблица разделов

записаны как в начале, так и в конце диска.

Теоретически, GPT позволяет создавать разделы диска размером до 9,4 ЗБ (9,4 × 1021 байт), в то время как MBR может работать только до 2,2 ТБ (2,2 × 1012 байт).

Для получения таблицы разделов на диске можно воспользоваться командой fdisk -l.

3.3. Создание разделов с использованием fdisk.

В Linux вы можете манипулировать разделами с помощь разнообразных утилит. Сред них основные:

fdisk
sfdisk
cfdisk
parted/gparted
blivet-gui

Интерактивная утилита `fdisk` позволяет оперировать с дисковыми разделами жестких магнитных дисков и обладает специальным набором собственных команд.

Утилита `sfdisk`, в отличие от `fdisk` , является утилитой для не интерактивного редактирования таблицы разделов на жестком диске.

Примечание: При неосторожном ее использовании легко можно утратить все данные на жестком диске, так как данные для редактирования таблицы разделов задаются в командной строке `sfdisk`.

Популярна также утилита `cfdisk`, которая позволяет редактировать таблицу разделов диска с помощью простого меню-образного интерфейса.

Утилита parted помимо создания и удаления разделов может перемещать разделы по диску. Может работать как интерактивно, так и не интерактивно. gparted — графическая утилита.

blivet-gui относительно новая графическая утилита для работы с разделами и файловыми системами.

Команда `fdisk` доступна только администратору. Для редактирования таблицы разделов на диске эту команду следует запустить в интерактивном режиме, указав файл устройства для требуемого жесткого диска в качестве аргумента команды:

Пример:

```
# fdisk /dev/sda
Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you decide to write
them.
Be careful before using the write command.
Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x806fa0ce.
Command (m for help): m
Help:
  DOS (MBR)
   a   toggle a bootable flag
   b   edit nested BSD disklabel
   c   toggle the dos compatibility flag
  Generic
   d   delete a partition
   F   list free unpartitioned space
   l   list known partition types
   n   add a new partition
   p   print the partition table
   t   change a partition type
   v   verify the partition table
   i   print information about a partition
  Misc
   m   print this menu
   u   change display/entry units
   x   extra functionality (experts only)
  Script
   I   load disk layout from sfdisk script file
   O   dump disk layout to sfdisk script file
  Save & Exit
   w   write table to disk and exit
   q   quit without saving changes
  Create a new label
   g   create a new empty GPT partition table
   G   create a new empty SGI (IRIX) partition table
   o   create a new empty DOS partition table
   s   create a new empty Sun partition table
```

Примечание: В этом примере команда `fdisk` *была выполнена с аргументом - файлом устройства первого SCSI диска. Далее была выполнена встроенная команда* `m`*, отобразившая список встроенных команд* `fdisk`*.*

Список основных команды утилиты `fdisk`:

`q` - завершение работы без сохранения изменений;

`l` - вывод списка возможных типов разделов;

g - преобразование в GPT диск;

`d` - удаление раздела (для удаления будет запрошен номер удаляемого раздела);

`n` - создание нового раздела;

`t` - установка типа вновь созданного раздела (для установки типа необходимо ввести номер типа раздела);

`a` - выбор активного раздела;

`w` – запись измененной таблицы разделов.

Встроенная команда `p` утилиты `fdisk` выводит информацию о таблице разделов на диске.

Пример:

```
Command (m for help): p
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x806fa0ce
```

Обычная последовательность работы с утилитой `fdisk` для создания нового раздела на жестком диске:

Выводится список существующих разделов на диске - команда `p`.

Удаляются ненужные разделы (при этом данные на них теряются безвозвратно) – команда `d`.

Создается новый раздел – команда `n`. При этом будет запрошены: тип раздела (`p` – первичный, `e` – расширенный, `l` – логический), номер раздела, первый и последний цилиндры раздела; при этом последний цилиндр нового раздела можно указать абсолютно или, после знака `+`, размер раздела в килобайтах (например, `+15000k`) или мегабайтах (`+1200M`).

Если новый раздел должен иметь тип, отличный от принятого по умолчанию (`83` – Linux Native), то необходимо указать тип раздела – команда `t` (получение списка возможных типов разделов – команда `L`).

Когда все требуемые разделы созданы, то необходимо сохранить изменения – команда `w`.

Пример:

```
Command (m for help): p
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x806fa0ce
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-41943039, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-41943039, default
41943039): +5G
Created a new partition 1 of type 'Linux' and of size 5 GiB.
Command (m for help): p
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x806fa0ce
Device     Boot Start      End  Sectors Size Id Type
/dev/sda1        2048 10487807 10485760   5G 83 Linux
Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): L
 0  Empty            24  NEC DOS          81  Minix / old Lin bf
Solaris
 1  FAT12            27  Hidden NTFS Win  82  Linux swap / So c1
DRDOS/sec (FAT-
 2  XENIX root       39  Plan 9           83  Linux           c4
DRDOS/sec (FAT-
 3  XENIX usr        3c  PartitionMagic   84  OS/2 hidden or  c6
DRDOS/sec (FAT-
 4  FAT16 <32M       40  Venix 80286      85  Linux extended  c7
Syrinx
 5  Extended         41  PPC PReP Boot    86  NTFS volume set da
Non-FS data
 6  FAT16            42  SFS              87  NTFS volume set db
CP/M / CTOS / .
 7  HPFS/NTFS/exFAT  4d  QNX4.x           88  Linux plaintext de
Dell Utility
 8  AIX              4e  QNX4.x 2nd part  8e  Linux LVM       df
BootIt
```

```
 9  AIX bootable     4f  QNX4.x 3rd part 93  Amoeba           e1
DOS access
 a  OS/2 Boot Manag 50  OnTrack DM       94  Amoeba BBT       e3
DOS R/O
 b  W95 FAT32       51  OnTrack DM6 Aux 9f  BSD/OS           e4
SpeedStor
 c  W95 FAT32 (LBA) 52  CP/M             a0  IBM Thinkpad hi ea
Rufus alignment
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a5  FreeBSD          eb
BeOS fs
 f  W95 Ext'd (LBA) 54  OnTrackDM6       a6  OpenBSD          ee
GPT
10  OPUS            55  EZ-Drive        a7  NeXTSTEP         ef
EFI (FAT-12/16/
11  Hidden FAT12    56  Golden Bow      a8  Darwin UFS       f0
Linux/PA-RISC b
12  Compaq diagnost 5c  Priam Edisk     a9  NetBSD           f1
SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       ab  Darwin boot      f4
SpeedStor
16  Hidden FAT16    63  GNU HURD or Sys af  HFS / HFS+       f2
DOS secondary
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs          fb
VMware VMFS
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap        fc
VMware VMKCORE
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid fd
Linux raid auto
1c  Hidden W95 FAT3 75  PC/IX           bc  Acronis FAT32 L fe
LANstep
1e  Hidden W95 FAT1 80  Old Minix       be  Solaris boot    ff
BBT
Hex code (type L to list all codes): 82
Changed type of partition 'Linux' to 'Linux swap / Solaris'.
Command (m for help): p
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x806fa0ce
Device     Boot Start      End  Sectors Size Id Type
/dev/sda1           2048 10487807 10485760    5G 82 Linux swap /
Solaris
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
[76896.597480]  sda: sda1
Syncing disks.
```

При изменении типа таблицы разделов все существующие разделы удаляются.

Для создания GPT разделов нужно явно сменить тип разбиения командой g.

Пример: изменение таблицы разделов на GPT и создание нового GPT раздела.

```
# fdisk /dev/sda
Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you decide to write
them.
Be careful before using the write command.
[77210.824171]  sda: sda1
Command (m for help): p
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x806fa0ce
Device     Boot Start      End  Sectors Size Id Type
/dev/sda1           2048 10487807 10485760    5G 82 Linux swap /
Solaris
Command (m for help): g
Created a new GPT disklabel (GUID: 419B9A4F-B2BD-9D47-A19F-
7466E0249CC5).
The old dos signature will be removed by a write command.
Command (m for help): p
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 419B9A4F-B2BD-9D47-A19F-7466E0249CC5
Command (m for help): n
Partition number (1-128, default 1): 1
First sector (2048-41943006, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-41943006, default
41943006): +5G
Created a new partition 1 of type 'Linux filesystem' and of size 5
GiB.
Command (m for help): p
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disklabel type: gpt
Disk identifier: 419B9A4F-B2BD-9D47-A19F-7466E0249CC5
Device     Start      End  Sectors Size Type
/dev/sda1   2048 10487807 10485760   5G Linux filesystem
Command (m for help): t
Selected partition 1
Partition type (type L to list all types): L
   1 EFI System                             C12A7328-F81F-11D2-BA4B-
00A0C93EC93B
   2 MBR partition scheme                   024DEE41-33E7-11D3-9D69-
0008C781F39F
   3 Intel Fast Flash                       D3BFE2DE-3DAF-11DF-BA40-
E3A556D89593
   4 BIOS boot                              21686148-6449-6E6F-744E-
656564454649
   5 Sony boot partition                    F4019732-066E-4E12-8273-
346C5641494F
   6 Lenovo boot partition                  BFBFAFE7-A34F-448A-9A5B-
6213EB736C22
   7 PowerPC PReP boot                      9E1A2D38-C612-4316-AA26-
8B49521E5A8B
   8 ONIE boot                              7412F7D5-A156-4B13-81DC-
867174929325
   9 ONIE config                            D4E6E2CD-4469-46F3-B5CB-
1BFF57AFC149
  10 Microsoft reserved                     E3C9E316-0B5C-4DB8-817D-
F92DF00215AE
  11 Microsoft basic data                   EBD0A0A2-B9E5-4433-87C0-
68B6B72699C7
  12 Microsoft LDM metadata                 5808C8AA-7E8F-42E0-85D2-
E1E90434CFB3
  13 Microsoft LDM data                     AF9B60A0-1431-4F62-BC68-
3311714A69AD
  14 Windows recovery environment           DE94BBA4-06D1-4D40-A16A-
BFD50179D6AC
  15 IBM General Parallel Fs                37AFFC90-EF7D-4E96-91C3-
2D7AE055B174
  16 Microsoft Storage Spaces               E75CAF8F-F680-4CEE-AFA3-
B001E56EFC2D
  17 HP-UX data                             75894C1E-3AEB-11D3-B7C1-
7B03A0000000
  18 HP-UX service                          E2A1E728-32E3-11D6-A682-
7B03A0000000
  19 Linux swap                             0657FD6D-A4AB-43C4-84E5-
0933C84B4F4F
  20 Linux filesystem                       0FC63DAF-8483-4772-8E79-
3D69D8477DE4
  21 Linux server data                      3B8F8425-20E0-4F3B-907F-
1A25A76F98E8
```

```
 22 Linux root (x86)                    44479540-F297-41B2-9AF7-
D131D5F0458A
 23 Linux root (ARM)                    69DAD710-2CE4-4E3C-B16C-
21A1D49ABED3
 24 Linux root (x86-64)                 4F68BCE3-E8CD-4DB1-96E7-
FBCAF984B709
 25 Linux root (ARM-64)                 B921B045-1DF0-41C3-AF44-
4C6F280D3FAE
 26 Linux root  (IA-64)                 993D8D3D-F80E-4225-855A-
9DAF8ED7EA97
 27 Linux reserved                      8DA63339-0007-60C0-C436-
083AC8230908
 28 Linux home                          933AC7E1-2EB4-4F13-B844-
0E14E2AEF915
 29 Linux RAID                          A19D880F-05FC-4D3B-A006-
743F0F84911E
 30 Linux extended boot                 BC13C2FF-59E6-4262-A352-
B275FD6F7172
 31 Linux LVM                           E6D6D379-F507-44C2-A23C-
238F2A3DF928
 32 FreeBSD data                        516E7CB4-6ECF-11D6-8FF8-
00022D09712B
 33 FreeBSD boot                        83BD6B9D-7F41-11DC-BE0B-
001560B84F0F
 34 FreeBSD swap                        516E7CB5-6ECF-11D6-8FF8-
00022D09712B
 35 FreeBSD UFS                         516E7CB6-6ECF-11D6-8FF8-
00022D09712B
 36 FreeBSD ZFS                         516E7CBA-6ECF-11D6-8FF8-
00022D09712B
 37 FreeBSD Vinum                       516E7CB8-6ECF-11D6-8FF8-
00022D09712B
 38 Apple HFS/HFS+                      48465300-0000-11AA-AA11-
00306543ECAC
 39 Apple UFS                           55465300-0000-11AA-AA11-
00306543ECAC
 40 Apple RAID                          52414944-0000-11AA-AA11-
00306543ECAC
 41 Apple RAID offline                  52414944-5F4F-11AA-AA11-
00306543ECAC
 42 Apple boot                          426F6F74-0000-11AA-AA11-
00306543ECAC
 43 Apple label                         4C616265-6C00-11AA-AA11-
00306543ECAC
 44 Apple TV recovery                   5265636F-7665-11AA-AA11-
00306543ECAC
 45 Apple Core storage                  53746F72-6167-11AA-AA11-
00306543ECAC
 46 Solaris boot                        6A82CB45-1DD2-11B2-99A6-
```

```
080020736631
 47 Solaris root                         6A85CF4D-1DD2-11B2-99A6-
080020736631
 48 Solaris /usr & Apple ZFS             6A898CC3-1DD2-11B2-99A6-
080020736631
 49 Solaris swap                         6A87C46F-1DD2-11B2-99A6-
080020736631
 50 Solaris backup                       6A8B642B-1DD2-11B2-99A6-
080020736631
 51 Solaris /var                         6A8EF2E9-1DD2-11B2-99A6-
080020736631
 52 Solaris /home                        6A90BA39-1DD2-11B2-99A6-
080020736631
 53 Solaris alternate sector             6A9283A5-1DD2-11B2-99A6-
080020736631
 54 Solaris reserved 1                   6A945A3B-1DD2-11B2-99A6-
080020736631
 55 Solaris reserved 2                   6A9630D1-1DD2-11B2-99A6-
080020736631
 56 Solaris reserved 3                   6A980767-1DD2-11B2-99A6-
080020736631
 57 Solaris reserved 4                   6A96237F-1DD2-11B2-99A6-
080020736631
 58 Solaris reserved 5                   6A8D2AC7-1DD2-11B2-99A6-
080020736631
 59 NetBSD swap                          49F48D32-B10E-11DC-B99B-
0019D1879648
 60 NetBSD FFS                           49F48D5A-B10E-11DC-B99B-
0019D1879648
 61 NetBSD LFS                           49F48D82-B10E-11DC-B99B-
0019D1879648
 62 NetBSD concatenated                  2DB519C4-B10E-11DC-B99B-
0019D1879648
 63 NetBSD encrypted                     2DB519EC-B10E-11DC-B99B-
0019D1879648
 64 NetBSD RAID                          49F48DAA-B10E-11DC-B99B-
0019D1879648
 65 ChromeOS kernel                      FE3A2A5D-4F32-41A7-B725-
ACCC3285A309
 66 ChromeOS root fs                     3CB8E202-3B7E-47DD-8A3C-
7FF2A13CFCEC
 67 ChromeOS reserved                    2E0A753D-9E48-43B0-8337-
B15192CB1B5E
 68 MidnightBSD data                     85D5E45A-237C-11E1-B4B3-
E89A8F7FC3A7
 69 MidnightBSD boot                     85D5E45E-237C-11E1-B4B3-
E89A8F7FC3A7
 70 MidnightBSD swap                     85D5E45B-237C-11E1-B4B3-
E89A8F7FC3A7
```

```
 71 MidnightBSD UFS                    0394EF8B-237E-11E1-B4B3-
E89A8F7FC3A7
 72 MidnightBSD ZFS                    85D5E45D-237C-11E1-B4B3-
E89A8F7FC3A7
 73 MidnightBSD Vinum                  85D5E45C-237C-11E1-B4B3-
E89A8F7FC3A7
 74 Ceph Journal                       45B0969E-9B03-4F30-B4C6-
B4B80CEFF106
 75 Ceph Encrypted Journal             45B0969E-9B03-4F30-B4C6-
5EC00CEFF106
 76 Ceph OSD                           4FBD7E29-9D25-41B8-AFD0-
062C0CEFF05D
 77 Ceph crypt OSD                     4FBD7E29-9D25-41B8-AFD0-
5EC00CEFF05D
 78 Ceph disk in creation              89C57F98-2FE5-4DC0-89C1-
F3AD0CEFF2BE
 79 Ceph crypt disk in creation        89C57F98-2FE5-4DC0-89C1-
5EC00CEFF2BE
 80 VMware VMFS                        AA31E02A-400F-11DB-9590-
000C2911D1B8
 81 VMware Diagnostic                  9D275380-40AD-11DB-BF97-
000C2911D1B8
 82 VMware Virtual SAN                 381CFCCC-7288-11E0-92EE-
000C2911D0B2
 83 VMware Virsto                      77719A0C-A4A0-11E3-A47E-
000C29745A24
 84 VMware Reserved                    9198EFFC-31C0-11DB-8F78-
000C2911D1B8
 85 OpenBSD data                       824CC7A0-36A8-11E3-890A-
952519AD3F61
 86 QNX6 file system                   CEF5A9AD-73BC-4601-89F3-
CDEEEEE321A1
 87 Plan 9 partition                   C91818F9-8025-47AF-89D2-
F030D7000C2C
Partition type (type L to list all types): 20
Changed type of partition 'Linux filesystem' to 'Linux
filesystem'.
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
[77277.892973]  sda: sda1
Syncing disks.
```

3.4. Создание файловой системы.

Для обеспечения возможности хранения информации в виде файлов в разделе должна быть создана файловая система, то есть должно быть произведено форматирование раздела.

При форматировании формируются суперблок, массив индексных дескрипторов и выделяется пространство для блоков данных.

Изначально в GNU/Linux основной файловой системой являлась `EXT2`. В настоящий момент помимо `EXT2` широко используются следующие файловые системы:

`EXT3` - осовремененная версия `EXT2` с поддержкой журналирования и с улучшенными показателями времени восстановления после сбоя, продвигаемая Red Hat;

EXT4 - следующая ступень развития файловых систем EXT;

`XFS` - высокопроизводительная файловая система для больших и очень больших объемов хранимой информации, разработанная в Silicon Graphics, распространяемая как Open Source (начиная с ядра 2.4.24 ее поддержка имеется в ядре Linux без необходимости установки специального патча);

BTRFS - современная файловая система, обеспечивающая такие возможности как отказоустойчивость на уровне ФС, снимки, оптимизация для дисков SSD, динамическое выделение inode.

`JFS` - высокопроизводительная файловая система от IBM, используемая при необходимости хранения очень больших объемов информации.

Чтобы подключить и использовать определенную файловую систему нужна поддержка этой ФС в ядре.

Для создания ФС поддержка со стороны ядра не обязательна, но необходимо иметь соответствующие утилиты для обслуживания этой ФС.

Примечание: если у вас есть подходящие утилиты вы можете создать на диске какую угодно ФС, но не факт, что вы сможете ее использовать (монтировать).

Для создания файловой системы используется команда `mkfs`, при вызове которой в качестве аргумента должен быть задан файл устройства, соответствующий разделу жесткого диска (могут быть использованы файлы устройств для дискет и прочих блочных устройств).

Команда `mkfs` по умолчанию создает файловую систему `EXT2`.

Пример: для создания файловой системы EXT2 на втором первичном разделе диска следует выполнить такую команду (форматирование дисков - привилегия администратора):

```
# mkfs /dev/sda2
```

При успешном выполнении команды `mkfs` на экран будет выведена

информация о размере блоков и их количестве в данной файловой системе, о местоположении копий суперблока и прочее.

Для создания иной файловой системы ее тип необходимо указать после опции `-t` команды `mkfs`.

Пример: для создания на том же разделе диска файловой системы EXT4 надо выполнить такую команду:

```
# mkfs -t ext4 /dev/sda2
```

Вместо опции `-t` для форматирования файловых систем можно использовать специализированные утилиты:

`mke2fs` - для создания `EXT2, EXT3` и EXT4 файловых систем;

`mkntfs` - для создания `NTFS`;

`mkdosfs` - для форматирования под `FAT`.

Альтернативный способ создания разных файловых систем – использование команд `mkfs.*` :

```
/sbin/mkfs.ext2
/sbin/mkfs.ext3
/sbin/mkfs.ext4
/sbin/mkfs.xfs
```

Примечание: Эти команды, в свою очередь, являются символьными ссылками, либо же жесткими связями со специализированными утилитами, упомянутыми выше.

Пример:

```
# ls -F /sbin/mkfs.*
/sbin/mkfs.btrfs*       /sbin/mkfs.ext4*           /sbin/mkfs.minix*
/sbin/mkfs.vfat@
/sbin/mkfs.cramfs*     /sbin/mkfs.fat*            /sbin/mkfs.msdos@
/sbin/mkfs.xfs*
/sbin/mkfs.exfat@    /sbin/mkfs.gfs2*      /sbin/mkfs.ntfs@
/sbin/mkfs.ext2*     /sbin/mkfs.hfsplus*   /sbin/mkfs.reiserfs@
/sbin/mkfs.ext3*     /sbin/mkfs.jffs2*     /sbin/mkfs.ubifs*
```

Команда `mkfs` предоставляет возможность использовать опцию `-c`, которая заставляет команду перед созданием файловой системы проверять поверхность диска на наличие плохих блоков.

3.5. Проверка целостности файловой системы.

Целостность структуры и данных файловой системы может быть

нарушена в результате сбоя системы.

Примечание: например, ставшего следствием сбоя питания или же аппаратной неисправности.

Если работа операционной системы была прервана и файловая система не была размонтирована, то в суперблоке файловой системы остается специальный флаг (`dirty` – грязный или `not clean`), который сообщает о том, что в файловой системе возможны нарушения.

Если файловая система была размонтирована правильно и сбоев не было, то говорят, что файловая система находится в состоянии `clean` - “чистая”.

Сбой файловой системы обычно сопровождается тем, что информация, находящаяся в кэше (дисковых буферах), не синхронизируется с информацией в файловой системе.

Сбои системы могут привести к следующим проблемам в файловой системе:

Искажению или потере информации в блоках данных

Появлению в системе блоков данных, которые считаются занятыми, хотя на них не указывает ни один индексный дескриптор.

Наличию перекрестных ссылок на блоки данных.

Появлению метаданных с ненулевым счетчиком ссылок, на которые не ссылаются никакие файлы.

Наличию противоречивых записей в каталогах и т.п.

Различают два класса возможных нарушений в файловой системе:

Нарушение целостности данных.

Нарушение целостности структуры файловой системы.

Для защиты целостности данных могут быть использованы разнообразные методы резервного хранения данных (backup).

Примечание: Утилиты восстановления целостности файловой системы не могут гарантировать восстановление данных после сбоя. Гарантом сохранности данных является только наличие правильно выполненных резервных копий данных.

Для восстановления структуры файловых систем в GNU/Linux используется утилита `fsck`

Для проверки различных типов файловых систем используются специализированные утилиты `fsck.*` является или опция -t

Внимание! Проверка целостности файловой системы должна осуществляться лишь на размонтированной файловой системе. Невыполнение этого требования может привести

к полной потере данных на файловой системе!

Пример:

```
# ls -F /sbin/fsck.*
/sbin/fsck.btrfs*            /sbin/fsck.ext3*        /sbin/fsck.hfs@
/sbin/fsck.ntfs@
/sbin/fsck.cramfs*       /sbin/fsck.ext4*       /sbin/fsck.hfsplus*
/sbin/fsck.reiserfs@
/sbin/fsck.exfat@         /sbin/fsck.fat*         /sbin/fsck.minix*
/sbin/fsck.vfat@
/sbin/fsck.ext2*            /sbin/fsck.gfs2*      /sbin/fsck.msdos@
/sbin/fsck.xfs*
```

Если вызвать утилиту `fsck` без опции `-t`, указывающей тип файловой системы, то будет вызвана утилита `e2fsck`, предназначенная для проверки файловой системы `EXT`.

Пример:

```
# fsck.ext3 /dev/sda1
e2fsck 1.44.6 (5-Mar-2019)
/dev/sda1: clean, 11/327680 files, 39535/1310720 blocks
```

Примечание: Эта команда проверит целостность файловой системы `ext3` *на первом первичном разделе первого SCSI диска в системе.*

Полная проверка осуществляется не всегда, а только при условии:

Наличия флага `dirty` в суперблоке, что бывает при сбое или выключении питания без размонтирования файловой системы.

Достижения максимального разрешенного количества монтирований файловой системы без проверки ее целостности.

Достижения максимального срока без проверки целостности файловой системы.

Если необходимо выполнить полную проверку файловой системы в отсутствии любого из приведенных выше условий, то надо использовать опцию `-f` команды `e2fsck`.

Для проверки поверхности диска перед проверкой файловой системы требуется использовать опцию `-c` команды `e2fsck`.

Пример: приведенная ниже команда выполнит полную проверку файловой системы, так как превышено максимально разрешенное для данной файловой системы число монтирований без проверки целостности файловой системы.

```
# e2fsck /dev/sda1
e2fsck 1.32 (09-Nov-2002)
USBDISK has been mounted 27 times without being checked, check
forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
USBDISK:     3585/2442240     files     (2.2%     non-contiguous),
3004291/19534832 blocks
```

Команда `e2fsck` используется специальный каталог `lost+found`, находящийся в корневом (верхнем) каталоге файловой системы устройства для сохранения потерянных цепочек блоков данных (файлов, у которых нет имени).

Для проверки целостности файловой системы `XFS` применяется команда `xfs_repair`.

Пример:

```
# xfs_repair /dev/sda2
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan and clear agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
```

```
Phase 5 - rebuild AG headers and trees...
        - reset superblock...
Phase 6 - check inode connectivity...
        - resetting contents of realtime bitmap and summary inodes
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify and correct link counts...
done
```

3.6. Монтирование файловых систем.

В GNU/Linux все файловые системы, доступные для работы с ними, должны быть “подцеплены” к логической структуре файлов и каталогов.

Процесс “подцепления” файловой системы, существующей на дисковом или ином блочном устройстве, в общее дерево файлов и каталогов называется монтированием.

Каталог, в который произошло “подцепление” отформатированного устройства, называется точкой монтирования.

За исключением файловых систем, для которых установлены специальные настройки в, например, файле `/etc/fstab`, монтирование файловых систем производится суперпользователем.

Стандарт FHS предписывает, что точки монтирования временных файловых систем должны находиться в каталоге `/mnt`.

Временные файловые системы для сменных носителей должны в соответствии с FHS находится в каталоге /media

Примечание: каталог `/media/cdrom` *может быть точкой монтирования для CD-ROM, а* `/media/usbflash` *– для флоппи диска.*

Команда `mount` монтирует файловую систему указанного с помощью опции `-t` типа (по умолчанию `ext2`) в каталог - точку монтирования.

Первый аргумент команды `mount` - файл блочного устройства, на котором находится монтируемая файловая система.

Второй аргумент - точка монтирования этой файловой системы.

Пример: для монтирования USB диска с файловой системой `EXT2` следует выполнить команду:

```
# mount /dev/sda1 /mnt/usbflash
```

Если команда `mount` вызвана без аргументов, то она показывает список

смонтированных файловых систем, то есть имена файлов устройств и соответствующих им точек монтирования.

Пример:

```
# mount | grep ^/dev
/dev/vda2 on / type ext4 (rw,relatime,seclabel)
/dev/vda1 on /boot type ext4 (rw,relatime,seclabel)
/dev/sda1 on /media/usbflash type ext3 (rw,relatime,seclabel)
```

Последние версии mount умеют сами определять тип подключаемой ФС, если нет, то при монтировании, например, CD ROM необходимо указать тип файловой системы `iso9660` :

Пример:

```
# mount -t iso9660 /dev/cdrom /media/cdrom
```

Монтирование файловой системы подменяет индексный дескриптор каталога - точки монтирования. До монтирования индексный дескриптор соответствует каталогу, находящемуся в файловой системе, к которой монтируется устройство. После монтирования - индексный дескриптор каталога - точки монтирования принадлежит уже смонтированной файловой системе.

Пример:

```
# ls -ldi /media/usbflash/
529793 drwxr-xr-x. 2 root root 4096 Feb 10 12:13 /media/usbflash/
# mount /dev/sda1 /media/usbflash
# ls -ldi /media/usbflash/
2 drwxr-xr-x. 3 root root 4096 Feb 10 08:47 /media/usbflash/
```

Примечание: Из этого примера заметно, что до монтирования временной файловой системы в каталог `/media/usbdisk`*, индексный дескриптор каталога был* `529793` *, а после монтирования* `2`*. У всех корневых каталогов файловых систем на диске индексный дескриптор имеет номер* `2`*. У виртуальных файловых систем корневой каталог имеет номер inode 1.*

Пример:

```
# ls -ldi /boot /
2 dr-xr-xr-x. 18 root root 4096 Nov  9  2019 /
2 dr-xr-xr-x.  6 root root 4096 Feb  9 10:39 /boot
```

```
# mount | grep ^/dev
/dev/vda2 on / type ext4 (rw,relatime,seclabel)
/dev/vda1 on /boot type ext4 (rw,relatime,seclabel)
/dev/sda1 on /media/usbflash type ext3 (rw,relatime,seclabel)
# ls -ldi /proc
1 dr-xr-xr-x. 91 root root 0 Feb  9 10:50 /proc
```

Примечание: В этом примере демонстрируется то, что для каждой файловой системы inode корневого каталога, то есть точки монтирования - `2`.

Как и операцию монтирования, размонтировать файловые системы (без специальных настроек в `/etc/fstab`) имеет право только суперпользователь.

Для размонтирования файловой системы применяется команда `umount`, которой в качестве аргумента должен быть задан единственный аргумент - либо точка монтирования, либо файл устройства.

Пример:

```
# ls /media/usbflash/
lost+found  somedocs.txt
[root@lin00 ~]# umount /media/usbflash/
[root@lin00 ~]# ls /media/usbflash/
```

Примечание: В этом примере показана работа команды `umount`. *После ее выполнения в каталоге - точке монтирования больше нет доступа к файлам на временной файловой системе.*

Хорошим правилом является следующее: не следует хранить какие-либо файлы в каталогах, являющихся точками монтирования временных файловых систем.

3.7. Файл информации о файловых системах /etc/fstab.

Конфигурационный файл `/etc/fstab (filesystem table)` содержит информацию о файловых системах, которые нужно смонтировать при загрузке или по требованию пользователя.

Информация в файле `/etc/fstab` представлена в виде таблицы:

Пример:

```
# cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Sat Nov  9 04:18:20 2019
#
```

```
# Accessible filesystems, by reference, are maintained under
'/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for
more info.
#
# After editing this file, run 'systemctl daemon-reload' to update
systemd
# units generated from this file.
#
UUID=1a5655e7-613b-4733-ba6b-e598633402fb                              /
ext4    defaults        1 1
UUID=897234d2-2307-406f-990d-a9620bcb4d1f                            /boot
ext4    defaults        1 2
UUID=5a4f53ca-c983-4def-8535-e2541d8419bb                             swap
swap    defaults        0 0
```

Строки, начинающиеся с символа # являются комментариями.

Таблица, содержащаяся в файле `/etc/fstab` , состоит из колонок полей, назначение которых следующее:

Первое поле (fs_spec) содержит указатель на монтируемое устройство. Здесь может быть имя файла устройства, UUID файловой системы, UUD GPT, метка GPT или метка тома в файловой системе.

Второе (fs_file) указывает каталог - точку монтирования

Третье (fs_vfstype) содержит тип файловой системы на данном носителе.

Четвертое (fs_mntops) содержит опции команды `mount` , которые должны быть использованы при монтировании данной файловой системы. Для разделов swap это поле должно содержать `sw`.

Пятое (fs_freq) указывает надо ли для данной файловой системы производить автоматическое резервное копирование (backup) командой `dump` . Если в этом поле находится `1`, то резервное копирование производится, если `0` – нет.

Шестое (fs_passno) предназначено для порядка проверки целостности файловых систем при загрузке операционной системы. Для корневой файловой системы в этом поле должно быть установлено значение `1`. Для других файловых систем, которые необходимо проверять при загрузке, следует указать `2`. Если проверка не требуется, то в этом поле ставится `0` .

Ниже приведена таблица, содержащая часто используемые опции команды mount, указываемые в поле fs_mntops файла `/etc/fstab` .

Опция	Назначение
defaults	Установки: rw,suid,dev,exec,auto,nouser,asynch.
asynch	Асинхронный режим ввода/вывода.
auto	Монтировать во время загрузки.
noauto	Не монтировать во время загрузки.
exec	Разрешение выполнения файлов с машинным кодом.
noexec	Запрет выполнения файлов с машинным кодом.
suid	Бит SUID устанавливать можно.
nosuid	Запрещена установка битов SUID.
user	Обычный пользователь может монтировать устройство.
nouser	Монтировать разрешено только суперпользователю.
ro	Режим только для чтения.
rw	Разрешено как чтение, так и запись.

Наличие записи в файле /etc/fstab означает, что данная файловая система может быть смонтирована без указания обоих аргументов командной строки mount – файла устройства и каталога – точки монтирования. Для монтирования файловой системы, указанной в /etc/fstab , достаточно указать либо точку монтирования, либо файл устройства.

В ОС построенных на systemd для монтирования разделов на основе /etc/fstab генерируются специальные юниты типа mount.

```
# systemctl list-units -t mount
UNIT                    LOAD   ACTIVE SUB     DESCRIPTION
-.mount                 loaded active mounted Root Mount
boot.mount              loaded active mounted /boot
dev-hugepages.mount        loaded active mounted Huge Pages File
System
dev-mqueue.mount          loaded active mounted POSIX Message Queue
File System
run-user-0.mount        loaded active mounted /run/user/0
sys-kernel-config.mount loaded active mounted Kernel Configuration
File System
sys-kernel-debug.mount   loaded active mounted Kernel Debug File
System
LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization
of SUB.
SUB     = The low-level unit activation state, values depend on
unit type.
7 loaded units listed. Pass --all to see loaded but inactive
units, too.
To show all installed unit files use 'systemctl list-unit-files'.
# systemctl status [-].mount
● -.mount - Root Mount
```

```
   Loaded: loaded (/etc/fstab; generated)
    Active: active (mounted) since Tue 2021-02-09 10:50:18 +05; 1
day 6h ago
    Where: /
     What: /dev/vda2
     Docs: man:fstab(5)
           man:systemd-fstab-generator(8)
```

Юнит для монтирования можно создать самостоятельно или написать специальные опции монтирования в файл /etc/fstab , которые сформируют юнит монтирования с нужными вам параметрами. (см. man 5 systemd.mount).

Использование /etc/fstab считается предпочтительным.

Пример: создадим юнит, который описывает новую точку монтирования:

```
# mkfs.ext4 /dev/sda1
mke2fs 1.44.6 (5-Mar-2019)
/dev/sda1 contains a ext3 file system
    last mounted on /media/usbflash on Wed Feb 10 12:16:19 2021
Proceed anyway? (y,N) y
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: a13a401f-cdf5-48e3-a277-21409ad76de0
Superblock backups stored on blocks:
...
# mkdir /storage1
# vi /etc/systemd/system/storage1.mount
# cat /etc/systemd/system/storage1.mount
[Unit]
Description=Persistent Mount Point for directory /storage1
[Mount]
What=/dev/disk/by-uuid/a13a401f-cdf5-48e3-a277-21409ad76de0
Where=/storage1
Type=ext4
Options=defaults
[Install]
WantedBy=sysinit.target
# systemctl start storage1.mount
# systemctl status storage1.mount
● storage1.mount - Persistent Mount Point for directory /storage1
     Loaded: loaded (/etc/systemd/system/storage1.mount; disabled;
vendor preset:>
    Active: active (mounted) since Wed 2021-02-10 17:49:04 +05; 6s
ago
     Where: /storage1
      What: /dev/sda1
     Tasks: 0 (limit: 12472)
    Memory: 76.0K
```

```
   CGroup: /system.slice/storage1.mount
Feb 10 17:49:04 lin00 systemd[1]: Mounting Persistent Mount Point
for directory>
Feb 10 17:49:04 lin00 systemd[1]: Mounted Persistent Mount Point
for directory
# systemctl enable storage1.mount
Created                                                  symlink
/etc/systemd/system/sysinit.target.wants/storage1.mount          →
/etc/systemd/system/storage1.mount.
# systemctl status storage1.mount
● storage1.mount - Persistent Mount Point for directory /storage1
     Loaded: loaded (/etc/systemd/system/storage1.mount; enabled;
vendor preset: >
    Active: active (mounted) since Wed 2021-02-10 17:49:04 +05;
2min 17s ago
...
```

Рассмотрим пример подключения сетевых директорий с помощью fstab.

1. Создайте файл с авторизационными данными:

```
# nano /root/.smbuser
username=имя_пользователя
password=пароль_пользователя
domain=ДОМЕН
```

Для безопасности ограничьте доступ к файлу с указанными данными:

```
# chmod 400 /root/.smbuser
```

2. Для подключения сетевой директории впишите в конец файла /etc/fstab её параметры.

Откройте его в текстовом редакторе:

```
# nano /etc/fstab
```

Синтаксис строки подключения для windows или samba-ресурсов следующий.

```
//<адрес_сервера>/<имя_сетевого_каталога>
<каталог_для_монтирования>                                    cifs
credentials=<путь_до_файла_с_авторизационными_данными>,iocharset=u
tf8,file_mode=0777,dir_mode=0777 0 0
```

При подключении используйте свои параметры.

В случае возникновения ошибки при монтировании общего ресурса, который находится на ПК с ОС Windows, нужно добавить в строку подключения следующий параметр:

```
vers=1.0
```

Пример:

```
//windows_pc/common                  /mnt/common/                  cifs
credentials=/root/.smbuser,vers=1.0,iocharset=utf8,file_mode=0777,
dir_mode=0777 0 0
```

3. Создайте каталог, куда будет монтироваться папка и назначьте ей права.

```
# mkdir /mnt/common
# chmod 777 /mnt/common
```

Для удобства, можете создать ярлык каталога на рабочем столе, для этого выполните нижеследующую команду, вписав имя своего пользователя.

```
ln -s /mnt/common/ /home/<имя_пользователя>/Рабочий\ стол/Общая
```

4. Выполните монтирование каталога

```
mount -a
```

Примечание: В некоторых случаях, при загрузке компьютера монтирование происходит быстрее получения IP-адреса, тем самым, монтирование не происходит. Связано это с особенностями конфигурации сети. Для предотвращения подобных случаев можно вписать команду монтирования в файл /etc/gdm/PreSession/Default. Тогда монтирование будет происходить при входе пользователей в систему.

Откройте файл /etc/gdm/PreSession/Default и впишите в конец строку монтирования, после неё переведите курсор на новую строку и сохраните.

```
# nano /etc/gdm/PreSession/Default
…

mount -a &
```

Примечание: При подключении сетевой папки на windows server 2003 в опции монтирования надо дописать параметр vers=1.0, а так же логин и пароль не должны содержать кириллических символов.

3.8. Мониторинг дисковых ресурсов.

Команда lsblk выдает информацию о дисках, разделах и точках монтирования.

Пример:

```
# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   20G  0 disk
├─sda1   8:1    0    5G  0 part /storage1
└─sda2   8:2    0    2G  0 part /storage2
sr0     11:0    1 1024M  0 rom
vda    253:0    0   20G  0 disk
├─vda1 253:1    0    1G  0 part /boot
├─vda2 253:2    0   17G  0 part /
└─vda3 253:3    0    2G  0 part [SWAP]
# lsblk -f
NAME                            FSTYPE          LABEL          UUID
MOUNTPOINT
sda
├─sda1 ext4                     a13a401f-cdf5-48e3-a277-21409ad76de0 /storage1
└─sda2 xfs                      e5b66aa7-c101-4bcd-bb63-2d10d2e4aa44 /storage2
sr0
vda
├─vda1 ext4   BOOT  897234d2-2307-406f-990d-a9620bcb4d1f /boot
├─vda2 ext4   ROOT  1a5655e7-613b-4733-ba6b-e598633402fb /
└─vda3 swap   SWAP1 5a4f53ca-c983-4def-8535-e2541d8419bb [SWAP]
```

Команда findmnt показывает точки монтирования в удобном для восприятия виде.

Пример:

```
# findmnt
TARGET                                  SOURCE     FSTYPE  OPTIONS
/                                                    /dev/vda2   ext4
rw,relatime,seclabel
├─/sys                                              sysfs        sysfs
rw,nosuid,nodev,noexec,
│ ├─/sys/kernel/security                            securityfs  securit
rw,nosuid,nodev,noexec,
│ ├─/sys/fs/cgroup                                  tmpfs        tmpfs
ro,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/systemd                       cgroup       cgroup
rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/perf_event                    cgroup       cgroup
rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/pids                          cgroup       cgroup
rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/freezer                       cgroup       cgroup
rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/cpuset                        cgroup       cgroup
```

```
rw,nosuid,nodev,noexec,
│   │   ├─/sys/fs/cgroup/net_cls,net_prio  cgroup            cgroup
rw,nosuid,nodev,noexec,
│   │   ├─/sys/fs/cgroup/cpu,cpuacct          cgroup          cgroup
rw,nosuid,nodev,noexec,
│   │   ├─/sys/fs/cgroup/rdma                  cgroup         cgroup
rw,nosuid,nodev,noexec,
│   │   ├─/sys/fs/cgroup/devices               cgroup         cgroup
rw,nosuid,nodev,noexec,
│   │   ├─/sys/fs/cgroup/memory                cgroup         cgroup
rw,nosuid,nodev,noexec,
│   │   ├─/sys/fs/cgroup/blkio                 cgroup         cgroup
rw,nosuid,nodev,noexec,
│   │   └─/sys/fs/cgroup/hugetlb               cgroup         cgroup
rw,nosuid,nodev,noexec,
│   ├─/sys/fs/pstore                           pstore         pstore
rw,nosuid,nodev,noexec,
│   ├─/sys/fs/bpf                                bpf            bpf
rw,nosuid,nodev,noexec,
│   ├─/sys/fs/selinux                          selinuxfs    selinux
rw,relatime
│   ├─/sys/kernel/debug                        debugfs      debugfs
rw,relatime,seclabel
│   └─/sys/kernel/config                       configfs     configf
rw,relatime
├─/proc                                         proc           proc
rw,nosuid,nodev,noexec,
│   └─/proc/sys/fs/binfmt_misc                 systemd-1    autofs
rw,relatime,fd=35,pgrp=
├─/dev                                         devtmpfs     devtmpf
rw,nosuid,seclabel,size
│   ├─/dev/shm                                 tmpfs          tmpfs
rw,nosuid,nodev,seclabe
│   ├─/dev/pts                                 devpts         devpts
rw,nosuid,noexec,relati
│   ├─/dev/mqueue                              mqueue         mqueue
rw,relatime,seclabel
│   └─/dev/hugepages                           hugetlbfs    hugetlb
rw,relatime,seclabel,pa
├─/run                                          tmpfs          tmpfs
rw,nosuid,nodev,seclabe
│   └─/run/user/0                              tmpfs          tmpfs
rw,nosuid,nodev,relatim
├─/boot                                           /dev/vda1    ext4
rw,relatime,seclabel
├─/storage1                                       /dev/sda1    ext4
rw,relatime,seclabel
└─/storage2                                        /dev/sda2    xfs
rw,relatime,seclabel,at
```

Команда blkid находит и печатает атрибуты блочных устройств.

Пример:

```
# blkid
/dev/vda1:        LABEL="BOOT"        UUID="897234d2-2307-406f-990d-
a9620bcb4d1f" TYPE="ext4" PARTUUID="706c87eb-01"
/dev/vda2:        LABEL="ROOT"        UUID="1a5655e7-613b-4733-ba6b-
e598633402fb" TYPE="ext4" PARTUUID="706c87eb-02"
/dev/vda3:       LABEL="SWAP1"        UUID="5a4f53ca-c983-4def-8535-
e2541d8419bb" TYPE="swap" PARTUUID="706c87eb-03"
/dev/sda1: UUID="a13a401f-cdf5-48e3-a277-21409ad76de0" TYPE="ext4"
PARTUUID="87ce5033-6121-0b42-b875-05ff23508aad"
/dev/sda2:  UUID="e5b66aa7-c101-4bcd-bb63-2d10d2e4aa44"  TYPE="xfs"
PARTUUID="fea45c49-5143-a34b-8b92-9e78bc409109"
# blkid -i /dev/sda1
/dev/sda1:     MINIMUM_IO_SIZE="512"     PHYSICAL_SECTOR_SIZE="512"
LOGICAL_SECTOR_SIZE="512"
# blkid -t TYPE="ext4"
/dev/vda1:        LABEL="BOOT"        UUID="897234d2-2307-406f-990d-
a9620bcb4d1f" TYPE="ext4" PARTUUID="706c87eb-01"
/dev/vda2:        LABEL="ROOT"        UUID="1a5655e7-613b-4733-ba6b-
e598633402fb" TYPE="ext4" PARTUUID="706c87eb-02"
/dev/sda1: UUID="a13a401f-cdf5-48e3-a277-21409ad76de0" TYPE="ext4"
PARTUUID="87ce5033-6121-0b42-b875-05ff23508aad"
```

Команда `df` устройство выводит количество свободного места в блоках на специфицированном устройстве, а если оно не указано, то на всех смонтированных файловых системах.

Удобно использовать опцию `-h`, для отображения информации в понятных для пользователя единицах (human readable format):

Пример:

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        975M     0  975M   0% /dev
tmpfs           989M     0  989M   0% /dev/shm
tmpfs           989M   17M  973M   2% /run
tmpfs           989M     0  989M   0% /sys/fs/cgroup
/dev/vda2        17G  3.2G   13G  21% /
/dev/vda1       976M  163M  747M  18% /boot
tmpfs           198M     0  198M   0% /run/user/0
/dev/sda1       4.9G   20M  4.6G   1% /storage1
/dev/sda2       2.0G   47M  2.0G   3% /storage2
```

Для получения информации о наличии свободных индексных

дескрипторов необходимо вызвать команду `df -i` :

Пример:

```
$ df -i
Filesystem      Inodes  IUsed   IFree IUse% Mounted on
devtmpfs        249450    376  249074    1% /dev
tmpfs           253090      1  253089    1% /dev/shm
tmpfs           253090    513  252577    1% /run
tmpfs           253090     17  253073    1% /sys/fs/cgroup
/dev/vda2      1114112 138925  975187   13% /
/dev/vda1        65536    320   65216    1% /boot
tmpfs           253090      5  253085    1% /run/user/0
/dev/sda1       327680     11  327669    1% /storage1
/dev/sda2      1048576      3 1048573    1% /storage2
```

Примечание: Обозначения K и M присутствующие в листинге выше следует понимать, как тысячи и миллионы индексных дескрипторов.

Для того, чтобы узнать сколько пространства занимают файлы в каталоге следует использовать команду `du`, отображающую количество блоков, занимаемое каждым каталогом и каждым его подкаталогом.

Можно использовать `du -h` для отображения информации в удобных единицах:

Пример:

```
$ du -h /etc/rc.d
161K    /etc/rc.d/init.d
1.0K    /etc/rc.d/rc0.d
1.0K    /etc/rc.d/rc1.d
1.0K    /etc/rc.d/rc2.d
1.0K    /etc/rc.d/rc3.d
1.0K    /etc/rc.d/rc4.d
1.0K    /etc/rc.d/rc5.d
1.0K    /etc/rc.d/rc6.d
18K     /etc/rc.d/scripts
207K    /etc/rc.d
```

Также можно отобразить лишь суммарную информацию о каталоге, без вывода подробностей о подкаталогах. Для этого используется `du -s`

Пример:

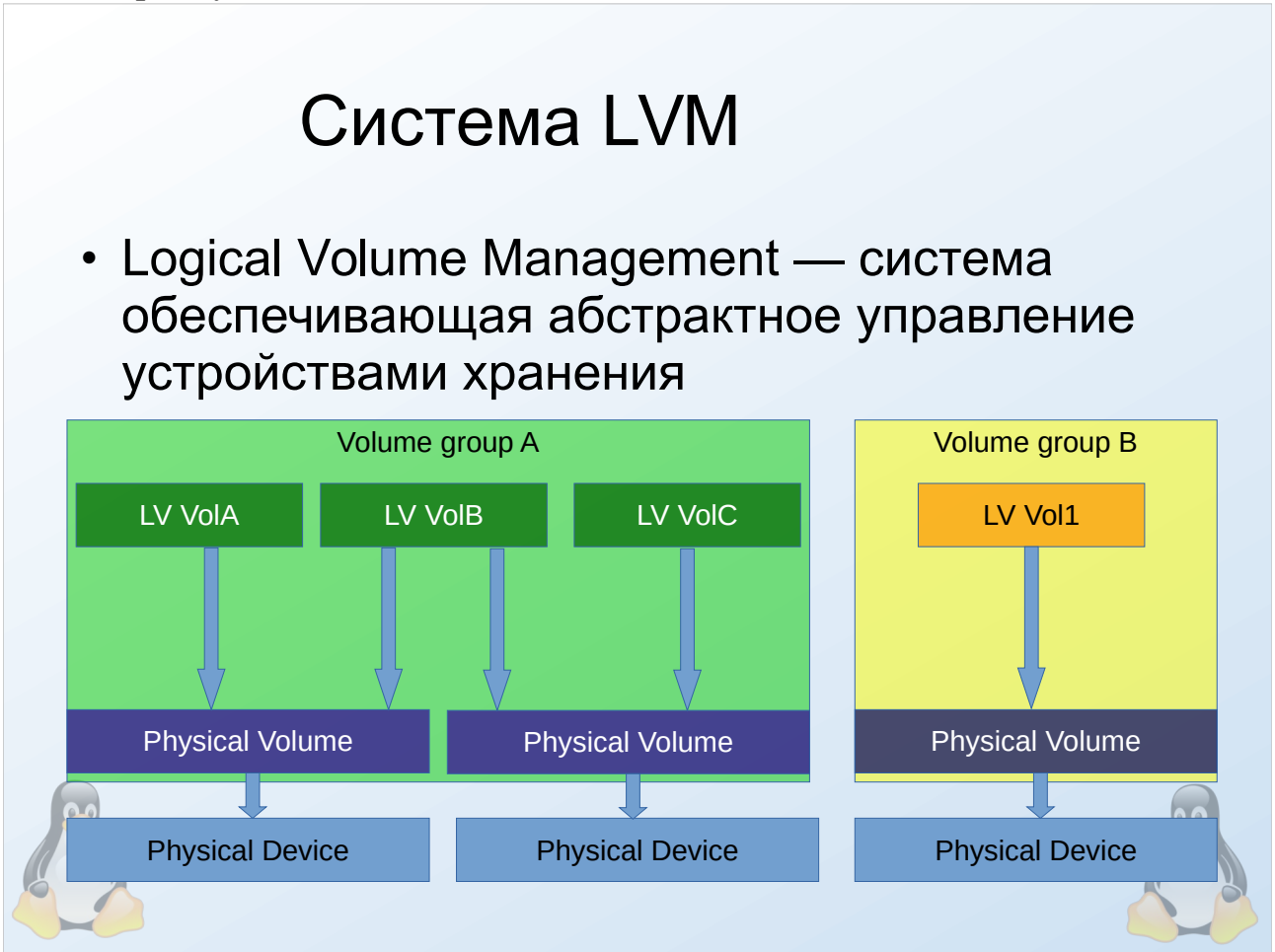
```
$ du -sh ~
467M    /home/user1
```

Примечание: В этом примере получена информация о суммарном пространстве, используемом домашним каталогом пользователя.

Глава 4. Расширенное администрирование устройств хранения данных

4.1. Система LVM.

4.1.1. Преимущества использования LVM



Система логического управления томами LVM (Logical Volume Management) обеспечивает дополнительный уровень абстракции от дисковых разделов и физических накопителей в системе. Вместо работы с системой дисковых разделов LVM предполагает операции с логическими томами, объединяющими, возможно, несколько разделов или жестких дисков.

Использование LVM позволяет объединять физические накопители в логические тома, обеспечивая более удобную работу с ними.

При использовании LVM администратор работает не с обычными файлами устройств, имена которых никоим образом не отражают суть хранимых на них данных, а с логическими устройствами, которым могут быть

назначены удобные имена.

Система LVM обеспечивает удобные возможности изменения размеров логических томов, существенно упрощая таким образом планирование исходного разбиения жесткого диска на разделы при установке системы.

Система LVM обеспечивает возможность получения так называемых snapshots - моментальных снимков состояния логического тома. Суть их состоит в фиксации информации на диске в определенный момент времени. Это используется, например, при необходимости резервного копирования на лету.

4.1.2. Терминология LVM

Наивысший уровень абстракции в LVM - это группа томов (volume group VG). Она объединяет наборы логических и физических томов в единую административную сущность.

Физический том (physical volume - PV) - обычно жесткий диск, устройство RAID или раздел жесткого диска.

Логический том (logical volume - LV) - субъект LVM, содержащий конкретную файловую систему. Доступен, как блочный файл устройства.

Физический экстент (physical extent - PE) - последовательность блоков жесткого диска, предназначенная для хранения одного какого-либо файла. Физические экстенты имеют одинаковые размеры со всеми логическими экстентами в одной группе томов.

Логический экстент (logical extent - LE) - непрерывная последовательность блоков логического тома. Размер логических экстентов одинаков для всех логических томов в группе томов.

Используется в основном две стратегии отображения логических экстентов в физические:

линейное отображение, в случае которого логические экстенты размещаются на физических томах последовательно. То есть, например, экстенты с первого по сотый - на первом физическом томе, а со сто первого по двухсотый - на втором физическом томе.

отображение с чередованием, в котором допускается части логических экстентов размещать на различных физических томах.

4.1.3. Использование LVM.

Перед использованием LVM физический диск или дисковый раздел

необходимо инициализировать командой pvcreate.

Пример:

```
pvcreate /dev/sdd
```

Если для LVM используется раздел диска, то вначале необходимо установить с помощью команды fdisk тип этого раздела 0x8e. Далее следует использовать команду pvcreate аналогично тому, как она применялась при инициализации диска.

Пример:

```
pvcreate /dev/sda5
```

Далее требуется создать группу томов (VG). Для этого используется команда vgcreate.

Пример: Создание группы томов на двух устройствах:

```
vgcreate testvg /dev/hda2 /dev/hdb1
```

В случае использования в системе devfs необходимо использовать реальные имена файлов устройств вместо символических ссылок на эти файлы.

Пример: Для системы, использующей devfs:

```
vgcreate testvg /dev/ide/host0/bus0/target0/lun0/part2 \
          /dev/ide/host0/bus0/target1/lun0/part1
```

Для вновь созданной группы томов будет установлен размер экстента по умолчанию, равный 4 Mb. При необходимости использования иного размера экстента следует указать требуемый размер после ключа -s.

После создания группы томов необходимо активировать эту группу томов. Это позволяет сделать команда vgchange -ay.

Пример: Активирование группы томов:

```
vgchange -ay testvg
```

Эта же команда vgchange, но с ключами -an позволяет деактивировать группу томов.

Пример: Деактивация группы томов:

```
vgchange -an testvg
```

После деактивации группы ее можно удалить с помощью команды vgremove. Однако, перед удалением следует убедиться, что в этой группе отсутствуют логические тома.

Пример: Удаление группы томов:

```
vgremove testvg
```

Команда vgextend позволяет добавлять физический том в существующую группу томов. Добавляемый физический том должен быть инициализирован.

Пример: добавлние физического тома /dev/sdd в группу томов testvg

```
vgextend testvg /dev/sdd
```

Существует возможность удаления физического тома из группы томов. Это можно сделать с помощью команды vgreduce.

Перед удалением физического тома из группы томов необходимо убедиться в том, что данный физический том не используется ни одним логическим томом. Такую информацию можно получить, используя команду pvdisplay.

Пример:

```
//    Проверка используется ли физический том:
   pvdisplay /dev/sdd
//    Удаление физического тома:
   vgreduce testvg /dev/sdd
```

Команды vgdisplay и pvdisplay предоставляют информацию, соответственно, о состоянии группы томов и физических томов в группе. Используя полученную с помощью этих команд информацию следует решить, на каких физических томах будут созданы логические тома.

Собственно создание логического тома осуществляется командой lvcreate, причем логический том может быть создан с использованием как линейного отображения на физические тома, так и с использованием чередования.

Пример:

```
lvcreate -L2000 -nfirstlv testvg
```

Примечание: данная команда создаст линейный логический том (LV), принадлежащий группе (VG) testvg. Логический том будет в данном случае именоваться, как firstlv, и ему будет соответствовать блочное устройство /dev/testvg/firstlv. Размер тома - 2 Гб.

```
lvcreate -l100 -i2 -I16 -nsecondlv testvg
```

Примечание: Команда, приведенная выше, создаст логический том с чередованием двух участков (опция -i2). Каждый участок имеет размер 16 Кб. Размер данного логического тома определяется количеством логических экстентов (LE), установленному в 50 (опция -l100).

Если необходимо удалить логический том, то для этого можно использовать команду lvremove.

Пример:

```
umount /dev/testvg/firstlv
lvremove /dev/testvg/firstlv
```

Примечание: В этом случае будет удален логический том /dev/testvg/firstlv.

Размер логического тома может быть увеличен с помощью команды lvextend, которой следует указать после опции -L либо размер, на который необходимо увеличить размер логического тома (-L+1G), либо желаемый размер увеличиваемого тома (-L10G).

Пример:

```
lvextend -L+1G /dev/testvg/secondlv
```

Примечание: Команда с использованием размера, на который увеличивается том:

```
lvextend -L10G /dev/testvg/secondlv
```

Примечание: Команда, увеличивающая размеры логического тома до 10 Гб:

Увеличение логического тома не означает автоматическое изменение размера файловой системы на нем.

Размер файловой системы могут быть изменены с помощью утилит,

предназначенных для конкретных файловых систем: -

ext - утилита resize2fs;

reiserfs - утилита resize_reiserfs;

xfs - утилита xfs_growfs.

Файловая система ext может быть как размонтирована так и смонтирована перед изменением ее размера.

При использовании xfs она может быть подвергнута изменению размера только тогда, когда она смонтирована. Причем, при изменении размера xfs нельзя указывать файл устройства. Вместо него необходимо указать точку монтирования:

Пример:

```
xfs_growfs /home
```

Примечание: Здесь будет произведено изменение размера файловой системы xfs, смонтированной в каталоге /home.

Пакет LVM предоставляет команду lvreduce, позволяющую уменьшать размеры логических томов. Размер логического тома не может быть меньше, чем размер файловой системы на нем.

ФС xfs уменьшать нельзя.

Пример:

```
lvreduce -L-2G /dev/testvg/secondlv
```

Примечание: Данная команда уменьшает размер логического тома на 2 Гб.

Уменьшение размера тома производится в три этапа.

Уменьшается ФС на томе, так чтобы размер ФС был чуть меньше предполагаемого размера тома

Уменьшается сам том.

ФС увеличивается, чтобы занть весь том.

Пример:

```
resize2fs /dev/testvg/firstlv 990M
```

Примечание: Только для размонтированной файловой системы ext2:

```
lvreduce -L1G /dev/testvg/firstlv
```

```
resize2fs /dev/testvg/firstlv
```

4.2. Установка квот на дисковое пространство

Система **diskquota** обеспечивает механизм для управления используемым дисковым пространством. Ограничения могут быть установлены для каждого пользователя в отдельности, для любой или для всех файловых систем. Система ограничений (**quota**) будет предупреждать пользователей, когда они превысят свой дозволенный лимит, но будет позволять использовать некоторое дополнительное пространство для текущей работы. Система ограничений (**quota system**) является частью ядра LINUX.

Команда **quota** позволяет просмотреть любые ограничения дискового пространства для каждого пользователя.

Доступны два типа ограничений, которые могут быть наложены на пользователя, обычно, если используется одно из ограничений, то и второе тоже будет использоваться. Ограничение может быть установлено как на все дисковое пространство пользователя, которое используется этим пользователем, так и на число файлов (**inodes**), которыми он может владеть. Quota обеспечивает информацию на ограничения, которые были установлены системным администратором на каждую из областей, которые используются в данный момент. Ограничения по **inodes** и **block** накладываются как на uid (идентификатор пользователя), так и на gid (идентификатор группы). Если вы входите в группу, которая превысила наложенное на нее ограничение, то вы не сможете использовать дисковое пространство, даже если вы все еще можете использовать его как пользователь.

Существуют четыре числа для каждого ограничения:

используемое в данное время ограничение;

"мягкое" ограничение (softlimit);

"жесткое" ограничение (hardlimit);

промежуток времени, после истечения которого "мягкое" ограничение интерпретируется как "жесткое".

"Мягкое" ограничение определяет число блоков размером 1Кбайт, которое пользователь может немного превысить. "Жесткое" ограничение не может быть превышено ни каким образом. Если пользователь пытается превысить данное число, то он получает сообщение о невозможности сделать это. При этом ядро возвращает код ошибки **EDQUOT**. После того как пользователь превысит доступное для него "мягкое" ограничение (softlimit)

устанавливается время, после истечения которого "мягкое" ограничение становится "жестким" (hardlimit). Обычно срок этого периода истекает после 7 дней (1 неделя). В этот период времени пользователь может удалить ненужные ему файлы, после чего он вновь может использовать "мягкое" ограничение до момента истечения указанного промежутка времени. После истечения указанного промежутка времени "мягкое" ограничение становится "жестким" и у пользователя больше нет ресурсов для создания новых файлов.

Для того чтобы установить систему ограничений дискового пространства (quota) в ОС, системному администратору необходимо сделать несколько шагов:

выбрать файловую систему, на которую будут накладываться ограничения;

разрешить (включить) систему ограничений;

произвести проверку файловой системы на ограничения дискового пространства;

произвести проверку ограничений дискового пространства как для пользователей, так и для групп;

запретить ограничения для пользователей и групп.

В первую очередь необходимо решить, на какую файловую систему необходимо наложить ограничения (quotas). Чаще всего ограничения накладываются на файловую систему, в которой располагаются домашние каталоги пользователей или на файловую систем, которая смонтирована в каталог **/usr**, и пользователи имеют право записывать на нее информацию. Для того чтобы разрешить ограничения на дисковое пространство на необходимой файловой системе, вы должны отредактировать файл **/etc/fstab**, добавив к указанной системе опции для ограничения дискового пространства (как для пользователей, так и для групп).

Отредактированный файл **/etc/fstab** может иметь вид:

```
 #
 # /etc/fstab
 #
 /dev/hda1               /              ext2    defaults
 /dev/hda2               none           swap    sw
 /dev/hda3               /usr           ext2    defaults
  /dev/hdb1                             /usr/users            ext2
defaults,usrquota,grpquota
 /dev/hdb2               /usr/src       ext2    defaults,usrquota
 none                    /proc          proc    defaults
```

Зарезервированное слово "usrquota" в поле опций включает ограничение дискового пространства (quotas) для пользователей (userquota) на данном устройстве. Зарезервированное слово "grpquota" включает ограничение дискового пространства для групп (groupquota) на данном устройстве. Во то время, когда вы используете опции "usrquota" и "grpquota" без '=', ваши файлы ограничений (quotafile) будут находиться в корневом каталоге каждой файловой системы, в которой используются ограничения на дисковое пространство. Файл называемый "aquota.user" будет использоваться для ограничений пользователей, а файл "aquota.group" будет использоваться для ограничений групп. Однако вы сами можете определить ваши файлы ограничений. Например, строка "**usrquota=/usr/adm/quotasrc.user**" установит файл ограничений для пользователей в каталоге **/usr/adm**, который будет называться "**quotarc.user**". Однако будьте внимательны и отслеживайте длину строки в файле **/etc/fstab** (смотрите ее определение в файле **mntent.h**).

Периодически (в основном после некорректной перезагрузки системы и во время первого разрешения ограничений на дисковое пространство) записи, содержащиеся в файле ограничений, должны быть проверены на целостность действительного числа блоков и файлов, выделенных для пользователя. Для выполнения этой операции может быть использована команда **quotacheck**.

Данную команду необязательно выполнять для неподмонтированных файловых систем или на файловых системах, на которых отключено ограничение на дисковое пространство (quotas). Для проверки файловой системы на число блоков, используемых пользователем, а так же для установки и изменения всех файлов ограничений (quotafiles), выполните команду:

```
quotacheck -avug
```

Вы можете вставить данную команду в один из rc-скриптов и запускать ее на файловой системе так же, как и **fsck**, только тогда, когда не установлен флаг **fastreboot**. Данная программа не поддерживает одновременную параллельную проверку нескольких файловых систем. Для того чтобы включить систему ограничений на дисковое пространство на вашем компьютере, добавьте строку **/usr/bin/quotaon -avug** в один из ваших **/etc/rc-файлов**. Данная команда будет включать поддержку ограничений дискового пространства во время загрузки вашей системы.

Только суперпользователь может использовать команду quota для проверки используемых ограничений любого пользователя и команду repquota

для проверки используемого пространства и ограничений для всех пользователей на данной файловой системе.

Для того чтобы получить информацию об ограничениях, наложенных на всех пользователей, необходимо выполнить команду:

```
repquota -ua
```

после чего на экране можно будет видеть таблицу, в которой указаны ограничения, наложенные на каждого пользователя.

Содержимое экрана после выполнения выше приведенной команды может иметь вид:

```
                        Block limits                File limits
User            used    soft     hard  grace    used  soft  hard
grace
root      --  487082       0        0          12147     0     0
daemon    --     399       0        0              2     0     0
news      --     626       0        0             46     0     0
```

Для получения информации о группах необходимо выполнить команду:

```
repquota -ga
```

Для редактирования ограничения дискового пространства необходимо использовать программу **edquota**. Для редактирования ограничений дискового пространства для конкретного пользователя

необходимо использовать программу **edquota** с опцией **-u**, а для группы необходимо использовать ту же программу, но с опцией **-g**. Редактировать необходимо только число , которое следует за словом *hard* или *soft*. Например, после выполнения команды:

```
edquota -u cola
```

содержимое экрана может иметь вид:

```
Quotas for user cola:
/dev/hdb3: blocks in use: 345, limits (soft = 1, hard = 0)
           inodes in use: 94, limits (soft = 0, hard = 0)
```

После чего вы можете изменить значения, указанные после слов hard и soft. При редактировании используется редактор, который указывается в переменной окружения **EDITOR**.

Для каждой файловой системы, имеющей ограничение на дисковое пространство, вы видите две строки. Слово **soft** означает, что на данную файловую систему наложено "мягкое" ограничение (softlimit), при этом

пользователь или группа имеют некоторый интервал времени (grace period), в течении которого они могут превысить указанное значение. Данный интервал времени можно изменить с помощью команды **edquota -t**.

Например, после выполнения команды:

```
edquota -t
```

содержимое экрана может иметь вид:

```
Time units may be: days, hours, minutes, or seconds
Grace period before enforcing soft limits for users:
/dev/hdb3: block grace period: 10 minutes, file grace period: 10
minutes
```

После чего вы можете изменить значения, указанные по умолчанию. По умолчанию используется значение, определенное в файле <LINUX quota.h>. Если пользователь или группа не удалят лишних файлов в течении указанного периода времени, то данные файлы переходят в разряд "жестких" ограничений (hardlimit). Слово **hard** означает, что на данную файловую систему наложено "жесткое" ограничение. Жесткое ограничение является максимальным значением, которое может иметь пользователь или группа на данной файловой системе. Пользователь/группа не может иметь больше файлов или inodes, чем указано в hard.

Строка:

```
/dev/hdb3: blocks in use: 345, limits (soft = 1, hard = 0)
```

сообщает, какое число блоков может быть выделено для данного пользователя или группы.

Строка:

```
inodes in use: 94, limits (soft = 0, hard = 0)
```

сообщает, какое число inode (файлов, устройств, поименованных каналов (pipes) и т.д.) может быть выделено для данного пользователя или группы.

В большинстве случаев вам приходится иметь группу пользователей, которые должны иметь одинаковые ограничения на дисковое пространство на указанной файловой системе. Самый быстрый путь - отредактировать ограничения на дисковое пространство для этих пользователей и создать прототип одного из пользователей или группы для всех других. С помощью команды:

```
edquota -u <имя_пользователя/группы_который_станет_прототипом>
```

отредактировать ограничения на дисковое пространство, после чего с помощью команды:

```
edquota -p <имя_прототипа> *
```

отредактировать ограничения для всех оставшихся пользователей.

Например, для того чтобы в качестве ограничений дискового пространства (quota) для пользователя ola использовались ограничения, наложенные на пользователя cola, вам необходимо выполнить команду:

```
edquota -p cola ola
```

Для проверки ограничений дискового пространства для пользователя и группы используется программа quota.

Синтаксис этой программы:

```
quota [-guqv]
quota [-qv] -u username ...
quota [-qv] -g groupname ...
```

Используйте опцию **-v** для вывода информации о:

файловых системах, которые не имеют включенными ограничения на дисковое пространство (quota);

файловых системах, на которых вы уже установили систему ограничения дискового пространства, но на которых еще не занят ни один блок.

Используйте опцию **-q** для просмотра файловых систем, на которых превышено значение "мягкого" (softlimit) и "жесткого" (hardlimit) ограничения.

Опция **-g** дает вам возможность просмотреть все ограничения, которые наложены на группы, членом которых вы являетесь.

Например, с помощью команды:

```
quota -u cola
```

вы можете получить информацию об ограничениях дискового пространства, которые наложены на пользователя cola. Содержимое экрана после выполнения данной команды может иметь вид:

```
Disk quotas for user cola (uid 1002):
  Filesystem   blocks    quota    limit    grace    files    quota
limit  grace
 /dev/hdb3     345*       1        0    none       94        0        0
```

Если вы захотели запретить ограничения, наложенные на какого-то пользователя или на группу, то вам необходимо воспользоваться программой

edquota для редактирования значений **hard** и **soft**. Установите эти значения в 0. После чего для данного пользователя или группы не будет существовать ограничений на дисковое пространство, и он/она сможет создавать неограниченное количество файлов (ограниченное только дисковым пространством).

В данной инструкции будет рассмотрено создание образа файловой системы для хранения нужной нам информации и последующего его подключения в точку монтирования.

Создайте каталог для точки монтирования образа:

```
mkdir -p /data/share/dirquota
```

Создайте образ файловой системы с указанием его объема:

```
dd if=/dev/zero of=/data/quota.img bs=1024 count=100000
```

Отформатируйте файловый образ:

```
mkfs.ext4 /data/quota.img
```

Примонтируйте образ в точку монтирования:

```
mount -o loop /data/quota.img /data/share/dirquota
```

Для автоматического монтирования добавьте строку в fstab:

```
nano /etc/fstab /data/quota.img/ /data/share/dirquota ext4 loop 1
2
```

Для увеличения или уменьшения объема файловой системы можно использовать resize2fs:

```
resize2fs -p quota.img 1000M
```

Для проверки целостности выполните:

```
e2fsck -f quota.img
```

4.3. Применение шифрования дисков.

Шифрование данных один из элементов по обеспечению физической безопасности компьютера. Даже если злоумышленник получит физический доступ к системе, то ему еще потребуется расшифровать файлы на диске.

В Линукс вы можете выбрать, какой объем данных следует зашифровать.

Прежде чем шифровать данные, необходимо провести предварительное планирование. Следует учесть следующие моменты:

Один из ключевых вопросов — что нужно шифровать? И нужно ли вообще что-либо шифровать? Чтобы ответь на эти вопросы нужно определить возможные угрозы системе и способно ли шифрование от них защитить.

Определите метод шифрования.

Какой способ будет использоваться для доступа к шифрованным данным

Автоматический. Ключи шифрования находятся в файле. Наименее безопасный способ.

Полуавтоматический. Ключ шифрования находятся на внешних устройствах.

Ручной. Помимо ключа шифрования или вместо него используется пароль.

Как будет производиться резервное копирование данных и ключей. Архивация данных может быть как в шифрованном виде так и в открытом.

Надо учитывать накладные расходы на шифрование. Процесс шифрования и расшифровки требует значительных вычислительных ресурсов.

Существует два принципиально разных подхода к шифрованию:

Блоковое шифрование, в котором шифруются блоки данных на диске и поверх этих блоков создается файловая система.

Шифрование на уровне файлов. В такой системе шифруется содержимое файлов.

Система dm-crypt используется для блокового шифрования.

Система работает посредством модуля ядра, который производит отображение шифрованного диска в виртуальное устройство. Такое устройство можно использовать как обычное устройство хранения.

Для управления ключами предполагается использовать систему LUKS. Но dm-crypt может работать и без LUKS.

dm-crypt имеет обратную совместимость с системами LoopAES и TrueCrypt. Но все же не является их полной заменой.

Определите что вы будете шифровать это может быть целый диск, раздел, LVM том. Например, можно зашифровать весь USB диск /dev/sdb.

Подготовка диска — заполнение диска случайными данными. Этот этап можно пропустить, но тогда данные будут уязвимыми.

Пример:

```
shred -v --iterations=1 /dev/sdb
```

Утилитой cryptsetup определите параметры шифрования.

Пример:

```
[root@sl0 ~]# cryptsetup  --cipher aes-cbc-essiv:sha256 --key-size
256 luksFormat /dev/sdb
WARNING!
========
This will overwrite data on /dev/sdb irrevocably.
Are you sure? (Type uppercase yes): YES
Enter passphrase:
Verify passphrase:
```

Создайте виртуальное устройство для шифрованного диска

Пример:

```
[root@sl0 ~]# cryptsetup luksOpen /dev/sdb encsdb
Enter passphrase for /dev/sdb:
[root@sl0 ~]# ls /dev/mapper/
centos-root  centos-swap  control  encsdb
```

Создайте файловую систему

Пример:

```
[root@sl0 ~]# mkfs.ext3 /dev/mapper/encsdb
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
65408 inodes, 261632 blocks
13081 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8176 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376
Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

Смонтируйте файловую систему.

Пример:

```
[root@sl0 ~]# mount /dev/mapper/encsdb /mnt
[root@sl0 ~]# ls /mnt/
lost+found
[root@sl0 ~]# echo 123 > /mnt/test
[root@sl0 ~]# umount /mnt
[root@sl0 ~]# cryptsetup luksClose encsdb
[root@sl0 ~]# ls /dev/mapper/
centos-root  centos-swap  control
[root@sl0 ~]# mount /dev/sdb /mnt
mount: unknown filesystem type 'crypto_LUKS'
```

Показанный выше пример показывает как применить шифрование в ручном виде.

Подключение шифрованных разделов во время старта системы контролируется целью cryptsetup.target

Пример:

```
[root@sl0 ~]# systemctl status cryptsetup.target
● cryptsetup.target - Encrypted Volumes
      Loaded:  loaded  (/usr/lib/systemd/system/cryptsetup.target;
static; vendor preset: disabled)
   Active: active since Wed 2017-07-26 21:46:48 +05; 3min 39s ago
     Docs: man:systemd.special(7)
```

Для настройки этой цели используется файл /etc/crypttab.

Пример:

```
[root@sl0 ~]# cat /etc/crypttab
encsdb /dev/sdb /root/myencpass auto
```

Первое поле файла имя виртуальное устройство

Второе поле имя шифрованного устройство

Третье поле файл с паролем для разблокирования. Вместо имени файла можно указать none или «-». Еще один вариант указать имя файла устройства, которое генерирует случайные данные. Последний случай применяется для подключения шифрованных разделов swap или tmp. Файл с ключом может иметь любое содержание, но его надо добавить командой cryptsetup luksAddKey

Пример:

```
dd if=/dev/urandom of=/root/myencpass bs=512 count=1
```

```
chmod 400 /root/myencpass
cryptsetup luksAddKey /dev/sdb /root/myencpass
Enter any existing passphrase:
```

Четвертое поле опции подключения.

Глава 5. Архивирование и резервное копирование в РЕД ОС

5.1. Утилиты для сжатия файлов.

В UNIX и GNU/Linux системах команды архивирования отделены от утилит сжатия файлов.

Все утилиты сжатия осуществляют компрессию файлов, указанных в качестве аргументов. При этом к исходным названиям файлов добавляются стандартные суффиксы, перечисленные ниже, а права доступа и владения сохраняются.

В GNU/Linux используются следующие утилиты сжатия:

`gzip` - используется наиболее часто, сжатые файлы имеют суффиксы `.gz`.

xz (lzma) — использует алгоритм LZMA, что обеспечивает очень высокую степень сжатия, сжатые файлы имеют суффиксы `.xz`.

`bzip2` - обеспечивает лучшую степень сжатия, чем `gzip`, но хуже xz, сжатые файлы имеют суффиксы `.bz2`.

`compress` - стандартная UNIX утилита сжатия, в GNU/Linux используется реже, чем предыдущие, сжатые файлы имеют суффиксы `.Z`.

Пример: Для сжатия файлов достаточно указать их в качестве аргументов:

```
$ ls -l distfiles.*
-rw-r--r--       1  user1      user1          239263  Авг  26  23:10
distfiles.lst
-rw-r--r--       1  user1      user1         1967881  Авг  22  15:01
distfiles.txt
$ gzip distfiles.*
$ ls -l distfiles.*
-rw-r--r--       1  user1      user1           63298  Авг  26  23:10
distfiles.lst.gz
-rw-r--r--       1  user1      user1          187311  Авг  22  15:01
distfiles.txt.gz
```

Примечание: В этом примере командой `gzip` *были сжаты два файла. После сжатия к их названиям был добавлен суффикс* `.gz`.

Если необходимо получить информацию о сжатых файлах `gzip`, то

следует использовать опцию `-l`

Пример:

```
$ gzip -l distfiles.*
         compressed        uncompressed  ratio uncompressed_name
              63298              239263  73.6% distfiles.lst
             187311             1967881  90.5% distfiles.txt
             250609             2207144  88.6% (totals)
```

Ниже приведены наиболее часто используемые опции команды `gzip` :

`-d` - произвести декомпрессию сжатых файлов, как `gunzip` ;

`-c` - копируют сжатое содержимое файлов в стандартный поток вывода, исходные файлы остаются неизменными;

`-r` - рекурсивно сжимать содержимое каталога;

`-S` - установить иной, чем `.gz` , суффикс;

`-t` - тестировать содержимое архива;

`-v` - выводить подробную информацию о процессе работы команды.

Пример: следующая команда тестирует сжатые файлы:

```
$ gzip -tv distfiles.*
distfiles.lst.gz:        OK
distfiles.txt.gz:        OK
```

Для декомпрессии сжатых `gzip` файлов удобно применять команду `gunzip` .

Пример:

```
$ gunzip -v distfiles.*
distfiles.lst.gz:        73.6% -- replaced with distfiles.lst
distfiles.txt.gz:        90.5% -- replaced with distfiles.txt
```

Команда `zcat` выводит в стандартный поток вывода содержимое сжатых `gzip` файлов.

В командах xz и `bzip2` реализован иной алгоритм сжатия, чаще всего обеспечивающий более высокий уровень компрессии.

Чем выше степень сжатия, тем больше потребуется процессорного времени для работы.

Многие опции команды `xz` и `bzip2` функционально идентичны соответствующим опциям `gzip` (но не все!):

Пример:

```
$ time gzip -c big_file.txt > big_file.txt.gz
real 0m2,354s
user 0m0,892s
sys 0m0,040s
$ time bzip2 -c big_file.txt > big_file.txt.bz2
real 0m7,613s
user 0m7,546s
sys 0m0,041s
$ time xz -c big_file.txt > big_file.txt.xz
real 0m9,674s
user 0m9,563s
sys 0m0,078s
$ stat -c '%s %n' big_file.txt*
49765537 big_file.txt
3313476 big_file.txt.bz2
4282449 big_file.txt.gz
3232704 big_file.txt.xz
```

Примечание: Если сравнить результаты работы, то заметно, что степень сжатия у xz команды выше, однако xz и работает медленнее остальных.

```
$ time xz -T4 -c big_file.txt > big_file.txt.xz
real 0m5,575s
user 0m10,025s
sys 0m0,177s
$ time xz -T8 -c big_file.txt > big_file.txt.xz
real 0m5,642s
user 0m10,159s
sys 0m0,174s
```

Примечание: даже запущенная в несколько потоков xz (опция -T) все равно медленнее gzip.

5.2. Команда tar.

Один из наиболее часто используемых инструментов резервного копирования - команда `tar` (tape archive).

С помощью этой команды можно создавать архивы файлов и каталогов, заданных ей в качестве аргументов.

Команда `tar` поддерживает BSD стиль задания опций, поэтому перед опциями можно не указывать знак дефиса - .

В команде tar одна из опций задает режим работы. Такая опция в команде может быть только одна, и обычно она указывается первой.

Наиболее важные из операционных опций:

-c, --create — создать архив, существующий файл перезаписывается;

-x, --extract, --get — извлечь файлы из архива;

-t, --list — просмотреть архив;

-r, --append — добавить файлы к архиву.

Опции `-f` команды `tar` указывает файл, в который будет помещен или извлечен архив.

Опция `-v` заставляет команду `tar` выводить информацию об обрабатываемых файлах.

Пример: приведенная ниже команда поместит архив, содержащий всю информацию в каталоге `/home` , на магнитную ленту.

```
# tar cvf /dev/st0 /home
```

Примечание: После выполнения этой команды на магнитной ленте будет создан архив файлов в каталоге (со всеми подкаталогами) `/home` *. Для работы с магнитными лентами предназначены символьные файлы устройств, поэтому информация на них записывается в виде потока байт.*

В различных UNIX системах поведение команды `tar` может отличаться в деталях.

Пример: используемая в GNU/Linux версия `tar` по умолчанию не сохраняет в архиве абсолютные имена файлов, удаляя из них лидирующую косую черту - имя корневого каталога. Так, имя файла `/home/user1/.bashrc` будет преобразовано в `home/user1/.bashrc` , что позволяет разархивировать содержимое архива в текущем каталоге.

Команда `tar` позволяет создавать архивы и сразу сжимать их при создании утилитами `gzip` при использовании опции `-z`, xz при использовании опции -J или `bzip2` при использовании опции `-j` . Могут быть использованы и другие утилиты сжатия.

Пример: содержимое каталогов `/bin` и `/sbin` помещено в архив `binaries.tar.gz` .

```
$ tar czvf  binaries.tar.gz /{,s}bin
```

Примечание: Стандартным расширением для архивов `tar` *является* `.tar` *. Если архив сжат, то к имени архива принято добавлять соответствующий суффикс (например,* `.gz`*).*

Для просмотра содержимого архива следует использовать опцию `-t` . Если архив сжат, то необходимо установить опцию, соответствующую примененной утилите сжатия.

Пример: для просмотра содержимого архива `binaries.tar.gz` следует использовать команду:

```
$ tar tzvf  binaries.tar.gz
```

Для извлечения файлов из архива следует использовать опцию `-x`. При использовании GNU версии команды `tar` содержимое архива будет извлечено в текущем каталоге. При этом будут созданы все подкаталоги, которые находятся в архиве.

Пример: команда, показанная ниже, развернет содержимое архива `binaries.tar.gz` в текущем каталоге:

```
$ tar xzvf  binaries.tar.gz
```

Если вместо имени файла архива после опции `-f` в командной строке `tar` указан знак дефиса `-` , то архив будет взят из стандартного потока ввода.

Пример: приведенная ниже команда, по действию аналогична предыдущей:

```
$ gzip -dc binaries.tar.gz | tar xvf -
```

Примечание: В этом примере архив сначала подвергается декомпрессии командой `gzip` *, причем использование опции* `-c` *заставляет* `gzip` *выводить имя обработанного файла в стандартный поток вывода. Из него через конвейер имя разжатого файла передается на стандартный поток ввода команде* `tar` *, которая извлекает файлы из уже не сжатого архива.*

Утилита tar по умолчанию сохраняет только стандартные атрибуты файлов, например права доступа, владельца и пр.. Если необходимо также сохранять расширенные атрибуты, то нужно явно это указать в команде. Это опции --acls, --selinux, --xattrs.

5.3. Виды резервного копирования.

Резервное копирование - это единственная операция, которая может гарантировать сохранность данных, находящихся в вычислительной системе.

Процесс архивирования данных можно разделить на следующие виды:

Полное (обычное, full, normal)

Инкрементальное (добавочное, incremental)

Разностное (differential)

Архивирование по признаку (data-specific).

При полном копировании архивируется все, что вы указали программе архивации. При этом программа архивации не анализирует данные, а просто помещает их в архивную копию.

Для инкрементального и разностного архивирования необходимо предварительно создать полную архивную копию данных.

В инкрементальном копировании в архивную копию попадают данные, которые изменились с момента последней архивации.

В разностном копировании в архивную копию попадают данные, изменившиеся с момента полной архивации.

Резервное копирование данных должно производиться тогда, когда эти данные не изменяются. В противном случае нельзя гарантировать сохранности этих данных.

Желательно регулярно выполнять резервное копирование в часы минимальной нагрузки на систему.

Всегда проверяйте ваши архивные копии, на предмет возможности восстановления данных.

Проводите пробные восстановления.

Не храните архивные копии в том же месте, где находится система, которую вы архивируете.

5.4. Разработка плана инкрементального архивирования.

При планировании резервного копирования следует определить следующие моменты:

какие данные должны быть скопированы и где они находятся (в каких каталогах или файловых системах, а также, возможно, на каких компьютерах);

какой тип носителей данных будет использован для резервного копирования;

с какой периодичностью будет производиться копирование данных;

каким образом и с какой частотой будет осуществляться ротация резервных носителей данных;

кто будет уполномочен производить резервное копирование;

будет ли требоваться человеческий контроль за ходом копирования или

же копирование будет производиться полностью автоматически;

в какие моменты времени должно осуществляться резервное копирование;

где и как будет производиться хранение носителей резервных копий;

каков будет порядок восстановления утраченных данных из резервных копий;

максимальная допустимая продолжительность периода времени, необходимого для восстановления данных.

При разработке плана резервного копирования важно оценить два важнейших параметра:

RPO (recovery point objective) – допустимая потеря данных. Любая информационная система должна обеспечивать (внутренними ли средствами, или сторонними) защиту своих данных от потери выше приемлемого уровня.

RTO (recovery time objective) – допустимое время восстановления данных Любая информационная система должна обеспечивать (внутренними ли средствами, или сторонними) возможность восстановления своей работы в приемлемый срок.

Суммарное время на восстановление данных состоит как минимум из:

времени, необходимого для поиска и доставки носителя резервной копии требуемых данных;

времени на поиск требуемой информации на носителе (если поиск допускается);

времени копирования информации с резервного носителя.

Разработка плана архивного копирования копирования и неукоснительное соблюдение этого плана помогает (но не гарантирует) сократить время простоев при аварийных сбоях, связанных с поиском и восстановлением данных.

При планировании инкрементального архивирования обычно составляют расписание, в соответствии, с которым, проводят архивацию.

В расписании необходимо указать:

периодичность полного архивирования;

периодичность и уровень инкрементального архивирования.

Пример: плана резервного копирования. Предположим имеются данные, с которыми ежедневно работают пользователи. Эти данные важны для нормального функционирования компании. Предполагается, что небольшая часть пользователей работает в выходные. При копировании требуется обеспечить целостность данных. Здесь возможны следующие варианты архивирования.

План 1.

Воскресенье	Понедельник	Вторник	Среда	Четверг	Пятница
Полный	Инкремент.	Инкремент.	Инкремент.	Инкремент.	Инкремент.

В этом плане в воскресенье, когда пользователи не работают в системе производится полное архивирование. В буднии дни инкрементальное копирование. Такой план позволяет снизить затраты времени и ресурсов на проведение архивирования, но требует более внимательного восстановление данных. Например, для восстановления всех данных за среду требуется сначала восстановить данные воскресного архивирования, затем последовательно восстанавливать данные из копий понедельника, вторника и среды.

План 2.

Воскресенье	Понедельник	Вторник	Среда	Четверг	Пятница
Полный	Разностный	Разностный	Разностный	Разностный	Разностный

В этом плане в воскресенье, когда пользователи не работают в системе производится полное архивирование. В будние дни разностное копирование. Такой план позволяет снизить затраты времени на проведение восстановления, но требует более длительного копирования данных. В этом плане требуется большее количество пространства на резервных носителях, чем в первом плане. Например, для восстановления всех данных за среду требуется сначала восстановить данные воскресного архивирования, затем данные из копии среды.

5.5. Полное и инкрементальное архивирование с помощью tar.

Один из наиболее часто используемых инструментов резервного копирования - команда `tar` (tape archive).

С помощью этой команды можно создавать архивы файлов и каталогов, заданных ей в качестве аргументов.

Опция `f` команды `tar` указывает файл, в который будет помещен архив.

Опция `c` команды предназначена для создания полного архива (create).

Опция `v` заставляет команду `tar` выводить информацию об

обрабатываемых файлах.

Пример: приведенная ниже команда поместит архив, содержащий всю информацию в каталоге `/home` **, на магнитную ленту.**

```
# tar cvf /dev/st0 /home
```

Примечание: После выполнения этой команды на магнитной ленте будет создан полный архив файлов в каталоге (со всеми подкаталогами) `/home` *. Для работы с магнитными лентами предназначены символьные файлы устройств, поэтому информация на них записывается в виде потока байт.*

Помимо создания файла архива в tar имеются опции по добавлению файлов в архив или слиянию архивов:

`-A` - добавление файлов `tar` архива в существующий архив (слияние);

`-r` - добавление файлов в конец архива;

`-u` - обновление архива версиями файлов, более новыми, чем в архиве;

Инкрементальное копирование можно организовать с помощью опции --listed-incremental (-g).

При первом вызове с опцией -g tar создает файл с информацией о времени архивирования.

При последующих вызовах с этой опцией обрабатываются только те файлы, которые изменились с момента последней архивации.

Пример:

```
tar cvf d1.1.tar --listed-incremental d1.inc d1/
```

Примечание: в данном примере создается архив с именем d1.1.tar и инкрементальный файл с именем d1.inc. Архивируется каталог d1

Для выполнения разностного архивирования команде tar необходимо подставлять всегда начальный инкрементальный файл, т.е. после полного архивирования сохранить инкрементальный файл в альтернативном месте, а перед каждым архивированием восстанавливать его.

5.6. Полное и инкрементальное архивирование с помощью cpio.

Команда `cpio` является вторым по значимости инструментом архивирования после `tar` , обладая при этом более гибкими возможностями по выбору файлов для архивирования, чем `tar` .

Примечание: Это связано с тем, что команда `cpio` *позволяет передавать через стандартный поток ввода имена файлов как для архивирования, так и имена архивов для извлечения из них файлов.*

Команда `cpio` способна работать в трех режимах, определяемых опциями:

`-o` - для копирования в архив (copy-out), в котором архивируемые файлы помещаются в архив, а сам поток байт архива копируется в выходной (output) файл;

`-i` - для копирования из архива (copy-in), в котором файлы извлекаются из архива, который передается команде на вход (input);

`-p` - проходном (pass-through), при использовании которого файлы копируются из одного каталога в другой без реального создания архива.

Чаще всего список файлов, которые должны быть помещены в архив, передается на стандартный поток ввода команды `cpio` с помощью команды `find`.

В cpio нет опций для инкрементального копирования, но организовать такое копирование можно самому. Для этого

Поместить все файлы в архив, для полного архивирования;

Создать файл, содержащий время архивирования;

Пример:

```
find directory -depth|cpio -oF dir.0.cpio && date +%y%m%d%k%M.%S > cpio.inc
```

При последующих вызовах cpio необходимо произвести следующие действия:

Извлечь время архивирования из файла, и сравнить время модификации файлов с этим временем;

Поместить найденные файлы в инкркментальный архив.

5.7. Утилита rsync

Rsync - это программное обеспечение с открытым исходным кодом, которое можно использовать для синхронизации файлов и папок с локального компьютера на удаленный и наоборот. Примечательная особенность Rsync - возможность передавать зашифрованные файлы с помощью SSH и SSL. Кроме того, здесь передача файлов выполняется в один поток, в отличие от других подобных программ, создающий отдельный поток для передачи каждого файла.

Это увеличивает скорость и убирает дополнительные задержки, которые становятся проблемой при передаче большого количества маленьких файлов. Основные параметры rsync:

`-v` - Выводить подробную информацию о процессе копирования;

`-q` - Минимум информации;

`-c` - Проверка контрольных сумм для файлов;

`-a` - Режим архивирования, когда сохраняются все атрибуты оригинальных файлов;

`-R` - Относительные пути;

`-b` - Создание резервной копии;

`-u` - Не перезаписывать более новые файлы;

`-l` - Копировать символьные ссылки;

`-L` - Копировать содержимое ссылок;

`-H` - Копировать жесткие ссылки;

`-p` - Сохранять права для файлов;

`-g` - Сохранять группу;

`-t` - Сохранять время модификации;

`-x` - Работать только в этой файловой системе;

`-e` - Использовать другой транспорт, например, ssh;

`-z` - Сжимать файлы перед передачей;

`--delete` - Удалять файлы которых нет в источнике;

`--exclude` - Исключить файлы по шаблону;

`--recursive` - Перебирать директории рекурсивно;

`--no-recursive` - Отключить рекурсию;

`--progress` - Выводить прогресс передачи файла;

`--stat` - Показать статистику передачи;

`--version` - Версия утилиты.

Rsync позволяет синхронизировать файлы и папки в пределах одной машины. Давайте сначала рассмотрим использование rsync для синхронизации файла на локальном компьютере:

```
rsync -zvh file /tmp/backups/
```

Синхронизация папок rsync выполняется так же, как и файлов:

```
rsync -zvh /var/account/* /tmp/backups/
```

Если вы хотите, чтобы все атрибуты файлов, такие, как дата изменения и создания сохранялись, необходимо использовать опцию -a:

```
rsync -azvh /var/account/* /tmp/backups/
```

Синхронизация файлов производится следующим образом. Скопируем локальную папку files, на удаленный сервер:

```
rsync -avz /home/files root@10.81.1.190:/home/user
```

По умолчанию rsync попытается использовать транспорт ssh. Если вы хотите использовать ранее созданный сервер rsync, нужно указать это явно:

```
rsync -avz /home/files rsync://10.81.1.190:/share
```

Точно также можно синхронизировать файлы с rsync из удаленного сервера:

```
rsync -avz root@10.81.1.190:/home/ /home/files/
```

5.8. Утилита Bacula

Bacula - это программа резервного копирования, позволяющая сохранять, восстанавливать и проверять данные через вашу сеть. Для Bacula существуют клиенты под Linux, Windows и Mac OS X, превращая ее в кросс-платформенное сетевое решение. Bacula состоит из нескольких компонентов и сервисов для управления тем, какие файлы сохранять и где хранить резервные копии: Bacula Director: сервис, который управляет всеми операциями резервного копирования, восстановления, проверки и архивации. Bacula Console: приложение, позволяющее взаимодействовать с Director. Существует три версии Console:

текстовая версия, основанная на командной строке.

графический пользовательский интерфейс (GUI) для Gnome, основанный на GTK+.

графический интерфейс на базе wxWidgets.

Bacula File: она же программа Bacula Client. Это приложение устанавливается на машине, на которой производится резервное копирование, и оно отвечает на данные, отправленные по запросу Director. **Bacula Storage**: программа, которая выполняет хранение и восстановление данных на физических носителях. **Bacula Catalog**: отвечает за поддержку файловых индексов и томов баз данных для всех сохраняемых файлов, допуская быстрое нахождение и восстановление сохраненных файлов. Catalog поддерживает три

различных базы данных: MySQL, PostgreSQL и SQLite. **Bacula Monitor**: позволяет отслеживать работу Director и сервисов File и Storage. На данный момент Monitor доступен только в виде GTK+ GUI приложения. Эти сервисы и приложения могут быть запущены на разных серверах и клиентах или их можно установить на одной машине, если требуется резервное копирование одного диска или тома. При установке по умолчанию пакет bacula будет использовать базу данных MySQL для Catalog. Если вы хотите использовать SQLite или PostgreSQL, установите соответственно пакет bacula-director-sqlite3 или bacula-director-pgsql. В процессе установки у вас спросят данные об администраторе базы данных и владельце базы данных bacula. Администратор базы данных требуется для получения необходимых прав на создание базы данных. Файлы настройки Bacula форматированы на основе ресурсов, включающих директивы, обрамленные фигурными скобками "{}". Каждый компонент Bacula имеет индивидуальный файл в каталоге **/etc/bacula**. Различные компоненты Bacula должны авторизовывать себя друг для друга. Это решается использованием директивы password. Например, пароль в ресурсе Storage файла **/etc/bacula/bacula-dir.conf** должен соответствовать паролю ресурса Director файла **/etc/bacula/bacula-sd.conf**. По умолчанию настраивается задание резервного копирования *Client1* для архивирования Bacula Catalog. Если вы планируете использовать сервер для резервного копирования более чем на одном клиенте, вам потребуется изменить имя этого задания на что-то более осмысленное. Для переименования отредактируйте файл **/etc/bacula/bacula-dir.conf**

```
#
# Define the main nightly save backup job
#   By default, this job will back up to disk in
Job {
  Name = "BackupServer"
  JobDefs = "DefaultJob"
  Write Bootstrap = "/var/lib/bacula/Client1.bsr"
}
```

В примере имя задания изменено на BackupServer в соответствии с сетевым именем машины. Можете заменить "BackupServer" на соответствующее сетевое имя вашего сервера или другое описательное название. Требуется использовать **Console** для запросов к *Director* по поводу заданий, но чтобы обычный пользователь мог использовать Console, он должен быть включен в группу bacula. Чтобы добавить пользователя в группу bacula

введите следующую команду в терминале:

```
adduser $username bacula
```

Замените $username на актуальное имя пользователя. Также, если вы добавили в группу текущего пользователя, вам придется выйти из системы и зайти снова, чтобы применились новые права доступа. Далее описывается как сделать резервную копию определенных каталогов на единственном сервере на локальное ленточное устройство.

Для начала требуется настроить устройство хранения. Отредактируйте **/etc/bacula/bacula-sd.conf**, добавив:

```
Device {
  Name = "Tape Drive"
  Device Type = tape
  Media Type = DDS-4
  Archive Device = /dev/st0
  Hardware end of medium = No;
   AutomaticMount = yes;                  # when device opened, read
it
  AlwaysOpen = Yes;
  RemovableMedia = yes;
  RandomAccess = no;
  Alert Command = "sh -c 'tapeinfo -f %c | grep TapeAlert'"
}
```

Этот пример для ленточного устройства DDS-4. Измените *"Media Type"* и *"Archive Device"* в соответствии с вашим оборудованием. Вы также можете раскомментировать один из примеров в этом файле.

После редактирования файла **/etc/bacula/bacula-sd.conf** сервис *Storage* требуется перезагрузить:

```
/etc/init.d/bacula-sd restart
```

Теперь добавьте ресурс Storage в файл /etc/bacula/bacula-dir.conf для использования нового устройства:

```
# Definition of "Tape Drive" storage device
Storage {
  Name = TapeDrive
  # Do not use "localhost" here
   Address = backupserver                      # N.B. Use a fully
qualified name here
  SDPort = 9103
  Password = "Cv70F6pf1t6pBopT4vQOnigDrR0v3LT3Cgkiyjc"
  Device = "Tape Drive"
```

```
  Media Type = tape
}
```

Директива *Address* должна быть полностью квалифицированным доменным именем (FQDN) сервера. Замените *backupserver* на актуальное сетевое имя. Также убедитесь, что директива *Password* соответствует строке пароля в **/etc/bacula/bacula-sd.conf.**

Создайте новый FileSet, который будет определять какие каталоги копировать, добавив:

```
# LocalhostBacup FileSet.
FileSet {
  Name = "LocalhostFiles"
  Include {
    Options {
      signature = MD5
      compression=GZIP
    }
    File = /etc
    File = /home
  }
}
```

Этот FileSet задает резервное копирование для каталогов /etc и /home. Директивы ресурса Options настраивают FileSet на создание контрольных сумм MD5 для каждого сохраненного файла и сжатие файлов с использованием GZIP.

Далее создайте новое расписание Schedule для задачи резервного копирования:

```
# LocalhostBackup Schedule -- Daily.
Schedule {
  Name = "LocalhostDaily"
  Run = Full daily at 00:01
}
```

Задание будет запускаться каждый день в 00:01. Существует множество других доступных опций расписаний.

Наконец, создайте задание:

```
# Localhost backup.
Job {
  Name = "LocalhostBackup"
  JobDefs = "DefaultJob"
  Enabled = yes
  Level = Full
```

```
  FileSet = "LocalhostFiles"
  Schedule = "LocalhostDaily"
  Storage = TapeDrive
  Write Bootstrap = "/var/lib/bacula/LocalhostBackup.bsr"
}
```

Задание будет выполнять полную резервную копию каждый день на ленту.

Каждая используемая лента должна иметь метку. Если текущая лента метки не имеет, Bacula отправит email с оповещением. Чтобы установить метку на ленту с помощью Console, введите в терминале:

```
bconsole
```

В командной строке Bacula Console введите:

```
label
```

Вам предложат выбрать один из ресурсов Storage:

```
Automatically selected Catalog: MyCatalog
Using Catalog "MyCatalog"
The defined Storage resources are:
     1: File
     2: TapeDrive
Select Storage resource (1-2):2
```

Введите новое имя тома:

```
Enter new Volume name: Sunday
Defined Pools:
     1: Default
     2: Scratch
```

Замените Sunday на соответствующую метку.

Теперь выберите накопитель:

```
Select the Pool (1-2): 1
Connecting to Storage daemon TapeDrive at backupserver:9103 ...
Sending label command for Volume "Sunday" Slot 0 ...
```

Таким образом настроена Bacula на резервное копирование локального сервера на подключенное ленточное устройство.

Глава 6. Планирование заданий в РЕД ОС

6.1. Отложенное выполнение заданий с помощью at.

Часто возникает необходимость выполнить какую-либо команду в заданный момент будущего или же в тот момент, когда загрузка системы минимальна.

Команда `at` позволяет указать момент времени, в который должна быть исполнена команда. При этом заданная команда выполняется только один раз в заданный момент времени.

Пример:

```
$ date
Втр Янв 13 22:12:38 YEKT 2004
$ at 22:20
warning: commands will be executed using /bin/sh
at> ps -ef | mail -s 'At command' user1
at> <EOT>
job 2 at 2004-01-13 22:20
```

Примечание: В `22:12` *была запущена команда at* `22:20` *. Команда* `at` *, вызванная таким образом предоставляет пользователю возможность интерактивно ввести после приглашения* `at>` *команды, которые необходимо выполнить в будущем. В этом примере была введена команда, посылающая в заданное время пользователю* `user1` *список всех процессов в системе в подробном формате.*

Помимо указания астрономического времени выполнения команды можно задавать период времени, через который должна быть выполнена заданная команда.

Пример:

```
$ at now +10 minutes
```

Примечание: В этом случае команда at выполнит команду, заданную в ее интерактивном сеансе, через десять минут относительно текущего момента времени.

Задавать команды для исполнения можно также направляя командую строку в стандартный поток ввода для команды `at` или же указывая после опции `-f` имя файла с командами.

Опция -m команды `at` посылает вызвавшему ее пользователю сообщение

по электронной почте о том, что задание выполнено.

Для просмотра списка заданий можно использовать команду `atq` или `at -l`:

Пример:

```
$ atq
3       2004-01-14 11:13 a user1
```

Примечание: При исполнении этой команды обычный пользователь получает список только его заданий, а суперпользователь - список заданий всех пользователей.

Для удаления задания из очереди следует вызвать команду `atrm` или `at -d`. Номер задания должен быть указан в качестве аргумента:

Пример:

```
$ echo 'wall Hello!' | at -m +1 minutes
warning: commands will be executed using /bin/sh
job 5 at 2004-01-14 12:04
$ atq
5       2004-01-14 12:04 a user1
$ atrm 5
$ atq
```

Примечание: В этом примере была задана команда `wall` *, которая должна была вывести сообщение* `Hello!` *на терминалы всех пользователей, находящихся в сеансе, через минуту после ее вызова. Однако, эта команда снята с исполнения командой* `atrm`.

Команда `batch` отличается от `at` тем, что она выполняет задание не через четко определенный промежуток времени, а в тот момент, когда средняя загрузка системы уменьшается до `0.8`. Значение средней загрузки системы берется из файла `/proc/loadavg`. В остальном команда `batch` аналогична команде `at`.

В системе могут иметься файлы `/etc/at.allow` и `/etc/at.deny`. Если существует файл `/etc/at.allow`, то только пользователи, перечисленные в нем, могут вызывать команду `at`.

Если этого файла нет, то проверяется наличие файла `/etc/at.deny`, в котором указываются пользователи, которым запрещено вызывать `at`.

6.2. Автоматизация выполнения регулярных рутинных задач.

В отличие от `at` команда система `cron` позволяет выполнить задание не

единожды, а выполнять его на регулярной основе с заданной периодичностью.

В различных версиях GNU/Linux используются различные пакеты `cron` .

Пример: `vixie-cron` **или** `cronie`**. Их работа несколько отличается в деталях.**

Основой регулярного выполнения заданий в любом из вариантов `cron` являются таблицы, в которых кодируется периодичность выполнения заданий.

Задания выполняет демон crond.

Обычно имеется общесистемная таблица заданий `crond` , находящаяся в файле `/etc/crontab`, и индивидуальные таблицы периодических заданий пользователей, находящиеся в файлах, обычно расположенных в каталоге `/var/spool/cron` . Имена таких файлов совпадают с именами пользователей, для которых созданы таблицы периодических заданий.

Примечание: В некоторых системах общесистемный файл `/etc/crontab` *не используется, а вместо этого в качестве общесистемной таблицы заданий* `cron` *работает таблица заданий суперпользователя.*

Файл общесистемной таблицы заданий `/etc/crontab` имеет семь полей, при этом строки, начинающиеся с решетки считаются комментариями.

Первые пять полей используются для указания периодичности выполнения задания. Шестое поле указывает от имени какого пользователя должно быть исполнено задание. В седьмом поле указывают саму команду, которая должна быть выполнена.

Формат задания времени следующий:

Минуты часа (`0-59`).

Час суток (`0-23`).

Календарное число (день месяца `1-31`).

Месяц (`1-12`).

День недели начиная с воскресенья (`0-6`).

Примечание: Если в команде задаются аргументы и опции командной строки, то команда не экранируется кавычками. Поэтому иногда создается впечатление, что в строках таблицы `crontab` *больше, чем семь полей.*

Пример:

```
# /etc/crontab: system-wide crontab
SHELL=/bin/sh
PATH=/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

```
# m h dom mon dow user   command
25 6 * * * root run-parts /etc/cron.daily
47 6 * * 7 root run-parts /etc/cron.weekly
52 6 1 * * root run-parts /etc/cron.monthly
08 * * * * root /usr/X11R6/bin/log_clean
```

Примечание: В таблице `crontab` *, показанной выше, установлены две переменные окружения -* `SHELL` *и* `PATH` *, и определены четыре задания. Первая из них указывает с помощью какой оболочки должны быть запущены задания* `cron` *, а вторая переопределяет пути доступа к исполняемым командам. При этом в первом, втором и третьем задании командой является* `run-parts` *, получающая в качестве аргумента имя каталога. Задание, определенное в последней строке* `crontab` *из примера выше, будет запускаться в восемь минут каждого часа, а предпоследнее задание - в 6 часов 52 минуты первого числа каждого месяца.*

Общесистемную таблицу заданий имеет право изменять лишь суперпользователь.

Вместо редактирования файла файла /etc/crontab принято создавать таблицы в виде отдельных файлов в каталоге /etc/cron.d.

Использование /etc/cron.d обычно удобнее, т. к. позволяет определить индивидуальные переменные для заданий в таких файлах.

Обычные пользователи могут иметь право устанавливать свои собственные таблицы заданий.

Таблицы заданий `cron` для обычных пользователей не имеют поля, в котором указывается имя пользователя, поэтому в них всего шесть полей.

Для манипулирования с заданиями crond существует специальная команда `crontab` . Если ей в качестве аргумента указан файл, содержащий таблицу заданий `crond` , то он будет считан, старая таблица `cron` будет стерта, а новая из файла будет сохранена.

Пример:

```
$ echo '30 21 * * * wall Hello!' > crontab.eg
$ crontab crontab.eg
```

Примечание: В этом примере был создан файл `crontab.eg` *, содержащий единственную строку с заданием, которое должно выполняться регулярно в 21:30. Эта таблица была считана и установлена командой* `crontab` .

Если вызвать команду `crontab` с опцией `-l` , то она выведет список регулярно выполняемых заданий пользователя:

Пример:

```
$ crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (crontab.eg installed on Wed Jan 14 21:51:49 2004)
# (Cron version -- $Id: crontab.c,v 1.15 2002/11/18 13:44:30 ldv
Exp $)
30 21 * * * wall Hello!
```

Суперпользователь может обращаться к таблицам пользователей, указывая их после опции `-u` команды `crontab` :

Пример:

```
# crontab -l -u user1
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (crontab.eg installed on Wed Jan 14 21:51:49 2004)
# (Cron version -- $Id: crontab.c,v 1.15 2002/11/18 13:44:30 ldv
Exp $)
30 21 * * * wall Hello!
```

Для редактирования текущей таблицы заданий в текстовом редакторе можно использовать команду `crontab -e` . При вызове этой команды будет запущен текстовый редактор по умолчанию, в нем будет открыта текущая таблица заданий `cron` , которую можно отредактировать. При выходе с сохранением будет установлена отредактированная версия таблицы заданий.

Крайне рекомендуется избегать подобной процедуры, поскольку ошибка при редактировании таблицы заданий может привести к ее порче и даже к потере.

Предлагается другой вариант редактирования:

Текущая таблица заданий пользователя сохраняется во временном файле для редактирования: `crontab -l > crontab.eg` .

Временный файл редактируется в текстовом редакторе: `vi crontab.eg` .

Отредактированная таблица заданий запоминается: `crontab crontab.eg` .

Если необходимо полностью удалить таблицу заданий, то необходимо выполнить команду `crontab -r` .

Пример:

```
$ crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
```

```
# (crontab.eg installed on Wed Jan 14 21:51:49 2004)
30 21 * * * wall Hello!
$ crontab -r
$ crontab -l
crontab: no crontab for user1
```

Можно ограничивать доступ пользователей к команде `crontab` с помощью файлов `/etc/cron.allow` и `/etc/cron.deny`. При этом:

Если существует файл `/etc/cron.allow`, то только пользователи, указанные в нем, имеют право использовать команду `crontab`.

Если этого файла нет, но имеется файл `/etc/cron.deny`, то пользователи, указанные в нем, не могут вызывать команду `crontab`.

Помимо создания таблиц с заданиями в /etc/crontab, вы можете создать скрипт с нужными вам командами и поместить этот сценарий в один из каталогов:

/etc/cron.hourly — задания выполняемые каждый час.

/etc/cron.daily — ежедневные задания.

/etc/cron.weekly — еженедельные задания.

/etc/cron.monthly — ежемесячные.

В дополнение к cron имеется еще одно средство выполнения регулярных заданий — anacron.

Anacron выполняет задания с заданной периодичностью, но при этом не предполагает что машина постоянно включена.

Конфигурация заданий находится в файле /etc/anacrontab.

Пример:

```
$ cat /etc/anacrontab
# /etc/anacrontab: configuration file for anacron
# See anacron(8) and anacrontab(5) for details.
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22
#period in days   delay in minutes   job-identifier   command
1   5 cron.daily          nice run-parts /etc/cron.daily
7   25    cron.weekly         nice run-parts /etc/cron.weekly
@monthly 45    cron.monthly        nice                  run-parts
/etc/cron.monthly
```

Задания в /etc/anacrontab описываются также в виде таблицы, где указывается:

Периодичность в днях.

Задержка в минутах.

Идентификатор заданиям.

Выполняемая команда.

Информация о выполненных заданиях записывается в каталоге /var/spool/anacron

Пример:

```
$ sudo cat /var/spool/anacron/cron.daily
20210211
```

6.3. Использование утилиты zeit

zeit – графическая утилита для планирования заданий cron.

К запланированным заданиям можно отнести такие операции, как резервное копирование, очистка дискового пространства и прочие задачи обслуживания системы.

Для установки утилиты zeit перейдите в сеанс пользователя root:

```
su -
```

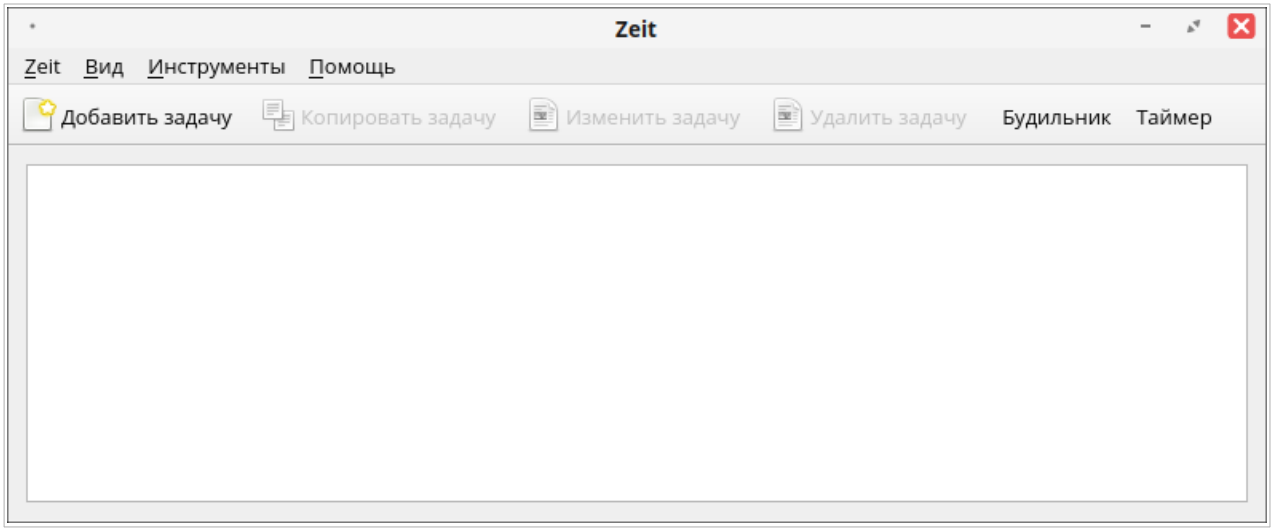
и выполните команду:

```
dnf install zeit
```

После установки запуск утилиты доступен из «Главного меню» – «Системные» – «Планировщик заданий» или через терминал (запуск производится с правами непривилегированного пользователя):

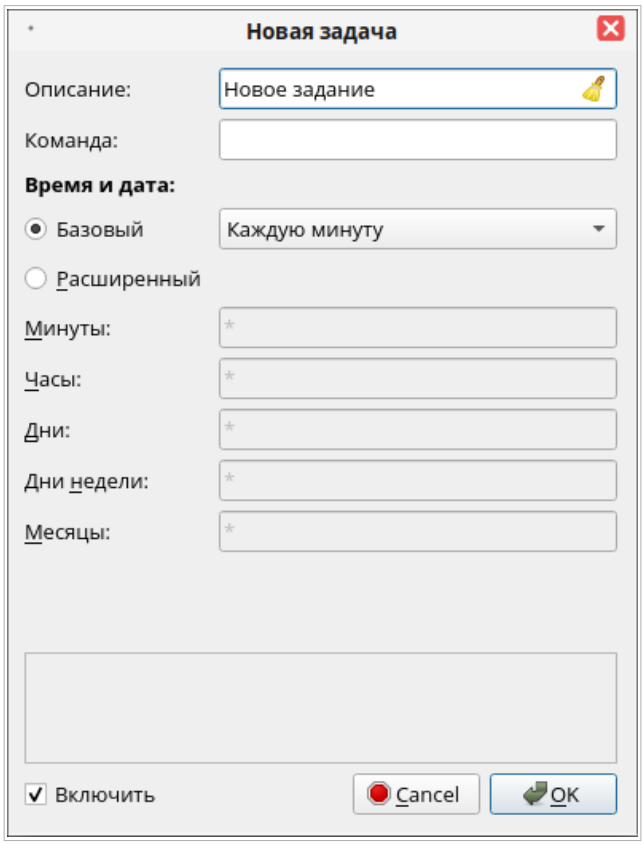
```
[ "$EUID" -ne 0 ] && zeit || echo "Команда не должна выполняться с правами пользователя root!"
```

Главное окно утилиты zeit выглядит следующим образом:



Интерфейс утилиты довольно прост и интуитивно понятен.

Для создания нового запланированного задания нажмите кнопку «Добавить задачу».

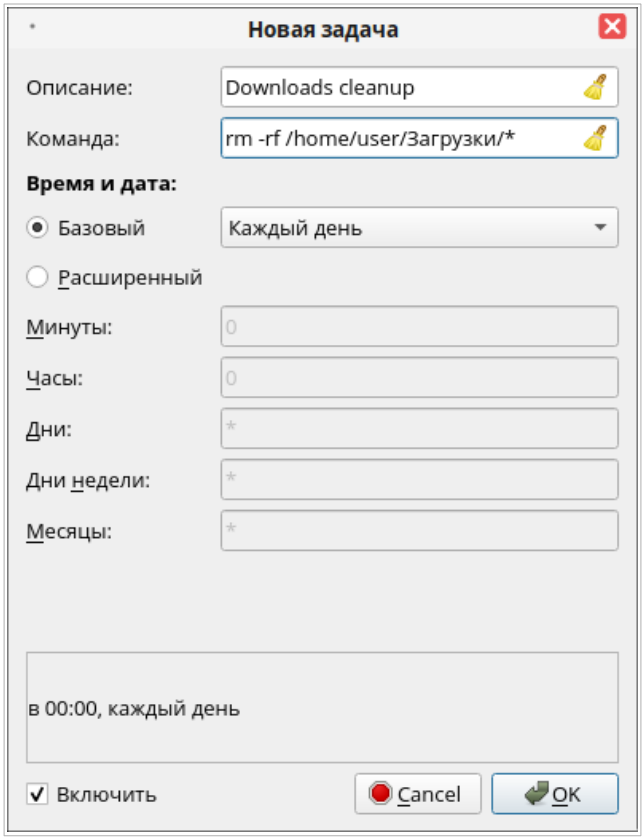


В появившемся окне задайте описание задания, непосредственно задание в виде команды bash, а также укажите временной интервал, по которому будет выполняться команда.

Если необходимо указать конкретную дату и время, выберите вариант Расширенный и установите собственные значения в соответствующих

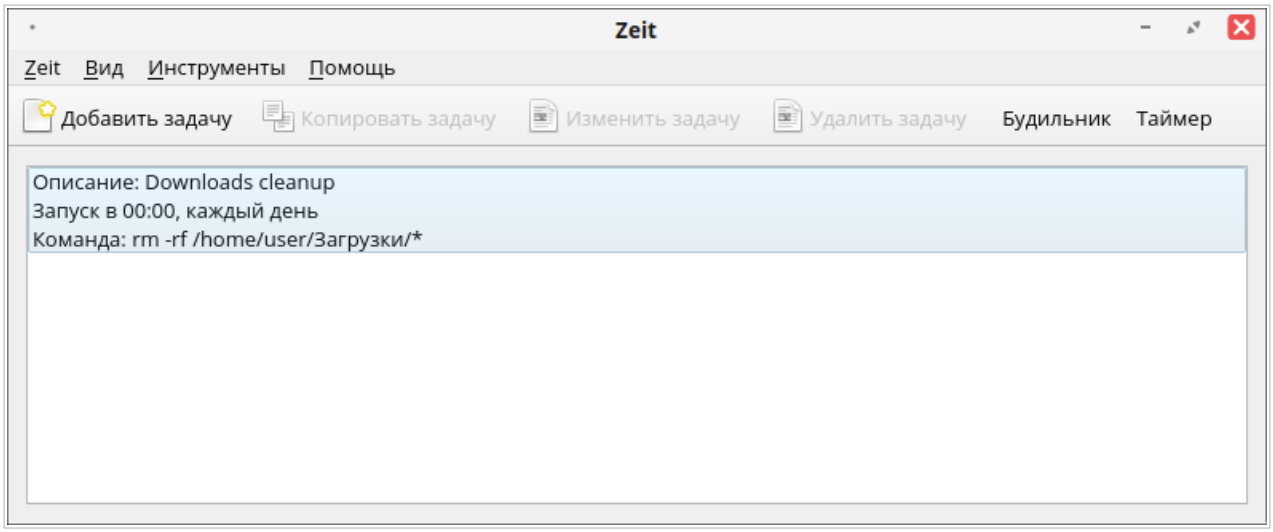
полях.

Для примера будет создана простая задача – ежедневная очистка каталога «Загрузки».



Нажмите ОК, чтобы создать задание cron.

После этого созданное задание будет отображено в списке.



Для проверки активности задания выведите содержимое файла crontab с помощью команды:

```
crontab -l
```

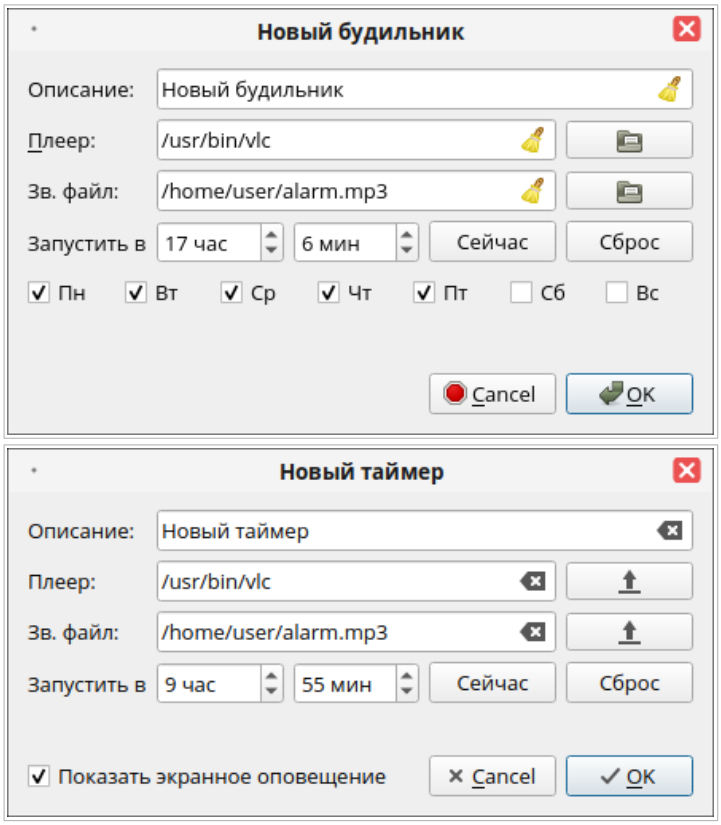
```
#Downloads cleanup
0 0 * * * rm -rf /home/user/Загрузки/*
```

Задание будет выполняться каждый день в 0:00.

Для редактирования запланированного задания выберите его из списка утилиты и нажмите кнопку «Изменить задачу». Внесите необходимые изменения и нажмите ОК для обновления задачи.

Для удаления запланированного задания выберите его из списка утилиты и нажмите кнопку «Удалить задачу».

В zeit также представлены две дополнительные опции: *будильник* и *таймер*. Будильник будет проигрывать звуковой сигнал в определенное время в выбранные дни недели, таймер будет проигрывать звуковой сигнал только в указанное время. Будильники/таймеры могут быть полезны для напоминания о чем-то важном.



6.4. Запуск заданий по расписанию с помощью systemd

Таймеры — файлы юнитов systemd, имя которых имеет суффикс .timer; они позволяют контролировать файлы .service или определенные события. Эти таймеры могут быть использованы в качестве замены cron. Таймеры имеют встроенную поддержку календарных и регулярных событий и могут

запускаться в асинхронном режиме.

Когда система РЕД ОС устанавливается на компьютер, создается несколько системных таймеров, являющихся частью процедур обслуживания системы.

Чтобы увидеть, какие таймеры работают на данный момент, введите команду:

```
systemctl status *timer

● dnf-makecache.timer - dnf makecache --timer
          Loaded: loaded (/usr/lib/systemd/system/dnf-makecache.timer;
enabled; vendo
      Active: active (waiting) since Thu 2022-06-23 10:03:35 MSK; 4h 37min
ago
     Trigger: Thu 2022-06-23 15:18:12 MSK; 37min left
    Triggers: ● dnf-makecache.service

 июн 23 10:03:35 localhost.localdomain systemd[1]: Started dnf makecache
--timer.

● sysstat-summary.timer - Generate summary of yesterday's process
accounting
          Loaded: loaded (/usr/lib/systemd/system/sysstat-summary.timer;
enabled; ven
      Active: active (waiting) since Thu 2022-06-23 10:03:35 MSK; 4h 37min
ago
     Trigger: Fri 2022-06-24 00:07:00 MSK; 9h left
    Triggers: ● sysstat-summary.service

 июн 23 10:03:35 localhost.localdomain systemd[1]: Started Generate
summary of ye

● systemd-tmpfiles-clean.timer - Daily Cleanup of Temporary Directories
             Loaded: loaded (/usr/lib/systemd/system/systemd-tmpfiles-
clean.timer; stati
      Active: active (waiting) since Thu 2022-06-23 10:03:35 MSK; 4h 37min
ago
     Trigger: Fri 2022-06-24 10:19:11 MSK; 19h left
    Triggers: ● systemd-tmpfiles-clean.service
        Docs: man:tmpfiles.d(5)
              man:systemd-tmpfiles(8)

 июн 23 10:03:35 localhost.localdomain systemd[1]: Started Daily Cleanup
of Tempo

● sysstat-collect.timer - Run system activity accounting tool every 10
minutes
          Loaded: loaded (/usr/lib/systemd/system/sysstat-collect.timer;
```

```
enabled; ven
     Active: active (waiting) since Thu 2022-06-23 10:03:35 MSK; 4h 37min
ago
    Trigger: Thu 2022-06-23 14:50:00 MSK; 9min left
   Triggers: ● sysstat-collect.service

 июн 23 10:03:35 localhost.localdomain systemd[1]: Started Run system
activity ac
```

С каждым таймером связано, по меньшей мере, шесть строк, содержащих сведения о нём:

1.Имя файла таймера и короткое описание цели существования этого таймера (но не всегда).

2.Сведения о состоянии таймера - загружен ли таймер, полный путь к файлу таймера, состояние vendor preset (disabled или enabled).

3.Сведения об активности таймера, куда входят данные о том, когда именно таймер был активирован.

4.Дата и время следующего запуска таймера и примерное время, оставшееся до его запуска.

5.Имя сервиса или события, вызываемого таймером.

6.Некоторые (но не все) таймеры содержат указатели на документацию.

Примеры использования таймеров:

Самым простым примером использования таймеров можно считать системный таймер systemd-tmpfiles-clean. Как следует из названия, данный таймер раз в некоторое время производит очистку временных файлов системы.

```
● systemd-tmpfiles-clean.timer - Daily Cleanup of Temporary Directories
             Loaded: loaded (/usr/lib/systemd/system/systemd-tmpfiles-
clean.timer; stati
     Active: active (waiting) since Thu 2022-06-23 10:03:35 MSK; 4h 37min
ago
    Trigger: Fri 2022-06-24 10:19:11 MSK; 19h left
   Triggers: ● systemd-tmpfiles-clean.service
       Docs: man:tmpfiles.d(5)
             man:systemd-tmpfiles(8)
```

Данный таймер является системным и включен по умолчанию.

В системе РЕД ОС юниты systemd располагаются в следующих директориях:

•/usr/lib/systemd/system – юниты, поставляемые вместе с системой и

устанавливаемыми приложениями;

- /etc/systemd/system – юниты системного администратора.

Для примера будет создан простой файл конфигурации сервиса, запускающий команду free (утилита, выводящая информацию об использовании оперативной памяти). Это может пригодиться, если необходимо регулярно отслеживать объём свободной памяти.

Создайте unit-файл с именем ramMonitor.service в папке /etc/systemd/system.

```
nano /etc/systemd/system/ramMonitor.service
```

И добавьте в него следующие строки:

```
[Unit]
Description=Monitors RAM # Описание сервиса
Wants=ramMonitor.timer # Зависимость, отсылается к таймеру (будет создан позже)

[Service]
Type=oneshot # Тип службы, означающий выполнение одного задания и завершение
ExecStart=/usr/bin/free # Указывается полный путь к исполняемому файлу программы

[Install]
WantedBy=multi-user.target # Указывается, на каком уровне происходит запуск сервиса. Параметр multi-user.target указывает на запуск в многопользовательском режиме без графики
```

Данный файл можно назвать простейшим файлом конфигурации сервиса.

После проведенных манипуляций, проверьте состояние вашего сервиса командой:

```
systemctl status ramMonitor.service

● ramMonitor.service - Monitors RAM
       Loaded: loaded (/etc/systemd/system/ramMonitor.service; disabled; vendor pr
     Active: inactive (dead)
```

Сервис существует, но не активен. Запустите сервис и проверьте его статус:

```
systemctl start ramMonitor.service
systemctl status ramMonitor.service

● ramMonitor.service - Monitors RAM
```

```
         Loaded: loaded (/etc/systemd/system/ramMonitor.service; disabled;
vendor pr
        Active: inactive (dead)

  июн  23  15:08:44  localhost.localdomain  systemd[1]:  Starting  Monitors
RAM...
  июн  23  15:08:44  localhost.localdomain  systemd[1]:  ramMonitor.service:
Succeeded.
 июн 23 15:08:44 localhost.localdomain systemd[1]: Finished Monitors RAM.
 июн  23  15:08:44  localhost.localdomain  free[3914]:                  total
use
 июн  23  15:08:44  localhost.localdomain  free[3914]:  Mem:           995784
53659
 июн  23  15:08:44  localhost.localdomain  free[3914]:  Swap:         2097148
927
```

При включении сервиса в терминале не выводится никаких сообщений. Это происходит, потому что по умолчанию стандартный вывод (stdout) от программ, запускаемых systemd, перенаправляется в журнал systemd.

Чтобы проверить этот журнал, воспользуйтесь командой:

```
 journalctl -u ramMonitor.service

  -- Logs begin at Thu 2022-04-21 11:49:59 MSK, end at Thu 2022-06-23
15:12:30 MSK
  июн  23  15:08:44  localhost.localdomain  systemd[1]:  Starting  Monitors
RAM...
  июн  23  15:08:44  localhost.localdomain  systemd[1]:  ramMonitor.service:
Succeeded.
 июн 23 15:08:44 localhost.localdomain systemd[1]: Finished Monitors RAM.
 июн  23  15:08:44  localhost.localdomain  free[3914]:                  total
use
 июн  23  15:08:44  localhost.localdomain  free[3914]:  Mem:           995784
53659
 июн  23  15:08:44  localhost.localdomain  free[3914]:  Swap:         2097148
927
```

С помощью ключа -S вы можете указать временной период, за который утилита journalctl будет искать записи. Например, journalctl -S today -u ramMonitor.service выведет записи, сделанные за сегодня.

Убедившись в работоспособности сервиса, создайте для него таймер. Для этого необходимо создать в той же папке файл с названием ramMonitor.timer.

```
nano /etc/systemd/system/ramMonitor.timer
```

В файл добавьте следующее содержимое:

```
[Unit]
Description=Monitors RAM
Requires=ramMonitor.service # Указание строгой зависимости

[Timer]
Unit=ramMonitor.service # Ссылка на сервис
OnCalendar=*-*-*  *:*:00  # Срабатывание таймера по условию
календаря. В данном случае – каждую минуту

[Install]
WantedBy=timers.target
```

В секции [Timer] указываются условия запуска. Таймеры могут быть двух типов: событийные и монотонные. Первые активируются по событию, вторые выполняются периодически. Из событий таймеров можно выделить OnBootSec, срабатывающий через определенное время после запуска системы. Из монотонных следует выделить:

•OnUnitActiveSec - срабатывает через указанное время после активации целевого юнита;

•OnUnitInactiveSec — срабатывает так же, как OnUnitActiveSec, только время отсчитывается с момента прекращения работы целевого юнита, хорошо подходит для «длительных» задач, например, бекапов.

•OnCalendar - срабатывает по условию календаря.

В качестве формата даты для календаря используется формат:

`DOW YYYY-MM-DD HH:MM:SS`

(где DOW – Day Of Week – *день недели*, необязательный параметр; за ним следует указание *года*, *месяца*, *дня* через дефис и *часы*, *минуты* и *секунды* через двоеточие. Для указания любого значения используется «*», перечисления делаются через запятую, а диапазоны через «..»).

После завершения настройки нужно запустить таймер и проверить его статус:

```
 systemctl start ramMonitor.timer
 systemctl status ramMonitor.timer

 ● ramMonitor.timer - Monitors RAM
         Loaded: loaded (/etc/systemd/system/ramMonitor.timer; disabled;
vendor pres
      Active: active (waiting) since Thu 2022-06-23 15:16:22 MSK; 7s ago
     Trigger: Thu 2022-06-23 15:17:00 MSK; 29s left
    Triggers: ● ramMonitor.service
июн 23 15:16:22 localhost.localdomain systemd[1]: Started Monitors
```

`RAM`.

Если все настроено верно, таймер будет иметь состояние active (waiting), а ниже будет указано время до его запуска.

Глава 7. Модули ядра и настройки ядра Linux

7.1. Работа с модулями ядра.

Современные ядра Linux построены в соответствии с модульной архитектурой, что о позволяет изменять функциональность ядра в работающей системе, а также включать в ядро и удалять драйверы устройств “на лету”.

Модули ядра представляют собой объектные файлы с кодом, который можно подключать к работающему ядру Linux.

Они образуются при сборке ядра (она будет рассмотрена в следующей главе).

В ядрах стабильной ветви версии до 2.6.0 файлы модулей ядра имеют суффикс `.o`

В ядрах серии 2.6 и выше - `.ko` (kernel object).

Модули располагаются в каталоге `/lib/modules` :

Пример:

```
$ ls -F /lib/modules/
2.4.22-gentoo-r5/
```

Примечание: Обратите внимание, что в каталоге `/lib/modules` *имеется подкаталог (или подкаталоги) с именем ядра. Это связано с тем, что может иметься несколько различных ядер для различных целей. В данном случае имеется единственный каталог с модулями ядра* `2.4.22-gentoo-r5`.

Имена подкаталогов `/lib/modules` всегда точно соответствуют именам ядер, имеющихся в системе.

Команда `uname -r` позволяет узнать имя ядра, запущенного в данный момент.

Пример:

```
$ uname -r
2.4.22-gentoo-r5
```

В каталоге `/lib/modules/<kernelname>` могут находиться не только модули ядра, но и драйверы для устройств, разработанные независимыми от команды разработчиков ядра Linux поставщиками.

Пример:

```
$ ls -F /lib/modules/`uname -r`
build@    modules.dep                              modules.isapnpmap
modules.pnpbiosmap
kernel/ modules.generic_string  modules.parportmap  modules.usbmap
misc/   modules.ieee1394map     modules.pcimap      pcmcia/
nvidia/
```

Примечание: В этом примере показано содержимое подкаталога `/lib/modules`*, имя которого задается с помощью командной подстановки и соответствует имени текущего ядра. Каталог* `nvidia` *содержит драйвер для видеокарты NVidia, полученный от производителя видеокарты.*

В зависимости от ветви используемого ядра, а также от обновлений (patches), наложенных на ядро, раскладка каталогов, приведенных ниже может существенно изменяться.

Основная часть модулей ядра, среди которых, в частности, драйверы устройств, находится в подкаталоге `kernel` .

Пример:

```
$ ls -F /lib/modules/`uname -r`/kernel
arch/  crypto/  drivers/  fs/  lib/  net/
```

В подкаталоге `fs` находятся драйверы, модули расширения функциональности и иное программное обеспечение для различных файловых систем.

Пример:

```
$ ls -F /lib/modules/`uname -r`/kernel/fs
affs/             coda/     freevxfs/   intermezzo/   lockd/   nfsd/
smbfs/
befs/               cramfs/   hfs/          jbd/             ntfs/
supermount/ vfat/
bfs/             efs/      hfsplus/    jffs/         msdos/   sysv/
xfs/
binfmt_aout.o  ext3/    hpfs/      jffs2/       ncpfs/  quota_v2.o
udf/
binfmt_misc.o   fat/      ibu/        jfs/          nfs/     romfs/
ufs/
```

Подкаталог `net` содержит программное обеспечение для поддержки сетевой инфраструктуры.

Пример:

```
$ ls -F /lib/modules/`uname -r`/kernel/net
8021q/      ax25/       core/    ipsec/  ipx/     lapb/    sched/
x25/
appletalk/  bluetooth/  decnet/  ipv4/   irda/    netrom/  sunrpc/
atm/            bridge/        econet/   ipv6/     khttpd/   rose/
wanrouter/
```

В подкаталоге `drivers` находятся драйверы для различных устройств.

Пример:

```
$ ls -F /lib/modules/`uname -r`/kernel/drivers
acpi/     bluetooth/   i2c/           input/   message/   parport/
sensors/    usb/
atm/    char/       ide/       isdn/   mtd/      pcmcia/   sound/
video/
block/    hotplug/       ieee1394/    media/    net/           scsi/
telephony/
```

Для получения списка модулей, установленных в настоящее время в ядро, следует выполнить команду `/sbin/lsmod` .

Пример:

```
$ /sbin/lsmod
Module                  Size  Used by    Not tainted
sd_mod                 12556   0 (autoclean) (unused)
sr_mod                 15128   0 (autoclean) (unused)
i830                   70112  13
agpgart                42168  12 (autoclean)
usb-storage           118192   0 (unused)
hw_random               2968   0 (unused)
i810_rng                2784   0 (unused)
i810_audio             26184   0
ac97_codec             13224   0 [i810_audio]
soundcore               3972   2 [i810_audio]
ohci1394               32328   0 (unused)
ieee1394              187300   0 [ohci1394]
eepro100               20468   1
mii                     2528   0 [eepro100]
ide-scsi               10128   0
scsi_mod                105876    4 [sd_mod sr_mod usb-storage ide-
scsi]
ds                      7240   1
yenta_socket           10752   1
pcmcia_core            45536   0 [ds yenta_socket]
```

```
keybdev                  1920   0 (unused)
mousedev                 4212   1
hid                     20996   0 (unused)
input                    3552   0 [keybdev mousedev hid]
uhci                    27152   0 (unused)
ehci-hcd                17992   0 (unused)
usbcore                 63904   1 [usb-storage hid uhci ehci-hcd]
```

Примечание: По сути программа `/sbin/lsmod` *просто выводит в форматированном виде информацию из файла* `/proc/modules`.

Пример:

```
$ cat /proc/modules
ext3                    97416   1 (autoclean)
jbd                     43312   1 (autoclean) [ext3]
sd_mod                  12556   2 (autoclean)
sr_mod                  15128   0 (autoclean) (unused)
i830                    70112  13
agpgart                 42168  12 (autoclean)
usb-storage            118192   1
hw_random                2968   0 (unused)
i810_rng                 2784   0 (unused)
i810_audio              26184   0
ac97_codec              13224   0 [i810_audio]
soundcore                3972   2 [i810_audio]
ohci1394                32328   0 (unused)
ieee1394               187300   0 [ohci1394]
eepro100                20468   1
mii                      2528   0 [eepro100]
ide-scsi                10128   0
scsi_mod                 105876    4 [sd_mod sr_mod usb-storage ide-
scsi]
ds                       7240   1
yenta_socket            10752   1
pcmcia_core             45536   0 [ds yenta_socket]
keybdev                  1920   0 (unused)
mousedev                 4212   1
hid                     20996   0 (unused)
input                    3552   0 [keybdev mousedev hid]
uhci                    27152   0 (unused)
ehci-hcd                17992   0 (unused)
usbcore                 63904   1 [usb-storage hid uhci ehci-hcd]
```

Для отображения доступной информации о модуле используйте команду `/sbin/modinfo`.

Пример:

```
$ /sbin/modinfo uhci
filename:
/lib/modules/2.4.22-gentoo-r5/kernel/drivers/usb/host/uhci.o
description: "USB Universal Host Controller Interface driver"
author:         "Linus 'Frodo Rabbit' Torvalds, Johannes Erdfelt,
Randy Dunlap, Georg Acher, Deti Fliegl, Thomas Sailer, Roman
Weissgaerber"
license:     "GPL"
parm:        debug int, description "Debug level"
```

Примечание: В этом примере команда `modinfo` *выдала информацию о модуле* `uhci`*, являющимся универсальным драйвером для USB хост контроллера.*

Информация о модулях включается в сами модули, хотя она при необходимости может быть скрыта.

Эта информация стандартизирована и представлена следующими полями:

`author` - автор модуля;

`description` – описание модуля;

`license` – лицензия;

`param` – дополнительные параметры;

`depend` – зависимости модуля от других модулей;

`alias` – псевдоним для модуля.

При использовании ядер серии 2.6 для отображения информации лишь по одному или нескольким полям следует использовать опцию `-F` команды `modinfo` с указанием требуемых полей.

При использовании более старой версии `modinfo` можно указывать опции:

`-a` – автор;

`-d` – описание;

`-l` – лицензия;

`-n` – имя файла модуля;

`-p` – параметры.

Новые модули могут быть загружены в работающее ядро Linux с помощью команды `/sbin/insmod`.

Установить модуль в ядро может только суперпользователь. В качестве аргумента для команды `insmod` нужно указывать имя файла модуля.

Для удаления модуля из ядра предназначена команда `/sbin/rmmod` .

Пример:

```
# rmmod -v parport
Checking parport for persistent data
# lsmod
Module                  Size  Used by    Not tainted
ext3                   97416   1  (autoclean)
jbd                    43312   1  (autoclean) [ext3]
```

Примечание: В этом примере команда `rmmod` *удалила драйвер параллельного порта, что демонстрирует сокращенный листинг, выведенный командой* `lsmod`*. Опция* `-v` *команды* `rmmod` *была использована для переключения ее в режим работы с повышенной информативностью вывода.*

Часто модули ядра для работоспособности требуют наличия других модулей ядра.

В каталоге, содержащем модули ядра (`/lib/modules/`uname -r``), можно обнаружить текстовый файл `modules.dep`. Он содержит имена файлов модулей и через двоеточие имена файлов модулей, от которых первые зависят.

Файл `modules.dep` генерируется с помощью утилиты `/sbin/depmod`.

Вместо использования команд `insmod` и `rmmod` предпочтительнее использовать команду `modprobe`, обеспечивающую более интеллектуальное поведение.

Эта команда использует файл `modules.dep` для автоматического учета зависимостей модулей, что позволяет загружать модули ядра не задумываясь об их зависимостях. Они будут учтены и требуемые модули будут загружены.

По умолчанию команда `modprobe` устанавливает модуль в ядро, но будучи вызванной с опцией `-r` она способна удалять модули.

Пример:

```
ifconfig eth0 down
modprobe -r eepro100
modprobe e100
ifconfig eth0 up
```

Примечание: В этом примере первая команда отключила сетевой интерфейс `eth0`*. Далее был выгружен модуль ядра – драйвер сетевого адаптера Intel EtherExpress* `eepro100`*. При этом был автоматически выгружен модуль ядра, требуемый для работы модуля* `eepro100`*. После этого был загружен другой модуль с альтернативным драйвером для этого же сетевого адаптера. Последняя команда вновь активизирует сетевой интерфейс.*

Использование опции `-a` позволяет установить сразу несколько модулей, имена которых заданы в качестве аргумента.

Если модуль требует передачи ядру некоторой информации, то ее можно указывать в качестве аргументов команды `modprobe`.

Пример: для установки драйвера сетевой платы ISA NE2000 требуется указать ее IRQ и IO/Base:

```
# modprobe ne irq=5 io=0x320
```

Опция `-c` позволяет получить полный список настроек команды `modprobe`.

Он определяется специальными конфигурационными файлами в каталоге `/etc/modprobe.d`

Примечание: В разных дистрибутивах GNU/Linux местоположение файлов конфигурации может отличаться.

Одним из главных предназначений файла конфигурации в `/etc/modprobe.d` является обеспечение передачи параметров, требуемых для модулей.

Помимо этого в данном файле можно задавать команды, которые надо выполнить до или после загрузки или удаления модулей.

Также в этих файлах могут задаваться псевдонимы для модулей, делающие работу с ними более удобной.

В каталоге /etc/modules-load.d/ можно создать файлы со списком модулей, которые должны быть загружены при старте системы.

Обычно в этих файлах прописываются модули драйверов для тех устройств, которые не поддерживают PnP.

Управлять модулями ядра можно при помощи утилиты sysctl.

Для начала работы нам необходимо вывести на экран все доступные опции, делается с помощью команды:

```
sysctl -a
```

Просмотреть все параметры *sysctl* может любой пользователь Linux (в нашем примере показываем на Ubuntu 20), а вот изменять параметры ядра может только пользователь с правами root.

Также можно посмотреть значение одного параметра, который вас интересует в конкретный момент, например*:

```
sysctl net.ipv4.ip_forward
```

Примечание: данная команда разрешает или запрещает маршрутизацию пакетов.

Надо отметить, что если значение параметра равно 0, то функция запрещена, а если равно 1, то разрешена.

Для вывода списка сразу всех переменных, которые начинаются, к примеру с vm, наберите в терминале команду:

```
sysctl vm
```

Ниже разъясним основные опции команды sysctl:

man sysctl — выводит подробную документацию по данной команде;

sysctl -n — выведет на экран только значения переменных;

sysctl -N — покажет вам только названия переменных, без их значений;

sysctl -A — показывает все доступные настройки ядра, но в табличной форме;

sysctl -w "переменная"="значение" — используется для изменения значения переменной;

sysctl -p — выводит на экран настройки из файла /etc/sysctl.conf (или другого указанного файла настроек).

Покажем на практике, как использовать данные команды в своей работе.

Например, попробуем разрешить маршрутизацию пакетов, выполнив команду:

```
sysctl -w net.ipv4.ip_forward=1
```

Если мы просто применим команду без опции -w, то сделанные изменения не сохранятся после перезагрузки ОС:

```
sysctl net.ipv4.ip_forward=1
```

Как правило, все показанные нами переменные параметров ядра Линукс, хранятся в конфигурационном файле /etc/sysctl.conf.

Однако, некоторые переменные могут быть записаны в следующих каталогах и файлах:

```
/run/sysctl.d/*.conf
/etc/sysctl.d/*.conf
/usr/local/lib/sysctl.d/*.conf
/usr/lib/sysctl.d/*.conf
/lib/sysctl.d/*.conf
```

Для того, чтобы вернуть настройки ядра к состоянию «до сделанных вами изменений», необходимо выполнить команду:

```
sysctl --system
```

Для начала, можно поэкспериментировать с настройками ядра в «ручном режиме» с помощью выполнения данных команд в терминале. Если сделанные вами настройки — «рабочие» и помогают в оптимизации работы ядра вашей ОС, то можете их внести на постоянной основе в файл /etc/sysctl.conf.

Существует еще один интересный способ для настройки параметров ядра Linux — это работа с директорией /proc/sys. Здесь мы используем команду echo для изменения значений параметров настроек ядра Линукс, а для вывода на экран этих значений используем cat.

Ниже покажем на простых примерах, как это можно сделать на практике (по аналогии с командами sysctl). Итак, выведем на экран значение параметра net.ipv4.ip_forward данным способом (на скриншоте ниже показано, как вывести значение данного параметра двумя способами):

```
cat /proc/sys/net/ipv4/ip_forward
```

Для изменения значения данного параметра с 0 на 1, выполним команду*:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Изменения, сделанные таким способом, будут работать до первой перезагрузки ОС.

Приведем некоторые наиболее важные и часто используемые переменные ядра Линукс и расскажем, что они означают и как их настроить:

fs.file-max — показывает максимальное значение для дескрипторов файлов, создаваемых и обрабатываемых ядром Линукс. "По умолчанию" устанавливается число, равное 10% оперативной памяти компьютера.

fs.aio-nr — обозначает заданное число операций ввода и вывода для файловой системы.

cad_pid — указывает идентификационный номер процесса (PID), который присваивается сигналу, при нажатии Ctrl+Alt+Del.

kernel.ctrl-alt-del — параметр, отвечающий за правильную перезагрузку по комбинации клавиш Ctrl+Alt+Del (связан с параметром cad_pid). При значении 0, ОС передает сигнал процессу, назначенному в cad_pid, если значение параметра больше 0, то ОС выполняет перезагрузку немедленно.

kernel.hostname — параметр позволяет произвести изменение имени

вашего компьютера «на лету» (без процесса перезагрузки).

kernel.modules_disabled — выключает загрузку модулей ядра.

kernel.panic — выставлено количество секунд, оставшихся до начала процесса перезагрузки (после случая ошибки в ядре).

kernel.randomize_va_space — важный параметр, обеспечивающий безопасность вашей ОС, (служит против атак на переполнение буфера), включен «по умолчанию»

kernel.sysrq — включает все функции (значение 1) или отключает (значение 0) для управления ядром при помощи SysRQ.

abi.vsyscall32 — параметр, разрешающий (при его значении 1) исполнение 32-разрядных программ в 64-разрядной ОС.

debug.exception-trace — служит для вывода значений регистров процессора, стека вызова процедур в случае ошибки в ядре. Установлен во включенном «по умолчанию» состоянии.

dev.cdrom.autoclose — параметр предназначен для управления CDROM, при попытке его монтирования, может его автоматически закрыть (отключен «по умолчанию»).

net.ipv4.ip_forward — данная настройка применяется для роутеров, она разрешает или запрещает маршрутизацию пакетов.

net.ipv4.ip_local_port_range — определяет диапазон локальных портов, разрешенных для использования вашими программами.

net.ipv4.tcp_rmem — указывает объем памяти, которая выделена для работы TCP.

vm.laptop_mode — данный параметр будет полезен для работы ноутбуков, в частности, для экономии заряда батареи. Эта настройка позволяет хранить данные какое-то время в оперативной памяти компьютера (а не сразу записывать их на винчестер).

vm.swappiness — этот параметр служит для установки % свободной памяти, при достижении этого значения, данные будут автоматически переводиться в swap. «По умолчанию» в Ubuntu установлено значение этого параметра, равное 60. Т.е. ОС будет переносить в раздел swap такие разделы памяти, которые были не использованы в течении длительного промежутка времени.

7.2. UDEV и каталоги /dev и /sys

Для работы с устройствами в Linux имеются два каталога:

/dev – каталог, в котором по умолчанию создаются файлы устройств. Эти файлы устройств позволяют нам «общаться» с устройствами посредством стандартных системных вызовов для работы с файлами: открыть файл open(), считать read() или записать write() данные.

/sys – виртуальная файловая системы, которая представляет структуры ядра для работы с устройствами в виде файлов.

В первых ОС на основе ядер Linux каталога /sys не было, а в каталоге /dev файлы устройств создавались вручную или во время инсталляции системы. В результате имелись файлы устройств, а самих устройств не было. Другой проблемой было отсутствие поддержки горячего подключения/отключения (hotplug) устройств.

Для решения проблемы с горячим подключениям и «лишними» файлами устройств изначально использовались такие утилиты как devfs, hotplug и HAL

Начиная с ядер версии 2.5 был представлен новый механизм — udev.

В апреле 2012 исходный код udev слился с исходным кодом systemd

udev — работающая в пространстве пользователя система, с помощью которой системный администратор может создавать обработчики событий.

События, получаемые udev, обычно генерируются ядром Linux в ответ на физические события, происходящие с периферийными устройствами. Например, при обнаружении периферийных устройств или "горячем" подключении udev может выполнить определённые действия, в том числе и вернуть управление ядру, если необходима загрузка модулей или прошивок.

Подобно предшественникам, утилитам devfsd и hotplug, udev управляет файлами устройств в каталоге /dev, добавляя их, переименовывая и создавая символические ссылки. udev полностью замещает функционал hotplug и hwdetect.

Благодаря udev в каталоге /dev находятся файлы только тех устройств, которые в настоящий момент подключены к системе. Каждое устройство имеет свой соответствующий файл. Если устройство отключается от системы, то данный файл удаляется.

Содержимое каталога /dev хранится на виртуальной файловой системе, и все файлы, находящиеся в нём, создаются при каждом запуске системы.

Обработка событий в udev происходит параллельно, что теоретически улучшает производительность старых систем. С другой стороны, это может усложнить администрирование. Так, при перезапуске системы порядок загрузки модулей ядра может измениться, а при наличии в машине нескольких

блочных устройств могут поменяться названия их файлов. Например, для системы с двумя жёсткими дисками файл /dev/sda после перезагрузки может превратиться в /dev/sdb

udev входит в состав systemd и установлен по умолчанию. Подробнее см. systemd-udevd.service(8).

Существует также отдельный от systemd форк, который можно установить с пакетом eudev или eudev-git

Для настройки udev создаются специальные правила.

Правила должны располагаться в каталоге /etc/udev/rules.d/ и иметь названия с суффиксом .rules

```
$ cat /etc/udev/rules.d/70-persistent-ipoib.rules
# This is a sample udev rules file that demonstrates how to get
udev to
# set the name of IPoIB interfaces to whatever you wish.  There is
a
# 16 character limit on network device names.
#
# Important items to note: ATTR{type}=="32" is IPoIB interfaces,
and the
# ATTR{address} match must start with ?* and only reference the
last 8
# bytes of the address or else the address might not match the
variable QPN
# portion.
#
# Modern udev is case sensitive and all addresses need to be in
lower case.
#
#       ACTION=="add",       SUBSYSTEM=="net",       DRIVERS=="?*",
ATTR{type}=="32",       ATTR{address}=="?*00:02:c9:03:00:31:78:f2",
NAME="mlx4_ib3"
```

Утилита udevadm предназначена для мониторинга и управления udev.

Пример: создадим специальные правила для подключения USB Flash накопителей определенной модели. Во-первых понаблюдаем за процессом подключения USB Flash:

```
# udevadm monitor
monitor will print the received events for:
UDEV - the event which udev sends out after rule processing
KERNEL - the kernel uevent
KERNEL[158.634005]                                              add
/devices/pci0000:00/0000:00:0b.0/usb1/1-1 (usb)
```

```
KERNEL[158.637827]                                          add
/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0 (usb)
...
UDEV                                [161.309815]            add
/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:
0/3:0:0:0/block/sdb (block)
UDEV                                [162.368973]            add
/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:
0/3:0:0:0/block/sdb/sdb3 (block)
UDEV                                [162.519992]            add
/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:
0/3:0:0:0/block/sdb/sdb2 (block)
UDEV                                [162.574735]            add
/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:
0/3:0:0:0/block/sdb/sdb1 (block)
```

Посмотрим характеристики устройства. Обратите внимание, что к тем путям, что выводит udevadm нужно просто в начале дописать /sys:

```
#                                                            ls
/sys/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target
3:0:0/3:0:0:0/block/sdb
alignment_offset   discard_alignment   hidden      power        sdb1
stat
bdi                 events              holders    queue        sdb2
subsystem
capability          events_async        inflight   range        sdb3
trace
dev                  events_poll_msecs  integrity  removable    size
uevent
device             ext_range          mq         ro           slaves
# ls /sys/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0
authorized          bInterfaceProtocol  ep_01     power
bAlternateSetting   bInterfaceSubClass  ep_81     subsystem
bInterfaceClass            bNumEndpoints                    host3
supports_autosuspend
bInterfaceNumber    driver              modalias  uevent
#  ls  /sys/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/driver
-l
lrwxrwxrwx.     1     root     root     0     Feb          6     12:28
/sys//devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/driver     -
> ../../../../../../bus/usb/drivers/usb-storage
```

Или более простой путь к устройству:

```
# ls  /sys/block/sdb/
alignment_offset   discard_alignment   hidden      power        sdb1
stat
bdi                 events              holders    queue        sdb2
```

```
subsystem
capability          events_async         inflight    range       sdb3
trace
dev                  events_poll_msecs   integrity   removable   size
uevent
device             ext_range           mq          ro          slaves
# cat /sys/block/sdb/size
15769600
# cat  /sys/block/sdb/uevent
MAJOR=8
MINOR=16
DEVNAME=sdb
DEVTYPE=disk
# cat /sys/block/sdb/removable
1
# cat /sys/block/sda/removable
0
```

Теперь мы с помощью udevadm определим производителя устройства:

```
# # udevadm info /dev/sdb | egrep 'VENDOR|SUBSYS'
E: ID_VENDOR=Generic
E: ID_VENDOR_ENC=Generic\x20
E: ID_VENDOR_ID=cd12
E: SCSI_VENDOR=Generic
E: SCSI_VENDOR_ENC=Generic\x20
E: SUBSYSTEM=block
```

Или

```
# # udevadm info -a /dev/sdb | egrep '/devices/|Vendor|SUBSYSTEM'
   looking at device '/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-
1:1.0/host3/target3:0:0/3:0:0:0/block/sdb':
    SUBSYSTEM=="block"
                  looking         at         parent         device
'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0
:0/3:0:0:0':
    SUBSYSTEMS=="scsi"
                  looking         at         parent         device
'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0
:0':
    SUBSYSTEMS=="scsi"
                  looking         at         parent         device
'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3':
    SUBSYSTEMS=="scsi"
                  looking         at         parent         device
'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0':
    SUBSYSTEMS=="usb"
                  looking         at         parent         device
'/devices/pci0000:00/0000:00:0b.0/usb1/1-1':
```

```
    SUBSYSTEMS=="usb"
    ATTRS{idVendor}=="cd12"
                looking         at         parent         device
'/devices/pci0000:00/0000:00:0b.0/usb1':
    SUBSYSTEMS=="usb"
    ATTRS{idVendor}=="1d6b"
  looking at parent device '/devices/pci0000:00/0000:00:0b.0':
    SUBSYSTEMS=="pci"
  looking at parent device '/devices/pci0000:00':
    SUBSYSTEMS==""
```

udevadm info значения префиксов

```
Prefix    Meaning
"P:"    Device path in /sys/
"M:"    Device name in /sys/
        (i.e. the last component
        of "P:")
"R:"    Device number in /sys/
        (i.e. the numeric suffix
        of the last component of
        "P:")
"U:"    Kernel subsystem
"T:"    Kernel device type
        within subsystem
"D:"    Kernel device node
        major/minor
"I:"    Network interface index
"N:"    Kernel device node name
"L:"    Device node symlink
        priority
"S:"    Device node symlink
"Q:"    Block device sequence
        number (DISKSEQ)
"V:"    Attached driver
"E:"    Device property
```

Создаем правило и проверяем его:

```
# cat /etc/udev/rules.d/10-myflash.rules
ACTION=="add",     SUBSYSTEM=="block",     ATTRS{idVendor}=="cd12",
RUN+="/usr/bin/chgrp myflash /dev/%k", SYMLINK+="myflsh%n"
```

Не забудьте создать группу:

```
# groupadd myflash
# gpasswd -a admuser myflash
# ls -l /dev/sdb*
```

```
brw-rw----. 1 root myflash 8, 16 Feb  6 14:56 /dev/sdb
brw-rw----. 1 root myflash 8, 17 Feb  6 14:56 /dev/sdb1
brw-rw----. 1 root myflash 8, 18 Feb  6 14:56 /dev/sdb2
brw-rw----. 1 root myflash 8, 19 Feb  6 14:56 /dev/sdb3
# ls -l /dev/myflsh*
lrwxrwxrwx. 1 root root 3 Feb  6 14:56 /dev/myflsh -> sdb
lrwxrwxrwx. 1 root root 4 Feb  6 14:56 /dev/myflsh1 -> sdb1
lrwxrwxrwx. 1 root root 4 Feb  6 14:56 /dev/myflsh2 -> sdb2
lrwxrwxrwx. 1 root root 4 Feb  6 14:56 /dev/myflsh3 -> sdb3
$ dd if=/dev/sdb1 of=/dev/null count=10
10+0 records in
10+0 records out
5120 bytes (5.1 kB, 5.0 KiB) copied, 0.0251715 s, 203 kB/s
$ dd if=/dev/sda1 of=/dev/null count=10
dd: failed to open '/dev/sda1': Permission denied
```

Для получения списка устройств PCI и USB вы можете воспользоваться утилитами lspci и lsusb:

```
$ lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC
[Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA
[Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE
(rev 01)
00:02.0 VGA compatible controller: VMware SVGA II Adapter
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit
Ethernet Controller (rev 02)
00:04.0 System peripheral: InnoTek Systemberatung GmbH VirtualBox
Guest Service
00:05.0 Multimedia audio controller: Intel Corporation 82801AA
AC'97 Audio Controller (rev 01)
00:06.0 USB controller: Apple Inc. KeyLargo/Intrepid USB
00:07.0 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev
08)
00:0b.0 USB controller: Intel Corporation 82801FB/FBM/FR/FW/FRW
(ICH6 Family) USB2 EHCI Controller
00:0d.0 SATA controller: Intel Corporation 82801HM/HEM
(ICH8M/ICH8M-E) SATA Controller [AHCI mode] (rev 02)
00:0e.0 Non-Volatile memory controller: InnoTek Systemberatung
GmbH Device 4e56
$ lsusb
Bus 001 Device 003: ID cd12:ef18 SMART TECHNOLOGY INDUSTRIAL LTD.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
$ lsusb -v | head -5
Bus 001 Device 003: ID cd12:ef18 SMART TECHNOLOGY INDUSTRIAL LTD.
Device Descriptor:
```

```
  bLength                18
  bDescriptorType         1
```

Глава 8. Сбор информации, мониторинг и журналирование в РЕД ОС

8.1. Аудит в РЕД ОС

Основными целями при разработке системных вызовов для подсистемы аудита были:

максимально низкая нагрузка на производительность;

отсутствие подобной функциональности в SELinux (и/или в любой другой подсистеме безопасности).

Инструментарий аудита будет работать без внедренных компонент безопасности, но он не может быть использован вместо них, т.е., например, он не будет предоставлять функциональные возможности без действующих подсистем безопасности.

Есть две основные части: одна работает всегда (базовое журналирование в **audit.c**), а другой можно управлять во время загрузки или работы сервера (аудит системных вызовов в **auditsc.c**). Патч включает в себя изменения в **security/selinux/avc.c** в качестве примера, как системные вызовы аудита могут быть использованы совместно с другим кодом, определяющим события для аудита.

Использует сетевой сокет для связи с пользовательским пространством. Если служба запущена - все сообщения журналируются через этот сетевой сокет. Иначе, сообщения журналируются через **printk** посредством службы **syslog** (по умолчанию).

Сообщения могут не журналироваться (настраивается отдельно) в зависимости от настроек аудита, скорости генерации сообщений или загруженности памяти.

Когда какая-то часть ядра генерирует часть сообщения аудита, эта часть будет немедленно послана в пользовательское пространство, и автоматически выставится флаг, указывающий, что этот системный вызов находится под аудитом. Таким образом, при выходе из системного вызова будет сформирована дополнительная информация (если включен аудит системных вызовов).

Во время создания процесса, формируется контекст аудита и привязывается к структуре, описывающей процесс.

Во время входа в системный вызов, заполняется следующая информация

в контексте аудита, если он есть: номер системного вызова, дата и время, но не аргументы.

В ходе работы системного вызова перехватываются обращения к **getname()** и **path_lookup()**. Эти процедуры вызываются, когда ядро действительно собирается искать информацию, для принятия решения, будет ли системный вызов успешно выполнен или нет. Перехватывать вызовы нужно для того, чтобы не допустить копирование информации, которую генерирует **getname**, поскольку **getname** уже сделал приватную (для ядра) копию этой информации.

Следует заметить, что сохранение копий всех аргументов системного вызова усложняет реализацию и требует увеличения расхода ресурсов, что, наиболее вероятно, не нужно. С этим патчем, к примеру, если пользователь непривилегированный, то **chroot** ("foo") будет завершен с ошибкой - "foo" не будет отражено в записи аудита, потому что ядро определяет еще перед поиском "foo", что работа системного вызова не может быть продолжена. Этот подход предотвращает сохранение пользовательской информации, которая может быть ложной или ненадежной (например, из-за атаки на совместно используемую память) в отличие от истинной информации, фактически используемой ядром.

Во время выхода их системного вызова генерируется та часть сообщения аудита, которая ответственна за информацию о системном вызове, включая имена файлов и номера **inode** (если доступны). Сообщение о системном вызове генерируется, только если выставлен флаг, указывающий, что системный вызов находится под аудитом (он выставляется, например, когда SELinux генерирует *avc* сообщение или когда другая часть ядра определяет, что должно формироваться сообщение для аудита). Следует заметить, что полное сообщение аудита приходит в пользовательское пространство по частям, это позволяет не хранить сообщения неопределенный срок внутри ядра.

Во время завершения процесса контекст аудита уничтожается.

Во время генерации сообщения аудита может быть выполнена простая фильтрация (например, для увеличения производительности - отключение аудита системных вызовов, выполняемых от имени пользователя, работающего с базой данных). Фильтрация может быть, как простой, так и сложной. В основном, инструментарий аудита использует компоненты ядра (например, SELinux), чтобы принимать решение о том, что необходимо подвергать аудиту, а что нет.

В файле **/etc/audit/auditd.conf** определяются параметры службы аудита. На одной строке может быть не больше одной директивы. Директива состоит из ключевого слова (названия параметра), знака равенства и соответствующих ему данных (значения параметра). Допустимые ключевые слова:

log_file, log_format, flush, freq, num_logs, max_log_file, max_log_file_action, space_left, action_mail_acct, space_left_action, admin_space_left, admin_space_left_action, disk_full_action и disk_error_action.

Далее даётся описание каждого из них.

Ключ	**Значение ключа**
log_file	Полное имя файла, в который следует записывать протокол.
log_format	Оформление данных в протоколе. Допустимы два значения: raw и nolog. При указании RAW, данные будут записываться в том виде, в котором они получаются от ядра. Значение NOLOG отключает запись данных об аудите. Этот параметр не влияет на обработку данных диспетчером событий системы аудита.
priority_boost	Неотрицательное число, определяющее повышение приоритета выполнения службы аудита. Значение по умолчанию: 3. Для того чтобы не изменять приоритет, укажите 0.
Flush	Стратегия работы с дисковым буфером. Допустимые значения: none, incremental, data и sync. Вариант none, отключает какие-либо дополнительные действия со стороны службы по синхронизации буфера с диском. При значении incremental, запросы на перенос данных из буфера на диск выполняются с частотой задаваемой параметром freq. При значении data данные файла синхронизируются немедленно. Значение sync указывает на необходимость немедленной синхронизации как данных, так и метаданных файла при записи на диск.
Freq	Максимальное число записей протокола, которые могут храниться в буфере. При достижении этого числа производится запись буферированных данных на диск. Данный параметр допустим, только если flush имеет значение incremental.
num_logs	Максимальное число файлов с протоколами. Используется, если параметр max_log_file_action имеет значение rotate. Если указано число меньше 2, при достижении ограничения на размер файла он обнуляется. Значение параметра не должно превышать 99. Значение по умолчанию: 0. При указании большого числа может потребоваться увеличить ограничение на количество ожидающих запросов. Это можно сделать в файле /etc/audit/rules.d/audit.rules.
Dispatcher	Диспетчер - программа, которой (на стандартный ввод) будут передаваться копии сообщений о событиях аудита. Она запускается (с правами администратора) службой аудита при загрузке последней.
disp_qos	Разрешить ли блокирование при взаимодействии с диспетчером. Для передачи информации диспетчеру используется буфер размером 128 Кб.

Ключ	Значение ключа
	Это значение является оптимальным для большинства случаев. Если блокирование запрещено (lossy), то все сообщения, поступающие при полном буфере, не будут доходить до диспетчера (записи о них по-прежнему будут вноситься в файл на диске, если только log_format не равно nolog). В случае, если блокирование разрешено (lossless), служба аудита будет ожидать появления свободного места в очереди, передавать сообщение диспетчеру и только потом записывать его на диск. Допустимые значения: lossy и lossless. Значение по умолчанию - lossy.
max_log_file	Ограничение на размер файла протокола в мегабайтах. Действие, выполняемое при достижении размера файла указанного значения, можно настроить с помощью следующего параметра.
max_log_file_action	Действие, предпринимаемое при достижении размером файла протокола максимального значения. Допустимые значения: ignore, syslog, suspend, rotate и keep_logs. Вариант ignore отключает контроль над размером файла. При значении syslog в системный протокол будет внесено соответствующее сообщение. При значении suspend дальнейшее ведение протокола будет прекращено. Служба по-прежнему будет работать. При значении rotate текущий файл будет переименован и для протокола будет создан новый файл. Имя предыдущего протокола будет дополнено числом 1, а номера других протоколов (если они имеются) будут увеличены на единицу. Таким образом, чем больше номер у протокола, тем он старше. Максимальное число файлов определяется параметром num_logs (естественно, соответствие ему достигается за счёт удаления самых старых протоколов). Такое поведение аналогично поведению утилиты logrotate. Вариант keep_logs аналогичен предыдущему, но число файлов не ограничено.
action_mail_acct	Адрес электронной почты. Значение по умолчанию: root. Если адрес не локальный по отношению к данной системе, необходимо чтобы в ней был настроен механизм отправки почты. В частности, требуется наличие программы /usr/lib/sendmail.
space_left	Минимум свободного пространства в мегабайтах, при достижении которого должно выполняться действие, определяемое следующим параметром.
space_left_action	Действие, предпринимаемое при достижении объёмом свободного пространства на диске указанного минимума. Допустимые значения - ignore, syslog, email, exec, suspend, single и halt. При значении ignore никаких действий не производится. При значении syslog в системный протокол добавляется соответствующая запись. При значении email по адресу, указанному в action_mail_acct, отправляется уведомление. При значении exec запускается программа по указанному пути. Передача параметров не поддерживается. При значении suspend служба аудита прекратит вести протокол событий на диске, но будет продолжать работать. Указание single приведёт к переводу компьютера в однопользовательский режим. Указание halt приведёт к выключению компьютера.
admin_space_left	Критический минимум свободного пространства в мегабайтах, при достижении которого должно выполняться действие, определяемое следующим параметром. Данное действие следует рассматривать как последнюю меру, предпринимаемую перед тем, как закончится место на

Ключ	Значение ключа
	диске. Значение настоящего параметра должно быть меньше значения space_left.
admin_space_left_action	Действие, предпринимаемое при достижении объёмом свободного пространства на диске указанного критического минимума. Допустимые значения - ignore, syslog, email, exec, suspend, single и halt. При значении ignore никаких действий не производится. При значении syslog в системный протокол добавляется соответствующая запись. При значении email по адресу, указанному в action_mail_acct, отправляется уведомление. При значении exec запускается программа по указанному пути. Передача параметров не поддерживается. При значении suspend служба аудита прекратит вести протокол событий на диске, но будет продолжать работать. Указание single приведёт к переводу компьютера в однопользовательский режим. Указание halt приведёт к выключению компьютера.
disk_full_action	Действие, предпринимаемое при обнаружении отсутствия свободного пространства на диске. Допустимые значения - ignore, syslog, email, exec, suspend, single и halt. При значении ignore никаких действий не производится. При значении syslog в системный протокол добавляется соответствующая запись. При значении email по адресу, указанному в action_mail_acct, отправляется уведомление. При значении exec запускается сценарий по указанному пути. Передача параметров сценарию не поддерживается. При значении suspend служба аудита прекратит вести протокол событий на диске, но будет продолжать работать. Указание single приведёт к переводу компьютера в однопользовательский режим. Указание halt приведёт к выключению компьютера.
disk_error_action	Действие, предпринимаемое при возникновении ошибки в работе с диском. Допустимые значения - ignore, syslog, email, exec, suspend, single и halt. При значении ignore никаких действий не производится. При значении syslog в системный протокол добавляется соответствующая запись. При значении email по адресу, указанному в action_mail_acct, отправляется уведомление. При значении exec запускается сценарий по указанному пути. Передача параметров сценарию не поддерживается. При значении suspend служба аудита прекратит вести протокол событий на диске, но будет продолжать работать. Указание single приведёт к переводу компьютера в однопользовательский режим. Указание halt приведёт к выключению компьютера.

В среде CAPP (Controlled Access Protection Profile - контролируемый профиль защиты доступа) ведение протоколов настолько важно, что невозможность его продолжения может служить основанием отказа в доступе к ресурсам. Поэтому рекомендуется выделять для файла **/var/log/audit** специальный раздел. Кроме того, параметру **flush** следует присвоить значение **sync** или **data**.

Для обеспечения полного использования раздела параметрам **max_log_file** и **num_logs** следует присвоить соответствующие значения.

Учитывайте, что чем больше файлов создаётся на диске (и, соответственно, переименовывается), тем больше времени будет уходить на обработку событий при достижении размером очередного файла максимума. Параметру **max_log_file_action** рекомендуется присвоить значение **keep_logs**.

Значение **space_left** должно быть таким, которое позволит администратору вовремя среагировать на предупреждение. Обычно в число действий, выполняемых администратором, входит запуск **aureport -t** и архивирование самых старых протоколов. Значение **space_left** зависит от системы, в частности от частоты поступления сообщений о событиях. Значение **space_left_action** рекомендуется установить в **email**. Если требуется отправка сообщения **snmp trap**, укажите вариант **exec**.

Установите значение **admin_space_left** таким образом, чтобы хватило свободного места для сохранения записей о последующих действиях администратора. Значение параметра **admin_space_left_action** следует установить в **single**, ограничив, таким образом, способы взаимодействия с системной консолью.

Действие, указанное в **disk_full_action**, выполняется, когда в разделе уже не осталось свободного места. Доступ к ресурсам машины должен быть полностью прекращён, т.к. нет возможности контролировать работу системы. Это можно сделать, указав значение **single** или **halt**.

Значение **disk_error_action** следует установить в **syslog**, **single**, либо **halt** в зависимости от соглашения относительно обращения со сбойным аппаратным обеспечением.

Служба **auditd** - это прикладной компонент системы аудита Linux. Он ведёт протокол аудита на диске. Для просмотра протоколов предназначены команды **ausearch** и **aureport**. Команда **auditctl** позволяет настраивать правила аудита. Кроме того, при загрузке загружаются правила из файла /etc/audit/rules.d/audit.rules. Некоторые параметры самой службы можно изменить в файле **auditd.conf**.

Синтаксис:

```
auditd [-f] [-l] [-n]
```

Опции:

Опция	Значение опции
-f	Не переходить в фоновый режим (для отладки). Сообщения программы будут направляться в стандартный вывод для ошибок (stderr), а не в файл.
-l	Включить следование по символическим ссылкам при поиске

Опция	Значение опции
	конфигурационных файлов.
-n	Не создавать дочерний процесс. Для запуска из inittab.

Сигналы:

Сигнал	Значение сигнала
SIGHUP	Перезагрузить конфигурацию - загрузить файл конфигурации с диска. Если в файле не окажется синтаксических ошибок, внесенные в него изменения вступят в силу. При этом в протокол будет добавлена запись о событии DAEMON_CONFIG. В противном случае действия службы будут зависеть от параметров space_left_action, admin_space_left_action, disk_full_action, disk_error_action файла auditd.conf.
SIGTERM	Прекратить обработку событий аудита и завершить работу, о чём предварительно занести запись в протокол.
SIGUSR1	Создать новый файл для протокола, перенумеровав старые файлы или удалив часть из них, в зависимости от параметра max_log_size_action.

Для того чтобы сделать возможным аудит всех процессов, запущенных до службы аудита, добавьте в строку параметров ядра (в конфигурации загрузчика) **audit=1**. В противном случае аудит некоторых процессов будет невозможен.

Утилита **auditctl** используется для контроля поведения, получения состояния и добавления/удаления правил аудита, предоставляемого Linux ядром.

Опции:

Опция	Значение опции
-b backlog	Установить максимальное количество доступных для аудита буферов, ожидающих обработки (значение в ядре по умолчанию - 64). Если все буфера заняты, то флаг сбоя будет выставлен ядром для его дальнейшей обработки.
-e [0..2]	Установить флаг блокировки. 0 позволит на время отключить аудит, включить его обратно можно, передав 1 как параметр. Если установлено значение опции 2, то защитить конфигурацию аудита от изменений. Каждый, кто захочет воспользоваться этой возможностью, может поставить эту команду последней в audit.rules. После этой команды все попытки изменить конфигурацию будут отвергнуты с уведомлением в журналах аудита. В этом случае, чтобы задействовать новую конфигурацию аудита, необходимо перезагрузить систему аудита.
-f [0..2]	Установить способ обработки для флага сбоя. 0=silent 1=printk 2=panic. Эта опция позволяет определить, каким образом ядро будет обрабатывать критические ошибки. Например, флаг сбоя выставляется при следующих условиях: ошибки передачи в пространство службы аудита, превышение лимита буферов, ожидающих обработки, выход за пределы памяти ядра, превышение лимита скорости выдачи сообщений. Значение по умолчанию: 1. Для систем с повышенными требованиями к безопасности, значение 2 может быть более предпочтительно.

Опция	Значение опции
-h	Краткая помощь по аргументам командной строки.
-i	Игнорировать ошибки при чтении правил из файла.
-l	Вывести список всех правил по одному правилу в строке.
-k <ключ>	Установить на правило ключ фильтрации. Ключ фильтрации - это произвольная текстовая строка длиной не больше 31 символа. Ключ помогает уникально идентифицировать записи, генерируемые в ходе аудита за точкой наблюдения.
-m <текст>	Послать в систему аудита пользовательское сообщение. Это может быть сделано только из-под учетной записи root.
-p [r|w|x|a]	Установить фильтр прав доступа для точки наблюдения. r=чтение, w=запись, x=исполнение, a=изменение атрибута. Не путайте эти права доступа с обычными правами доступа к файлу - они определяют типы системных вызовов, которые выполняют данные действия. Заметьте, системные вызовы read и write не включены в этот набор, поскольку логи аудита были бы перегружены информацией о работе этих вызовов.
-r <частота>	Установить ограничение скорости выдачи сообщений в секунду (0 - нет ограничения). Если эта частота не нулевая и она превышается в ходе аудита, флаг сбоя выставляется ядром для выполнения соответствующего действия. Значение по умолчанию: 0.
-R <файл>	Читать правила из файла. Правила должны быть расположены по одному в строке и в том порядке, в каком они должны исполняться. Следующие ограничения накладываются на файл: владельцем должен быть root и доступ на чтение должен быть только у него. Файл может содержать комментарии, начинающиеся с символа «#». Правила, расположенные в файле, идентичны тем, что набираются в командной строке, без указания «auditctl».
-s	Получить статус аудита.
-a <список, действие>	Добавить правило с указанным действием к концу списка. Заметьте, что запятая разделяет эти два значения. Отсутствие запятой вызовет ошибку. Ниже описаны имена доступных списков: **task** - добавить правило к списку, отвечающему за процессы. Этот список правил используется только во время создания процесса - когда родительский процесс вызывает fork() или clone(). При использовании этого списка вы можете использовать только те поля, которые известны во время создания процесса: uid, gid и т.д. entry - добавить правило к списку, отвечающему за точки входа системных вызовов. Этот список применяется, когда необходимо создать событие для аудита, привязанное к точкам входа системных вызовов. **exit** - добавить правило к списку, отвечающему за точки выхода из системных вызовов. Этот список применяется, когда необходимо создать событие для аудита, привязанное к точкам выхода из системных вызовов. **user** - добавить правило, отвечающее за список фильтрации пользовательских сообщений. Этот список используется ядром, чтобы отфильтровать события приходящие из пользовательского пространства перед тем, как они будут переданы службе аудита. Необходимо отметить, что только следующие поля могут быть использованы: uid, auid, gid и pid. Все остальные поля будут

Опция	Значение опции
	обработаны, как если бы они не совпали. **exclude** - добавить правило к списку, отвечающему за фильтрацию событий определенного типа. Этот список используется, чтобы отфильтровывать ненужные события. Например, если вы не хотите видеть avc сообщения, вы должны использовать этот список. Тип сообщения задается в поле msgtype.

Ниже описаны доступные действия для правил:

Действие	Описание действия
never	Аудит не будет генерировать никаких записей. Это может быть использовано для подавления генерации событий. Обычно необходимо подавлять генерацию вверху списка, а не внизу, т.к. событие инициируется на первом совпавшем правиле.
always	Установить контекст аудита. Всегда заполнять его во время входа в системный вызов и всегда генерировать запись во время выхода из системного вызова.
-A <список, действие>	Добавить правило с указанным действием в начало списка.
-d <список, действие>	Удалить правило с указанным действием из списка. Правило удаляется только в том случае, если полностью совпали и имя системного вызова, и поля сравнения.
-D	Удалить все правила и точки наблюдения.
-S <имя или номер системного вызова>/[all]	Любой номер или имя системного вызова может быть использован. Также возможно использование ключевого слова all. Если какой-либо процесс выполняет указанный системный вызов, то аудит генерирует соответствующую запись. Если значения полей сравнения заданы, а системный вызов не указан, правило будет применяться ко всем системным вызовам. В одном правиле может быть задано несколько системных вызовов - это положительно сказывается на производительности, поскольку заменяет обработку нескольких правил.
-F [n=v | n!=v | n<v | n>v | n<=v | n>=v | n&v | n&=v]	Задать поле сравнения для правила. Атрибуты поля следующие: объект, операция, значение. Вы можете задать до 64 полей сравнения в одной команде. Каждое новое поле должно начинаться с -F. Аудит будет генерировать запись, если произошло совпадение по всем полями сравнения. Допустимо использование одного из следующих 8 операторов: равно, не равно, меньше, больше, меньше либо равно, больше либо равно, битовая маска (n&v) и битовая проверка (n&=v). Битовая проверка выполняет операцию «and» над значениями и проверяет, равны ли они. Битовая маска просто выполняет операцию «and». Поля, оперирующие с идентификатором пользователя, могут также работать с именем пользователя - программа автоматически получит идентификатор пользователя из его имени. То же самое можно сказать и про имя группы. Поля сравнения могут быть заданы для следующих объектов: a0, a1, a2, a3 - четыре первых аргумента, переданных системному вызову. Строковые аргументы не поддерживаются. Это связано с тем, что ядро должно получать указатель на строку, а проверка поля по значению адреса указателя не желательна. Таким образом, вы должны использовать только цифровые

Действие	Описание действия
	значения. arch - архитектура процессора, на котором выполняется системный вызов. Используйте 'uname -m', чтобы определить архитектуру. Если вы не знаете архитектуру вашей машины, но хотите использовать таблицу 32-х битных системных вызовов, и ваша машина поддерживает 32 бита, вы можете использовать b32. Подобно этому b64 может быть использовано для использования таблицы 64-х битных системных вызовов. auid - это аббревиатура: audit uid - идентификатор пользователя, использованный для входа в систему. devmajor - главный номер устройства (Device Major Number). devminor - вспомогательный номер устройства (Device Minor Number). egid - действительный идентификатор группы; euid - действительный идентификатор пользователя; exit - значение, возвращаемое системным вызовом при выходе; fsgid - идентификатор группы, применяемый к файловой системе; fsuid - идентификатор пользователя, применяемый к файловой системе; gid - идентификатор группы; inode - номер inode; key - альтернативный способ установить ключ фильтрации. Смотри выше описание опции -k; msgtype спользуется для проверки совпадения с числом, описывающим тип сообщения. Может быть использован только в списке exclude; obj_user - имя пользователя-владельца ресурса (в контексте SELinux); obj_role - роль ресурса (в контексте SELinux); obj_type - тип ресурса (в контексте SELinux); obj_lev_low - нижний уровень ресурса (в контексте SELinux); obj_lev_high - верхний уровень ресурса (в контексте SELinux); path - полный путь к файлу для точки наблюдения. Смотри ниже описание опции "-w". Может быть использован только в списке exit; perm - фильтр прав доступа для файловых операций. Смотри выше описание опции "-p". Может быть использован только в списке exit; pers - персональный номер операционной системы; pid - идентификатор процесса; ppid - идентификатор родительского процесса; subj_user - имя пользователя-владельца процесса (в контексте SELinux); subj_role - роль процесса (в контексте SELinux); subj_type - тип процесса (в контексте SELinux); subj_sen - чувствительность процесса (в контексте SELinux); subj_clr - допуск процесса (в контексте SELinux); sgid - установленный идентификатор группы; success - если значение, возвращаемое системным вызовом, больше либо равно 0, данный объект будет равен "true/yes", иначе "false/no". При создании правила используйте 1 вместо "true/yes" и 0 вместо "false/no"; suid - установленный идентификатор пользователя; uid - идентификатор пользователя.
-w <путь>	Добавить точку наблюдения за файловым объектом, находящемуся по указанному пути. Вы не можете добавлять точку наблюдения к каталогу

Действие	Описание действия
	верхнего уровня - это запрещено ядром. Групповые символы (wildcards) также не могут быть использованы, попытки их использования будут генерировать предупреждающее сообщение. Внутренне точки наблюдения реализованы как слежение за inode. Таким образом, если вы установите точку наблюдения за каталогом, вы увидите файловые события, которые в действительности будут означать обновления метаданных этой inode, и вы можете не увидеть событий, непосредственно связанных с файлами. Если вам необходимо следить за всеми файлами в каталоге, рекомендуется создавать индивидуальную точку наблюдения для каждого файла. В противоположность к правилам аудита системных вызовов, точки наблюдения не оказывают влияния на производительность, связанную с количеством правил посылаемых в ядро.
-W <путь>	Удалить точку наблюдения за файловым объектом, находящемуся по указанному пути.

Примеры базовых правил и конструкций

Приведенные ниже команды будут выполняться только в ходе работы одного сеанса ОС, если вы хотите сделать это правило постоянным, добавьте его в конец файла /etc/audit/rules.d/audit.rules следующим образом:

-w /var/log/audit/ -p wra -k auditlog

Самоаудит

Отображение успешных и неудачных попыток чтения информации из записей аудита:

```
auditctl -w /var/log/audit/ -p wra -k auditlog
```

Отображение изменения конфигурации аудита, происходящей во время работы функций журналирования:

```
auditctl -w /etc/audit/ -p wra -k auditconfig
auditctl -w /etc/libaudit.conf -p wra -k auditconfig
```

Отслеживание использования инструментов управления аудитом:

```
auditctl -w /sbin/auditctl -p x -k audittools
auditctl -w /sbin/auditd -p x -k audittools
auditctl -w /usr/sbin/auditd -p x -k audittools
auditctl -w /usr/sbin/augenrules -p x -k audittoolss
```

Отслеживание доступа к прочим утилитам службы аудирования:

```
 auditctl -a always,exit -F path=/usr/sbin/ausearch -F perm=x -k
auditreport
 auditctl -a always,exit -F path=/usr/sbin/aureport -F perm=x -k
auditreport
 auditctl -a always,exit -F path=/usr/sbin/aulast -F perm=x -k
```

```
auditreport
 auditctl -a always,exit -F path=/usr/sbin/aulastlogin -F perm=x -
k auditreport
  auditctl -a always,exit -F path=/usr/sbin/auvirt -F perm=x -k
auditreport
```

Фильтры

Игнорирование событий SELinux:

```
 auditctl -a always,exclude -F msgtype=AVC
```

Игнорирование событий с меткой рабочей директории:

```
 auditctl -a always,exclude -F msgtype=CWD
```

Игнорирование событий с меткой идентификатора криптографического ключа, используемого в криптографических целях:

```
 auditctl -a always,exclude -F msgtype=CRYPTO_KEY_USER
```

Игнорирование событий cron:

```
 auditctl -a never,user -F subj_type=crond_t
 auditctl -a exit,never -F subj_type=crond_t
```

Игнорирование событий chrony:

```
 auditctl -a never,exit -F arch=b32 -S adjtimex -F auid=unset -F
uid=chrony -F subj_type=chronyd_t
 auditctl -a never,exit -F arch=b64 -S adjtimex -F auid=unset -F
uid=chrony -F subj_type=chronyd_t
```

Игнорирование сообщений, связанных с инструментами виртуальной машины:

```
 ###Игнорирование ошибок VMware tools

  auditctl -a never,exit -F arch=b32 -S fork -F success=0 -F
path=/usr/lib/vmware-tools -F subj_type=initrc_t -F exit=-2
  auditctl -a never,exit -F arch=b64 -S fork -F success=0 -F
path=/usr/lib/vmware-tools -F subj_type=initrc_t -F exit=-2
    auditctl -a exit,never -F arch=b32 -S all -F
exe=/usr/bin/vmtoolsd
    auditctl -a exit,never -F arch=b64 -S all -F
exe=/usr/bin/vmtoolsd
```

Фильтр событий большого объема:

```
 ### Игнорирование событий, связанных с работой shm и lvm

   auditctl -a never,exit -F arch=b32 -F dir=/dev/shm -k
```

```
sharedmemaccess
   auditctl   -a   never,exit   -F   arch=b64   -F   dir=/dev/shm   -k
sharedmemaccess
  auditctl  -a  never,exit  -F  arch=b32  -F  dir=/var/lock/lvm  -k
locklvm
  auditctl  -a  never,exit  -F  arch=b64  -F  dir=/var/lock/lvm  -k
locklvm
```

Игнорирование событий с меткой объекта FileBeat:

```
  auditctl  -a  never,exit  -F  arch=b32  -F  path=/opt/filebeat  -k
filebeat
  auditctl  -a  never,exit  -F  arch=b64  -F  path=/opt/filebeat  -k
filebeat
```

Базовые правила

Отслеживание изменений параметров Kernel:

```
 auditctl -w /etc/sysctl.conf -p wa -k sysctl
 auditctl -w /etc/sysctl.d -p wa -k sysctl
```

Отслеживание удачной и неудачной загрузки Kernel module:

```
    auditctl    -a    always,exit    -F    perm=x    -F    auid!=-1    -F
path=/sbin/insmod -k modules
    auditctl    -a    always,exit    -F    perm=x    -F    auid!=-1    -F
path=/sbin/modprobe -k modules
 auditctl -a always,exit -F perm=x -F auid!=-1 -F path=/sbin/rmmod
-k modules
   auditctl   -a   always,exit   -F   arch=b64   -S   finit_module   -S
init_module -S delete_module -F auid!=-1 -k modules
   auditctl   -a   always,exit   -F   arch=b32   -S   finit_module   -S
init_module -S delete_module -F auid!=-1 -k modules
```

Отслеживание изменений конфигурации Modprobe:

```
 auditctl -w /etc/modprobe.conf -p wa -k modprobe
 auditctl -w /etc/modprobe.d -p wa -k modprobe
```

Отслеживание использования Kexec:

```
 auditctl -a always,exit -F arch=b64 -S kexec_load -k KEXEC
 auditctl -a always,exit -F arch=b32 -S sys_kexec_load -k KEXEC
```

Отслеживание использования специальных файлов (утилита mknod):

```
  auditctl  -a  always,exit  -F  arch=b32  -S  mknod  -S  mknodat  -k
specialfiles
  auditctl  -a  always,exit  -F  arch=b64  -S  mknod  -S  mknodat  -k
specialfiles
```

Отслеживание использования утилиты mount:

```
 auditctl -a always,exit -F arch=b64 -S mount -S umount2 -F auid!
=-1 -k mount
 auditctl -a always,exit -F arch=b32 -S mount -S umount -S umount2
-F auid!=-1 -k mount
```

Отслеживание использования утилиты swap:

```
 auditctl -a always,exit -F arch=b64 -S swapon -S swapoff -F auid!
=-1 -k swap
 auditctl -a always,exit -F arch=b32 -S swapon -S swapoff -F auid!
=-1 -k swap
```

Отслеживание изменения времени:

```
 auditctl -a always,exit -F arch=b32 -F uid!=ntp -S adjtimex -S
settimeofday -S clock_settime -k time
 auditctl -a always,exit -F arch=b64 -F uid!=ntp -S adjtimex -S
settimeofday -S clock_settime -k time
```

Отслеживание изменения локального времени:

```
auditctl -w /etc/localtime -p wa -k localtime
```

Отслеживание использования утилиты stunnel:

```
auditctl -w /usr/sbin/stunnel -p x -k stunnel
auditctl -w /usr/bin/stunnel -p x -k stunnel
```

Отслеживание изменения файлов конфигурации cron и запланированных заданий:

```
auditctl -w /etc/cron.allow -p wa -k cron
auditctl -w /etc/cron.deny -p wa -k cron
auditctl -w /etc/cron.d/ -p wa -k cron
auditctl -w /etc/cron.daily/ -p wa -k cron
auditctl -w /etc/cron.hourly/ -p wa -k cron
auditctl -w /etc/cron.monthly/ -p wa -k cron
auditctl -w /etc/cron.weekly/ -p wa -k cron
auditctl -w /etc/crontab -p wa -k cron
auditctl -w /var/spool/cron/ -k cron
```

Отслеживание изменения баз пользователей, групп, паролей:

```
auditctl -w /etc/group -p wa -k etcgroup
auditctl -w /etc/passwd -p wa -k etcpasswd
auditctl -w /etc/gshadow -k etcgroup
auditctl -w /etc/shadow -k etcpasswd
auditctl -w /etc/security/opasswd -k opasswd
```

Отслеживание изменения файлов sudo:

```
auditctl -w /etc/sudoers -p wa -k actions
auditctl -w /etc/sudoers.d/ -p wa -k actions
```

Отслеживание использования passwd:

```
auditctl -w /usr/bin/passwd -p x -k passwd_modification
```

Отслеживание использования инструментов изменения идентификаторов группы:

```
auditctl -w /usr/sbin/groupadd -p x -k group_modification
auditctl -w /usr/sbin/groupmod -p x -k group_modification
auditctl -w /usr/sbin/addgroup -p x -k group_modification
auditctl -w /usr/sbin/useradd -p x -k user_modification
auditctl -w /usr/sbin/userdel -p x -k user_modification
auditctl -w /usr/sbin/usermod -p x -k user_modification
auditctl -w /usr/sbin/adduser -p x -k user_modification
```

Отслеживание изменения информации и конфигурации login:

```
auditctl -w /etc/login.defs -p wa -k login
auditctl -w /etc/securetty -p wa -k login
auditctl -w /var/log/faillog -p wa -k login
auditctl -w /var/log/lastlog -p wa -k login
auditctl -w /var/log/tallylog -p wa -k login
```

Правила сетевого окружения

Отслеживание изменения hostname:

```
  auditctl -a always,exit -F arch=b32 -S sethostname -S
setdomainname -k network_modifications
  auditctl -a always,exit -F arch=b64 -S sethostname -S
setdomainname -k network_modifications
```

Отслеживание удачных попыток подключения IPv4:

```
 auditctl -a always,exit -F arch=b64 -S connect -F a2=16 -F
success=1 -F key=network_connect_4
 auditctl -a always,exit -F arch=b32 -S connect -F a2=16 -F
success=1 -F key=network_connect_4
```

Отслеживание удачных попыток подключения IPv6:

```
 auditctl -a always,exit -F arch=b64 -S connect -F a2=28 -F
success=1 -F key=network_connect_6
 auditctl -a always,exit -F arch=b32 -S connect -F a2=28 -F
success=1 -F key=network_connect_6
```

Отслеживание изменения файлов конфигурации сетевых настроек:

```
 auditctl -w /etc/hosts -p wa -k network_modifications
 auditctl -w /etc/sysconfig/network -p wa -k network_modifications
    auditctl    -w    /etc/sysconfig/network-scripts    -p    w    -k
network_modifications
 auditctl -w /etc/network/ -p wa -k network
 auditctl -a always,exit -F dir=/etc/NetworkManager/ -F perm=wa -k
network_modifications
```

Отслеживание изменения файлов issue:

```
 auditctl -w /etc/issue -p wa -k etcissue
 auditctl -w /etc/issue.net -p wa -k etcissue
```

Отслеживание изменения скриптов при запуске ОС:

```
 auditctl -w /etc/inittab -p wa -k init
 auditctl -w /etc/init.d/ -p wa -k init
 auditctl -w /etc/init/ -p wa -k init
```

Отслеживание изменения файлов libpath:

```
 auditctl -w /etc/ld.so.conf -p wa -k libpath
 auditctl -w /etc/ld.so.conf.d -p wa -k libpath
```

Отслеживание изменения предварительной загрузки общесистемной библиотеки
(LD_PRELOAD):

```
 auditctl -w /etc/ld.so.preload -p wa -k systemwide_preloads
```

Отслеживание изменения конфигурации PAM и связанных с ним файлов:

```
 auditctl -w /etc/pam.d/ -p wa -k pam
 auditctl -w /etc/security/limits.conf -p wa -k pam
 auditctl -w /etc/security/limits.d -p wa -k pam
 auditctl -w /etc/security/pam_env.conf -p wa -k pam
 auditctl -w /etc/security/namespace.conf -p wa -k pam
 auditctl -w /etc/security/namespace.d -p wa -k pam
 auditctl -w /etc/security/namespace.init -p wa -k pam
```

Отслеживание изменения конфигурации mail:

```
 auditctl -w /etc/aliases -p wa -k mail
 auditctl -w /etc/postfix/ -p wa -k mail
 auditctl -w /etc/exim4/ -p wa -k mail
```

Отслеживание изменения конфигурации ssh:

```
 auditctl -w /etc/ssh/sshd_config -k sshd
 auditctl -w /etc/ssh/sshd_config.d -k sshd
```

Отслеживание изменения подделки ключа ssh:

```
auditctl -w /root/.ssh -p wa -k rootkey
```

Правила служб

Отслеживание изменения и использования файлов systemd:

```
auditctl -w /bin/systemctl -p x -k systemd
auditctl -w /etc/systemd/ -p wa -k systemd
auditctl -w /usr/lib/systemd -p wa -k systemd
```

Отслеживание событий SELinux, которые изменяют систему мандатного управления доступом:

```
auditctl -w /etc/selinux/ -p wa -k mac_policy
```

Отслеживание событий сбоя доступа к критическим элементам:

```
 auditctl -a always,exit -F arch=b64 -S open -F dir=/etc -F
success=0 -k unauthedfileaccess
 auditctl -a always,exit -F arch=b64 -S open -F dir=/bin -F
success=0 -k unauthedfileaccess
 auditctl -a always,exit -F arch=b64 -S open -F dir=/sbin -F
success=0 -k unauthedfileaccess
 auditctl -a always,exit -F arch=b64 -S open -F dir=/usr/bin -F
success=0 -k unauthedfileaccess
 auditctl -a always,exit -F arch=b64 -S open -F dir=/usr/sbin -F
success=0 -k unauthedfileaccess
 auditctl -a always,exit -F arch=b64 -S open -F dir=/var -F
success=0 -k unauthedfileaccess
 auditctl -a always,exit -F arch=b64 -S open -F dir=/home -F
success=0 -k unauthedfileaccess
 auditctl -a always,exit -F arch=b64 -S open -F dir=/srv -F
success=0 -k unauthedfileaccess
```

Отслеживание использования приложений для изменения идентификатора процесса (переключение учетных записей):

```
auditctl -w /bin/su -p x -k priv_esc
auditctl -w /usr/bin/sudo -p x -k priv_esc
auditctl -w /etc/sudoers -p rw -k priv_esc
auditctl -w /etc/sudoers.d -p rw -k priv_esc
```

Отслеживание использования приложений, изменяющих состояние питания:

```
auditctl -w /sbin/shutdown -p x -k power
auditctl -w /sbin/poweroff -p x -k power
auditctl -w /sbin/reboot -p x -k power
```

```
auditctl -w /sbin/halt -p x -k power
```

Отслеживание изменения информации об инициации сеанса:

```
auditctl -w /var/run/utmp -p wa -k session
auditctl -w /var/log/btmp -p wa -k session
auditctl -w /var/log/wtmp -p wa -k session
```

Отслеживание модификации дискреционного контроля доступа:

```
 auditctl -a always,exit -F arch=b32 -S chmod -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S chown -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S fchmod -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S fchmodat -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S fchown -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S fchownat -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S fremovexattr -F auid>=1000
-F auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S fsetxattr -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S lchown -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S lremovexattr -F auid>=1000
-F auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S lsetxattr -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S removexattr -F auid>=1000
-F auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b32 -S setxattr -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S chmod -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S chown -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S fchmod -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S fchmodat -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S fchown -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S fchownat -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S fremovexattr -F auid>=1000
-F auid!=-1 -k perm_mod
```

```
 auditctl -a always,exit -F arch=b64 -S fsetxattr -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S lchown -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S lremovexattr -F auid>=1000
-F auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S lsetxattr -F auid>=1000 -F
auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S removexattr -F auid>=1000
-F auid!=-1 -k perm_mod
 auditctl -a always,exit -F arch=b64 -S setxattr -F auid>=1000 -F
auid!=-1 -k perm_mod
```

Специальные правила

Отслеживание запросов на получение информации о пользователе:

```
 auditctl -w /usr/bin/whoami -p x -k recon
 auditctl -w /usr/bin/id -p x -k recon
 auditctl -w /bin/hostname -p x -k recon
 auditctl -w /bin/uname -p x -k recon
 auditctl -w /etc/issue -p r -k recon
 auditctl -w /etc/hostname -p r -k recon
```

Отслеживание подозрительной деятельности:

```
 ### Подразумевается, что использование специализированных утилит,
связанных с работой с/в
   ### внутренней/внешней сетью, может быть отслежено и
зажурналировано.
 auditctl -w /usr/bin/wget -p x -k susp_activity
 auditctl -w /usr/bin/curl -p x -k susp_activity
 auditctl -w /usr/bin/base64 -p x -k susp_activity
 auditctl -w /bin/nc -p x -k susp_activity
 auditctl -w /bin/netcat -p x -k susp_activity
 auditctl -w /usr/bin/ncat -p x -k susp_activity
 auditctl -w /usr/bin/ss -p x -k susp_activity
 auditctl -w /usr/bin/netstat -p x -k susp_activity
 auditctl -w /usr/bin/ssh -p x -k susp_activity
 auditctl -w /usr/bin/scp -p x -k susp_activity
 auditctl -w /usr/bin/sftp -p x -k susp_activity
 auditctl -w /usr/bin/ftp -p x -k susp_activity
 auditctl -w /usr/bin/socat -p x -k susp_activity
 auditctl -w /usr/bin/wireshark -p x -k susp_activity
 auditctl -w /usr/bin/tshark -p x -k susp_activity
 auditctl -w /usr/bin/rawshark -p x -k susp_activity
 auditctl -w /usr/bin/rdesktop -p x -k Remote_Access_Tools
 auditctl -w /usr/local/bin/rdesktop -p x -k Remote_Access_Tools
 auditctl -w /usr/bin/wlfreerdp -p x -k susp_activity
 auditctl -w /usr/bin/xfreerdp -p x -k Remote_Access_Tools
```

```
auditctl -w /usr/local/bin/xfreerdp -p x -k Remote_Access_Tools
auditctl -w /usr/bin/nmap -p x -k susp_activity
```

Отслеживание использования утилит архивации и сжатия данных:

```
auditctl -w /usr/bin/zip -p x -k Data_Compressed
auditctl -w /usr/bin/gzip -p x -k Data_Compressed
auditctl -w /usr/bin/tar -p x -k Data_Compressed
auditctl -w /usr/bin/bzip2 -p x -k Data_Compressed
auditctl -w /usr/bin/lzip -p x -k Data_Compressed
auditctl -w /usr/local/bin/lzip -p x -k Data_Compressed
auditctl -w /usr/bin/lz4 -p x -k Data_Compressed
auditctl -w /usr/local/bin/lz4 -p x -k Data_Compressed
auditctl -w /usr/bin/lzop -p x -k Data_Compressed
auditctl -w /usr/local/bin/lzop -p x -k Data_Compressed
auditctl -w /usr/bin/plzip -p x -k Data_Compressed
auditctl -w /usr/local/bin/plzip -p x -k Data_Compressed
auditctl -w /usr/bin/pbzip2 -p x -k Data_Compressed
auditctl -w /usr/local/bin/pbzip2 -p x -k Data_Compressed
auditctl -w /usr/bin/lbzip2 -p x -k Data_Compressed
auditctl -w /usr/local/bin/lbzip2 -p x -k Data_Compressed
auditctl -w /usr/bin/pixz -p x -k Data_Compressed
auditctl -w /usr/local/bin/pixz -p x -k Data_Compressed
auditctl -w /usr/bin/pigz -p x -k Data_Compressed
auditctl -w /usr/local/bin/pigz -p x -k Data_Compressed
auditctl -w /usr/bin/unpigz -p x -k Data_Compressed
auditctl -w /usr/local/bin/unpigz -p x -k Data_Compressed
auditctl -w /usr/bin/zstd -p x -k Data_Compressed
auditctl -w /usr/local/bin/zstd -p x -k Data_Compressed
```

Отслеживание подозрительной активности в sbin:

```
### Подразумевается, что использование специализированных утилит
может быть отслежено и
### зажурналировано.
auditctl -w /sbin/iptables -p x -k sbin_susp
auditctl -w /sbin/ip6tables -p x -k sbin_susp
auditctl -w /sbin/ifconfig -p x -k sbin_susp
auditctl -w /usr/sbin/arptables -p x -k sbin_susp
auditctl -w /usr/sbin/ebtables -p x -k sbin_susp
auditctl -w /sbin/xtables-nft-multi -p x -k sbin_susp
auditctl -w /usr/sbin/nft -p x -k sbin_susp
auditctl -w /usr/sbin/tcpdump -p x -k sbin_susp
auditctl -w /usr/sbin/traceroute -p x -k sbin_susp
auditctl -w /usr/sbin/ufw -p x -k sbin_susp
```

Отслеживание отправки вызовов dbus:

```
### Может указывать на повышение привилегий CVE-2021-3560
auditctl -w /usr/bin/dbus-send -p x -k dbus_send
```

```
auditctl -w /usr/bin/gdbus -p x -k gdubs_call
```

Отслеживание вызовов pkexec:

```
### Может указывать на повышение привилегий CVE-2021-4034
auditctl -w /usr/bin/pkexec -p x -k pkexec
```

Отслеживание подозрительной активности, связанной с shell:

```
auditctl -w /bin/ash -p x -k susp_shell
auditctl -w /bin/bash -p x -k susp_shell
auditctl -w /bin/csh -p x -k susp_shell
auditctl -w /bin/dash -p x -k susp_shell
auditctl -w /bin/busybox -p x -k susp_shell
auditctl -w /bin/ksh -p x -k susp_shell
auditctl -w /bin/fish -p x -k susp_shell
auditctl -w /bin/tcsh -p x -k susp_shell
auditctl -w /bin/tclsh -p x -k susp_shell
auditctl -w /bin/zsh -p x -k susp_shell
auditctl -w /bin/xonsh -p x -k susp_shell
auditctl -w /usr/local/bin/xonsh -p x -k susp_shell
auditctl -w /bin/open -p x -k susp_shell
auditctl -w /bin/sh -p x -k susp_shell
auditctl -w /bin/rbash -p x -k susp_shell
```

Отслеживание изменения конфигурации Shell/profile configurations:

```
auditctl -w /etc/profile.d/ -p wa -k shell_profiles
auditctl -w /etc/profile -p wa -k shell_profiles
auditctl -w /etc/shells -p wa -k shell_profiles
auditctl -w /etc/bashrc -p wa -k shell_profiles
auditctl -w /etc/csh.cshrc -p wa -k shell_profiles
auditctl -w /etc/csh.login -p wa -k shell_profiles
auditctl -w /etc/fish/ -p wa -k shell_profiles
auditctl -w /etc/zsh/ -p wa -k shell_profiles
```

Отслеживание злоупотребления привилегиями:

```
 ### Целью этого правила является обнаружение случаев, когда
администратор злоупотребляет полномочиями,
### путем просмотра домашнего каталога пользователя.
 auditctl -a always,exit -F dir=/home -F uid=0 -F auid>=1000 -F
auid!=-1 -C auid!=obj_uid -k power_abuse
```

Отслеживание создания сокетов:

```
### Отслеживает как IPv4, так и IPv6
 auditctl -a always,exit -F arch=b32 -S socket -F a0=2 -k
Exfiltration_Over_Other_Network_Medium
 auditctl -a always,exit -F arch=b64 -S socket -F a0=2 -k
Exfiltration_Over_Other_Network_Medium
```

```
  auditctl -a always,exit -F arch=b32 -S socket -F a0=10 -k
Exfiltration_Over_Other_Network_Medium
  auditctl -a always,exit -F arch=b64 -S socket -F a0=10 -k
Exfiltration_Over_Other_Network_Medium
```

Отслеживание создания анонимного файла:

```
   ### Эти правила следят за использованием memfd_create,
"memfd_create" создает анонимный файл и возвращает
 ### дескриптор файла для доступа к нему
  auditctl -a always,exit -F arch=b64 -S memfd_create -F
key=anon_file_create
  auditctl -a always,exit -F arch=b32 -S memfd_create -F
key=anon_file_create
```

Правила управления ПО

Отслеживание использования rpm/yum:

```
auditctl -w /usr/bin/rpm -p x -k software_mgmt
auditctl -w /usr/bin/yum -p x -k software_mgmt
```

Отслеживание использования dnf:

```
auditctl -w /usr/bin/dnf -p x -k software_mgmt
```

Отслеживание использования pip:

```
auditctl -w /usr/bin/pip -p x -k third_party_software
auditctl -w /usr/local/bin/pip -p x -k third_party_software
auditctl -w /usr/bin/pip3 -p x -k third_party_software
auditctl -w /usr/local/bin/pip3 -p x -k third_party_software
```

Отслеживание использования npm:

```
auditctl -w /usr/bin/npm -p x -k third_party_software
```

Отслеживание использования CPAN (Comprehensive Perl Archive Network):

```
auditctl -w /usr/bin/cpan -p x -k third_party_software
```

Отслеживание использования ruby (RubyGems):

```
auditctl -w /usr/bin/gem -p x -k third_party_software
```

Отслеживание использования Lua:

```
auditctl -w /usr/bin/luarocks -p x -k third_party_software
```

Правила управления специальным ПО

Отслеживание изменений файлов puppet:

```
auditctl -w /etc/puppet/ssl -p wa -k puppet_ssl
```

Отслеживание изменений файлов chef:

```
auditctl -w /etc/chef -p wa -k soft_chef
```

Отслеживание изменений файлов otter:

```
auditctl -w /etc/otter -p wa -k soft_otter
```

Отслеживание использования утилиты grep:

```
auditctl -w /usr/bin/grep -p x -k T1081_Credentials_In_Files
auditctl -w /usr/bin/egrep -p x -k T1081_Credentials_In_Files
auditctl -w /usr/bin/ugrep -p x -k T1081_Credentials_In_Files
```

Отслеживание использования и изменения файлов конфигурации docker:

```
auditctl -w /usr/bin/dockerd -k docker
auditctl -w /usr/bin/docker -k docker
auditctl -w /usr/bin/docker-containerd -k docker
auditctl -w /usr/bin/docker-runc -k docker
auditctl -w /var/lib/docker -k docker
auditctl -w /etc/docker -k docker
auditctl -w /etc/sysconfig/docker -k docker
auditctl -w /etc/sysconfig/docker-storage -k docker
auditctl -w /usr/lib/systemd/system/docker.service -k docker
auditctl -w /usr/lib/systemd/system/docker.socket -k docker
```

Отслеживание использования систем виртуализации:

```
 auditctl -w /usr/bin/qemu-system-x86_64 -p x -k qemu-system-
x86_64
auditctl -w /usr/bin/qemu-img -p x -k qemu-img
auditctl -w /usr/bin/qemu-kvm -p x -k qemu-kvm
auditctl -w /usr/bin/qemu -p x -k qemu
auditctl -w /usr/bin/virtualbox -p x -k virtualbox
auditctl -w /usr/bin/virt-manager -p x -k virt-manager
auditctl -w /usr/bin/VBoxManage -p x -k VboxManage
```

Отслеживание использования утилиты kubelet (Kubernetes):

```
auditctl -w /usr/bin/kubelet -k kubelet
```

«Тяжелые» события

Примечание.

Правила, представленные ниже, требуют большого объема свободного пространства в среде.

Отслеживание применения команд с правами пользователя root:

```
 auditctl -a always,exit -F arch=b64 -F euid=0 -F auid>=1000 -F
auid!=4294967295 -S execve -k rootcmd
 auditctl -a always,exit -F arch=b32 -F euid=0 -F auid>=1000 -F
auid!=4294967295 -S execve -k rootcmd
```

Отслеживание события удаления файлов пользователем:

```
 auditctl -a always,exit -F arch=b32 -S rmdir -S unlink -S
unlinkat -S rename -S renameat -F auid>=1000 -F auid!=-1 -k delete
 auditctl -a always,exit -F arch=b64 -S rmdir -S unlink -S
unlinkat -S rename -S renameat -F auid>=1000 -F auid!=-1 -k delete
```

Отслеживание события неудачного доступа к файлу:

```
auditctl -a always,exit -F arch=b32 -S creat -S open -S openat -S
open_by_handle_at -S truncate -S ftruncate -F exit=-EACCES -F
auid>=1000 -F auid!=-1 -k file_access
auditctl -a always,exit -F arch=b32 -S creat -S open -S openat -S
open_by_handle_at -S truncate -S ftruncate -F exit=-EPERM -F
auid>=1000 -F auid!=-1 -k file_access
auditctl -a always,exit -F arch=b64 -S creat -S open -S openat -S
open_by_handle_at -S truncate -S ftruncate -F exit=-EACCES -F
auid>=1000 -F auid!=-1 -k file_access
auditctl -a always,exit -F arch=b64 -S creat -S open -S openat -S
open_by_handle_at -S truncate -S ftruncate -F exit=-EPERM -F
auid>=1000 -F auid!=-1 -k file_access
```

Отслеживание события неудачного создания:

```
     auditctl     -a     always,exit     -F     arch=b32     -S
creat,link,mknod,mkdir,symlink,mknodat,linkat,symlinkat -F exit=-
EACCES -k file_creation
     auditctl     -a     always,exit     -F     arch=b64     -S
mkdir,creat,link,symlink,mknod,mknodat,linkat,symlinkat -F exit=-
EACCES -k file_creation
auditctl -a always,exit -F arch=b32 -S link,mkdir,symlink,mkdirat
-F exit=-EPERM -k file_creation
auditctl -a always,exit -F arch=b64 -S mkdir,link,symlink,mkdirat
-F exit=-EPERM -k file_creation
```

Отслеживание события неудачной модификации:

```
 auditctl -a always,exit -F arch=b32 -S rename -S renameat -S
truncate -S chmod -S setxattr -S lsetxattr -S removexattr -S
lremovexattr -F exit=-EACCES -k file_modification
 auditctl -a always,exit -F arch=b64 -S rename -S renameat -S
truncate -S chmod -S setxattr -S lsetxattr -S removexattr -S
lremovexattr -F exit=-EACCES -k file_modification
 auditctl -a always,exit -F arch=b32 -S rename -S renameat -S
truncate -S chmod -S setxattr -S lsetxattr -S removexattr -S
lremovexattr -F exit=-EPERM -k file_modification
```

```
 auditctl -a always,exit -F arch=b64 -S rename -S renameat -S
truncate -S chmod -S setxattr -S lsetxattr -S removexattr -S
lremovexattr -F exit=-EPERM -k file_modification
```

Отслеживание события использования 32-битного API для 64-битной системы:

```
 ### Это правило обнаружит любое использование 32-битных системных
вызовов, потому как это может быть
 ### признаком использования уязвимости 32-битного API.
 auditctl -a always,exit -F arch=b32 -S all -k 32bit_api
```

Утилита **aureport** - это инструмент, который генерирует итоговые отчеты на основе логов службы аудита. **aureport** может также принимать данные со стандартного ввода (*stdin*) до тех пор, пока на входе будут необработанные данные логов. В шапке каждого отчета для каждого столбца есть заголовок - это облегчает понимание данных. Все отчеты, кроме основного итогового отчета, содержат номера событий аудита. Используя их, вы можете найти полные данные о событии с помощью **ausearch -a** . Если в отчете слишком много данных, можно задать время начала и время окончания для уточнения временного промежутка. Отчеты, генерируемые **aureport**, могут быть использованы как исходный материал для получения более развернутых отчетов.

Опции:

Опция	Значение опции
-au, --auth	Отчет обо всех попытках аутентификации.
-a, --avc	Отчет обо всех avc сообщениях.
-c, --config	Отчет об изменениях конфигурации.
-cr, --crypto	Отчет о событиях, связанных с шифрованием.
-e, --event	Отчет о событиях.
-f, --file	Отчет о файлах.
--failed	Для обработки в отчетах выбирать только неудачные события. По умолчанию показываются и удачные, и неудачные события.
-h, --host	Отчет о хостах.
-i, --interpret	Транслировать числовые значения в текстовые. Например, идентификатор пользователя будет оттранслирован в имя пользователя. Трансляция выполняется с использованием данных с той машины, где запущен aureport. Т.е. если вы переименовали учетные записи пользователей или не имеете таких же учетных записей на вашей машине, то вы можете получить результаты, вводящие в заблуждение.
-if, --input файл	Использовать указанный файл вместо логов аудита. Это может быть полезно при анализе логов с другой машины или при анализе частично

Опция	Значение опции
	сохраненных логов.
-l, --login	Отчет о попытках входа в систему.
-m, --mods	Отчет об изменениях пользовательских учетных записей.
-ma, --mac	Отчет о событиях в системе, обеспечивающей мандатное управление доступом - Mandatory Access Control (MAC).
-p, --pid	Отчет о процессах.
-r, --response	Отчет о реакциях на аномальные события.
-s, --syscall	Отчеты о системных вызовах.
--success	Для обработки в отчетах выбирать только удачные события. По умолчанию показываются и удачные, и неудачные события.
--summary	Генерировать итоговый отчет, который дает информацию только о количестве элементов в том или ином отчете. Такой режим есть не у всех отчетов.
-t, --log	Этот параметр генерирует отчет о временных рамках каждого отчета.
-te, --end	Искать события, которые произошли раньше (или во время) указанной временной точки. Формат даты и времени зависит от ваших региональных настроек. Если дата не указана, то подразумевается текущий день (today). Если не указано время, то подразумевается текущий момент (now). Используйте 24-часовую нотацию времени, а не AM/PM. Например, дата может быть задана как 10/24/2005, а время - как 18:00:00. Вы можете также использовать ключевые слова: now, recent, today, yesterday, this-week, this-month, this-year. today означает первую секунду после полуночи текущего дня. recent - 10 минут назад. yesterday - первую секунду после полуночи предыдущего дня. this-week означает первую секунду после полуночи первого дня текущей недели, первый день недели определяется из ваших региональных настроек. this-month означает первую секунду после полуночи первого числа текущего месяца. this-year означает первую секунду после полуночи первого числа первого месяца текущего года.
-tm, --terminal	Отчет о терминалах.
-ts, --start	Искать события, которые произошли после (или во время) указанной временной точки. Формат даты и времени зависит от ваших региональных настроек. Если дата не указана, то подразумевается текущий день (today). Если не указано время, то подразумевается полночь (midnight). Используйте 24-часовую нотацию времени, а не AM/PM. Например, дата может быть задана как 10/24/2005, а время - как 18:00:00. Вы можете также использовать ключевые слова: now, recent, today, yesterday, this-week, this-month, this-year. today означает первую секунду после полуночи текущего дня. recent - 10 минут назад. yesterday - первую секунду после полуночи предыдущего дня. this-week означает первую секунду после полуночи первого дня текущей недели, первый день недели определяется из ваших региональных настроек. this-month

Опция	Значение опции
	означает первую секунду после полуночи первого числа текущего месяца. this-year означает первую секунду после полуночи первого числа первого месяца текущего года.
-u, --user	Отчет о пользователях.
-v, --version	Вывести версию программы и выйти.

Программа **ausearch** является инструментом поиска по журналу аудита. Утилита **ausearch** может также принимать данные со стандартного ввода (*stdin*) до тех пор, пока на входе будут необработанные данные логов. Все условия, указанные в параметрах, объединяются логическим «**И**». К примеру, при указании **- m** и **-ui** в качестве параметров будут показаны события, соответствующие заданному типу и идентификатору пользователя.

Стоит отметить, что каждый системный вызов ядра из пользовательского пространства и возвращение данных в пользовательское пространство имеет один уникальный (для каждого системного вызова) идентификатор события.

Различные части ядра могут добавлять дополнительные записи. Например, в событие аудита для системного вызова «**open**» добавляется запись *PATH* с именем файла. **ausearch** показывает все записи события вместе. Это означает, что при запросе определенных записей результат может содержать записи *SYSCALL*.

Также помните, что не все типы записей содержат указанную информацию. Например, запись *PATH* не содержит имя узла или *loginuid*.

Опции:

Опция	Значение опции
-a, --event audit-event-id	Искать события с заданным идентификатором события. Сообщения обычно начинаются примерно так: msg=audit(1116360555.329:2401771). Идентификатор события - это число после «:». Все события аудита, связанные с одним системным вызовом имеют одинаковый идентификатор.
-c, --comm comm-name	Искать события с заданным comm name. comm name - имя исполняемого файла задачи.
-f, --file file-name	Искать события с заданным именем файла.
-ga, --gid-all all-group-id	Искать события с заданным эффективным или обычным идентификатором группы.
-ge, --gid-effective effective-group-id	Искать события с заданным эффективным идентификатором группы или именем группы.
-gi, --gidgroup-id	Искать события с заданным идентификатором группы или именем группы.
-h, --help	Справка.
-hn, --host host-name	Транслировать числовые значения в текстовые. Например,

Опция	Значение опции
	идентификатор пользователя будет оттранслирован в имя пользователя. Трансляция выполняется с использованием данных с той машины, где запущен ausearch. Т.е. если вы переименовали учетные записи пользователей или не имеете таких же учетных записей на вашей машине, то вы можете получить результаты, вводящие в заблуждение.
-if, --input file-name	Использовать указанный файл вместо логов аудита. Это может быть полезно при анализе логов с другой машины или при анализе частично сохраненных логов.
-k, --key key-string	Искать события с заданным ключевым словом.
-m, --message message-type | comma-sep-message-type-list	Искать события с заданным типом. Вы можете указать список значений, разделенных запятыми. Можно указать несуществующий в событиях тип ALL, который позволяет получить все сообщения системы аудита. Список допустимых типов большой и будет показан, если указать эту опцию без значения. Тип сообщения может быть строкой или числом. В списке значений этого параметра в качестве разделителя используются запятые и пробелы недопустимы.
-o, --object SE-Linux-context-string	Искать события с заданным контекстом (объектом).
-p, --pid process-id	Искать события с заданным идентификатором процесса.
-pp, --ppid parent-process-id	Искать события с заданным идентификатором родительского процесса.
-r, --raw	Необработанный вывод. Используется для извлечения записей для дальнейшего анализа.
-sc, --success syscall-name-or-value	Искать события с заданным системным вызовом. Вы можете указать его номер или имя. Если вы указали имя, оно будет проверено на машине, где запущен ausearch.
-se, --context SE-Linux-context-string	Искать события с заданным контекстом SELinux (stcontext/subject или tcontext/object).
-su, --subject SE-Linux-context-string	Искать события с заданным контекстом SELinux - scontext (subject).
-sv, --success success-value	Искать события с заданным флагом успешного выполнения. Допустимые значения: yes (успешно) и no(неудачно).
-te, --end [end-date] [end-time]	Искать события, которые произошли раньше (или во время) указанной временной точки. Формат даты и времени зависит от ваших региональных настроек. Если дата не указана, то подразумевается текущий день (today). Если не указано время, то подразумевается текущий момент (now). Используйте 24-часовую нотацию времени, а не AM/PM. Например, дата может быть задана как 10/24/2005, а время - как 18:00:00. Вы можете также использовать ключевые слова: now, recent, today, yesterday, this-week, this-month, this-year. today означает

Опция	Значение опции
	первую секунду после полуночи текущего дня. recent - 10 минут назад. yesterday - первую секунду после полуночи предыдущего дня. this-week означает первую секунду после полуночи первого дня текущей недели, первый день недели определяется из ваших региональных настроек (см. localtime). this-month означает первую секунду после полуночи первого числа текущего месяца. this-year означает первую секунду после полуночи первого числа первого месяца текущего года.
-ts, --start [start-date] [start-time]	Искать события, которые произошли после (или во время) указанной временной точки. Формат даты и времени зависит от ваших региональных настроек. Если дата не указана, то подразумевается текущий день (today). Если не указано время, то подразумевается полночь (midnight). Используйте 24-часовую нотацию времени, а не AM/PM. Например, дата может быть задана как 10/24/2005, а время - как 18:00:00. Вы можете также использовать ключевые слова: now, recent, today, yesterday, this-week, this-month, this-year. today означает первую секунду после полуночи текущего дня. recent - 10 минут назад. yesterday - первую секунду после полуночи предыдущего дня. this-week означает первую секунду после полуночи первого дня текущей недели, первый день недели определяется из ваших региональных настроек. this-month означает первую секунду после полуночи первого числа текущего месяца. this-year означает первую секунду после полуночи первого числа первого месяца текущего года.
-tm, --terminal terminal	Искать события с заданным терминалом. Некоторые службы (такие как cron и atd) используют имя службы как имя терминала.
-ua, --uid-all all-user-id	Искать события, у которых любой из идентификатора пользователя, эффективного идентификатора пользователя или loginuid (auid) совпадают с заданным идентификатором пользователя.
-ue, --uid-effective effective-user-id	Искать события с заданным эффективным идентификатором пользователя.
-ui, --uid user-id	Искать события с заданным идентификатором пользователя.
-ul, --loginuid login-id	Искать события с заданным идентификатором пользователя. Все программы, которые его используют, должны использовать pam_loginuid.
-v, --verbose	Показать версию и выйти.
-w, --word	Совпадение с полным словом. Поддерживается для имени файла, имени узла, терминала и контекста SELinux.
-x, --executable executable	Искать события с заданным именем исполняемой программы.

Утилита **autrace** - это программа, которая добавляет правила аудита для

того, чтобы следить за использованием системных вызовов в указанном процессе подобно тому, как это делает **strace**. После добавления правил она запускает процесс с указанными аргументами. Результаты аудита будут либо в логах аудита (если служба аудита запущена), либо в системных логах. Внутри **autrace** устроена так, что удаляет все предыдущие правила аудита, перед тем как запустить указанный процесс и после его завершения. Поэтому, в качестве дополнительной меры предосторожности, программа не запустится, если перед ее использованием правила не будут удалены с помощью **audtictl** - предупреждающее сообщение известит об этом.

Опции:

Опция	Значение опции
-r	Ограничить сбор информации о системных вызовах только теми, которые необходимы для анализа использования ресурсов. Это может быть полезно при моделировании внештатных ситуаций, к тому же позволяет уменьшить нагрузку на логи.

Примеры:

Ниже представлен пример обычного использования программы:

```
autrace /bin/ls /tmp
ausearch --start recent -p 2442 -i
```

Еще один пример для режима ограниченного сбора информации:

```
autrace -r /bin/ls
ausearch --start recent -p 2450 --raw | aureport --file –summary
ausearch --start recent -p 2450 --raw | aureport --host –summary
```

8.2. Служба rsyslog.

В GNU/Linux принято сохранять информацию разной степени детализации о процессе работы программ в специальных текстовых файлах, называемых журналами.

Стандартное место расположения журналов - это каталог `/var/log` .

Все журнальные файлы принадлежат к одной из двух категорий: системные журналы и журналы прикладных программ.

Служба syslog (rsyslog) предназначена для обеспечения сохранения информации, поступающей от различных системных служб.

Некоторые службы ведут собственные журналы, которые не зависят от syslog, например веб сервер Apache.

Для таких служб принято создавать отдельные подкаталоги в /var/log.

Примечание: Она может быть также использована и прикладными программами, однако чаще всего прикладные программы, если уж возникает такая необходимость, ведут собственные журналы без службы syslog . При этом файлы журналов для таких программ обычно находятся где-либо в подкаталогах `/var/log` .

Примечание: Не все журналы, находящиеся в `/var/log` *, обслуживаются службой syslog. Так, например, бинарный файл базы данных с информацией о последних входах в сеанс wtmp также находится в этом каталоге (имеется аналогичный файл* `/var/run/utmp` *с информацией о пользователях, находящихся в сеансе – см. команды* `who` *и* `last`*).*

Syslog это не только служба для регистрации событий, но и протокол для передачи журнальных сообщений.

Согласно протоколу syslog сообщения характеризуются двумя параметрами:

facility — источник события;

severity — важность события.

Определены следующие источники событий:

LOG_AUTH сообщения безопасности/авторизации

LOG_AUTHPRIV сообщения безопасности/авторизации (private)

LOG_CRON планировщик заданий (cron и at)

LOG_DAEMON системные службы без определенного значения объекта

LOG_FTP служба FTP

LOG_KERN сообщения ядра (не могут быть созданы пользовательскими процессами)

с LOG_LOCAL0 по LOG_LOCAL7 зарезервировано для локального использования

LOG_LPR подсистема печати

LOG_MAIL почтовая подсистема

LOG_NEWS подсистема новостей USENET

LOG_SYSLOG сообщения, сгенерированные самой службой syslog

LOG_USER (по умолчанию) обычные сообщения пользовательского уровня

LOG_UUCP подсистема UUCP

В других операционных системах могут быть определены и другие источники событий.

В качестве уровня событий используются:

LOG_EMERG система в нерабочем состоянии

LOG_ALERT необходимо срочное вмешательство

LOG_CRIT критические состояния
LOG_ERR ошибки
LOG_WARNING предупреждения
LOG_NOTICE обычные, но важные сообщения
LOG_INFO информационные сообщения
LOG_DEBUG сообщения уровня отладки
Уровни указаны в порядке убывания приоритета.

Служба syslog в большинстве современных Linux представлена демоном `rsyslogd` (The rocket-fast Syslog Server), запускаемым при старте операционной системы автоматически и работающем в фоновом режиме:

Пример:

```
$ ps -C rsyslogd
    PID TTY          TIME CMD
    862 ?        00:00:29 rsyslogd
```

Реже используется syslog-ng. Где-то может встретиться и другие реализации syslog.

Задачей демона `rsyslogd` является сбор сообщений от системных служб и сохранение этих сообщений в заранее известных файлах журналов.

Примечание: В GNU/Linux обычно сообщения, поступающие от различных служб, не записываются в один единственный журнал. Наоборот, принято называть файлы журналов так, чтобы по их названию можно было судить об источнике сообщений.

Конфигурационным файлом демона `rsyslogd` является `/etc/rsyslog.conf` .

Строки этого файла, начинающиеся с решетки `#` являются комментариями.

Проект rsyslog относительно молодой (начат в 2004 г.), поэтому в разных версиях rsyslog применяются разные форматы конфигурации.

Пример: проверим версию rsyslog:

```
$ rsyslogd -v | head -1
rsyslogd  8.2010.0 (aka 2020.10) compiled with:
```

Rsyslog модульная система, это означает, что вам необходимо для начала определить какие модули вы будете использовать.

Модули делятся на две категории:

Входящие (Input) с префиксом im в названии. Такие модули открывают

канал поступления информации.

Исходящие (Output) — om. Эти модули позволяют передавать сообщения в разные каналы.

Парсер (Parser) — pm. Разбирают содержимое сообщение на части.

Модификаторы сообщений (Message Modification) — mm. Используются для изменения содержимого сообщения.

Генераторы строки (String Generator) — sm. Генерируют строки на основе содержимого сообщения.

Библиотечные модули (Library). Часть самого rsyslog загружаются автоматически.

Пример:

```
$ grep module /etc/rsyslog.conf
module(load="builtin:omfile"
Template="RSYSLOG_TraditionalFileFormat")
module(load="imuxsock"      # provides support for local system
logging (e.g. via logger command)
module(load="imjournal"         # provides access to the systemd
journal
#module(load="imklog") # reads kernel messages (the same are read
from journald)
#module(load="immark") # provides --MARK-- message capability
#module(load="imudp") # needs to be done just once
#module(load="imtcp") # needs to be done just once
```

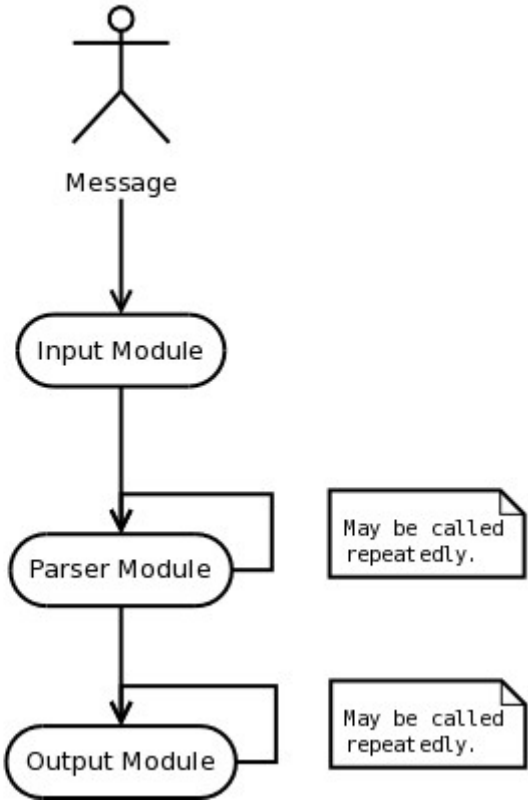


Рисунок 2: Обработка сообщений

Структура строк, каждая из которых направляет некоторый поток сообщений в заданный файл (или на удаленный компьютер - сервер ведения журналов), представлена двумя полями:

Определение сообщения (selector) - поле, в котором указывается от каких классов программ должны собираться сообщения в данный поток. И какие именно сообщения.

Поле действия (action), указывающее куда должен быть записан поток сообщений. Чаще всего - это имя файла журнала в `/var/log` .

Пример:

```
$ grep '\/var\/log\/' /etc/rsyslog.conf
*.info;mail.none;authpriv.none;cron.none
/var/log/messages
authpriv.*
/var/log/secure
mail.*
-/var/log/maillog
cron.*
/var/log/cron
uucp,news.crit
/var/log/spooler
```

```
local7.*
/var/log/boot.log
```

Примечание: Из приведенного выше примера заметно, что поле определения сообщения состоит из двух частей, разделенных точкой. Первая часть - источник сообщения (facility), а вторая - уровень важности (priority) сообщения. Эти две части уникально определяют все возможные сообщения, обрабатываемые syslog .

Знак звездочка является метасимволом, обозначающим либо все источники, если он указан перед точкой – разделителем, либо все уровни важности, если этот метасимвол установлен после точки.

При указании источника сообщения и уровня важности, разделенных точкой, определяется, что сообщения, поступающие от этого источника и имеющие указанный и вышележащие уровни важности, будут записаны в данный канал.

Пример:

```
uucp,news.crit
/var/log/spooler
```

При необходимости запретить запись в какой-либо канал сообщений с заданным уровнем важности и выше, можно использовать знак восклицания перед уровнем важности.

Пример:

```
daemon.info;daemon.!err        -/var/log/daemons
```

Примечание: В этом случае все сообщения от демонов с уровнями важности от `info` *до* `warning` *будут записываться в журнал* `/var/log/daemons` *, а сообщения с уровнями важности, начиная с* `err` *, записаны туда не будут.*

Если же необходимо записывать в журнал сообщения только с определенным уровнем важности и ни с какими другими более, то перед требуемым уровнем важности следует поставить знак равно.

Пример:

```
daemon.=err                    /var/log/daemons.err
```

Примечание: В файл `/var/log/daemons.err` *будут записываться только сообщения об ошибках в работе демонов.*

Для исключения из потока сообщений те из них, которые имеют заданный уровень важности используют восклицательный знак и знак равенства.

Пример:

```
kern.*;kern.!=info              /var/log/kernel
```

Примечание: Здесь в журнал будут записываться все сообщения от ядра, кроме информационных.

Если в канал не должны быть записаны любые сообщения от каких-либо источников, то удобно использовать директиву `none` :

Пример:

```
*.crit;lpr,cron,mail.none      /var/log/critical
```

Примечание: При этом в файл `/var/log/critical` *будут записываться сообщения от всех источников о критических и более важных событиях, кроме любых сообщений от служб печати, почты,* `at` *и* `cron` .

Rsyslog поддерживает и другие фильтры записи событий в журналы.

Property-Based Filters — фильтр, который анализирует различные свойства сообщений.

Expression-Based Filters — фильтр, в котором вы можете определять различные логические выражения всестороннего для анализа сообщений.

Правила фильтрации сообщений могут быть объединены в наборы правил (RuleSet). Наборы правил помогают более эффективно управлять потоком информации.

Для составления сообщений в правильном формате или для формирования имен файлов вы можете использовать шаблоны (Templates)

Пример: разделение локальных сообщений и сообщений из сети, причем имена файлов куда записываются сообщения формируются на основе имени узла, которое это сообщение прислало.

```
# cat /etc/rsyslog.conf | egrep -v '^(#|$)'
$ModLoad imuxsock # provides support for local system logging
(e.g. via logger command)
$ModLoad imklog    # provides kernel logging support (previously
done by rklogd)
$ModLoad imudp
```

```
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$RuleSet local
kern.*
/dev/console
*.info;mail.none;authpriv.none;cron.none
/var/log/messages
authpriv.*
/var/log/secure
mail.*
-/var/log/maillog
cron.*
/var/log/cron
*.emerg                                                    *
uucp,news.crit
/var/log/spooler
local7.*
/var/log/boot.log
$DefaultRuleset local
$template RemHost,"/var/log/network/%HOSTNAME%.log"
$RuleSet remote_udp
*.* ?RemHost
$InputUDPServerBindRuleset remote_udp
$UDPServerRun 514
```

8.3. Служба ротации журналов.

С течением времени накапливающиеся сообщения в файлах журналов могут переполнить файловую систему.

Для предотвращения этого предназначена программа `logrotate` , обеспечивающая ротацию журналов. Стандартный путь ее вызова – использование ее, как ежедневного задания `crond` .

Пример:

```
$ cat /etc/cron.daily/logrotate
#!/bin/sh
exec /usr/sbin/logrotate /etc/logrotate.conf
```

Примечание: В данном примере показано, что скрипт вызова команды `logrotate` *находится в каталоге* `/etc/cron.daily` *, задания в котором выполняются каждую ночь (это настройки данной системы). Файл* `/etc/logrotate.conf` *содержит настройки для этой утилиты.*

Утилита `logrotate` способна производить следующие действия с файлами журналов:

Удалять.

Переименовывать.

Сжимать с помощью программ – компрессоров.

Создавать новые пустые файлы журналов.

Посылать ротируемые файлы журналов по электронной почте.

Пример: Ротации с заданной периодичностью файла `/var/log/messages` **:**

```
# ls -w 1 /var/log/messages*
/var/log/messages
/var/log/messages.1.bz2
/var/log/messages.2.bz2
/var/log/messages.3.bz2
/var/log/messages.4.bz2
/var/log/messages.5.bz2
```

Примечание: При наступлении момента времени, когда необходимо осуществить первую ротацию файл `messages` *переименовывается в* `messages.1.bz2` *(в данном примере используется компрессия журналов утилитой* `bzip2` *).*

При второй ротации файл `messages.1.bz2` *переименовывается в* `messages.2.bz2` *, а файл* `messages` *переименовывается в* `messages.1.bz2` .

При третьей ротации файл `messages.2.bz2` *переименовывается в* `messages.3.bz2` *и так далее …*

Утилита `logrotate` *удаляет архивные копии старых журналов по достижении заданного количества копий. В этом примере ротация первой архивной копии журнала (файл* `messages.1.bz2` *) осуществляется четыре раза (до* `messages.5.bz2` *).*

Файла конфигурации `logrotate` - `/etc/logrotate.conf` .

Настройки, находящиеся в начале файла `/etc/logrotate.conf` и не связанные с именами файлов журналов, являются глобальными.

Для каждого конкретного файла журнала можно указывать отдельные настройки.

Обычно применяются следующие настройки:

`daily`, `weekly` и `monthly` определяют периодичность ротации равной, соответственно, одному дню, неделе или месяцу.

Настройка `rotate` определяет количество ротаций первой архивной копии журнала до ее удаления.

`create` заставляет создавать пустой файл журнала после его ротации.

Причем, настройка `create` позволяет указывать права доступа и владения создаваемых журнальных файлов.

`compress` сжатие ротированных файлов. Если не надо сжимать файла архивных копий, то следует использовать настройку `nocompress` .

`copy` копировать файлы оставляя при этом оригинальные файлы журналов нетронутыми.

`notifempty` позволяет не осуществлять ротацию пустых файлов.

`include` позволяет включать в файл конфигурации дополнительные настройки, указанные в файле – аргументе этой директивы. Если аргументом является каталог, то в основной файл конфигурации включается содержимое всех конфигурационных файлов, находящихся в этом каталоге.

`mail` позволяет получать копии ротируемых журнальных файлов по электронной почте.

`prerotate` и `postrotate` позволяют указывать скрипты, которые будут исполнены, соответственно, до и после ротации.

`size` можно указывать размер файла журнала, по превышении которого должна осуществляться его ротация.

Пример:

```
weekly
rotate 4
create
compress
notifempty
include /etc/logrotate.d
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 4
}
```

8.4. Журналы systemd

Система systemd собирает сведения о своем функционировании в бинарные файлы.

Файл /etc/systemd/journald.conf определяет параметры журналирования.

Важнейшая опция Storage в этом файле определяет способ ведения журнала постоянный (persistent), непостоянный (volatile) или никакой (none).

Команда journalctl --header показывает в том числе название файла в

который в данный момент записывается информация.

Каждый раз при старте системы создается новый журнал.

Команда journalctl выводит текущий журнал с начала с самого раннего события.

Полезные опции:

-e — показывает журнал с конца.

-x — показывает объяснения событий, если они есть.

-u юнит — показывать события связанные с юнитом.

-f – показывать вновь поступающие сообщения (аналогично tail -f).

Глава 9. Сеть и сетевые сервисы в РЕД ОС

9.1. Настройка сети

Для просмотра сетевых настроек нужно ввести команду:

ifconfig

или воспользоваться командой ip с параметром addr:

ip addr

Тут можно увидеть параметры и название сетевой карты.

```
ip addr

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP group default qlen 1000
link/ether be:cd:c8:c8:7a:60 brd ff:ff:ff:ff:ff:ff
inet 10.10.10.83/24 brd 10.10.10.255 scope global dynamic
noprefixroute enp0s3
valid_lft 435sec preferred_lft 435sec
inet6 fe80::f42e:4ae5:4dc0:ff68/64 scope link noprefixroute
valid_lft forever preferred_lft forever+
```

По умолчанию сетевой адаптер настроен для получения ip адреса по dhcp. Настройки хранятся в файле /etc/sysconfig/network-scripts/ifcfg-<имя сетевого интерфейса>. Именуются они например так enp0s3 или enp0s4. В данном примере сетевой интерфейс имеет имя enp0s3, используя команду cat можно посмотреть содержимое файла настроек:

```
cat /etc/sysconfig/network-scripts/ifcfg-enp0s3
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
```

```
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp0s3
UUID=70708f9c-d298-43d4-a2e3-503354afcc29
DEVICE=enp0s3
ONBOOT=yes
```

Для редактирования файла настроек можно использовать любой текстовый редактор (например **mcedit** или **vi**). Для установки статического IP адреса нам необходимо на строчке **BOOTPROTO** установить **BOOTPROTO=none** и дописать:

Указать ДНС:

```
DNS1=8.8.8.8
```

Указать IP:

```
IPADDR0=172.16.0.30
```

Указать нужную маску:

```
PREFIX0=24
```

И шлюз по умолчанию:

```
GATEWAY0=172.16.0.1
```

Для того, чтобы сетевая карта активировалась при запуске ОС, необходимо в этом файле найти параметр **ONBOOT** и задать значение **yes**.

Для немедленного применения изменений перезапустим сеть:

```
systemctl restart NetworkManager
```

В файле настройки сетевой карты можно добавить столько DNS серверов, сколько требуется. Например:

```
DNS1=172.16.0.1 DNS2=8.8.8.8 DNS3=8.8.4.4
```

В работе используются максимум 3 DNS - сервера (те, что указаны первыми).

Проверить шлюз по умолчанию установленный в системе:

```
netstat -nr
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 172.16.0.1 0.0.0.0 UG 0 0 0 eno16777736
172.16.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eno16777736
```

Строка с Destination 0.0.0.0 определяет адрес шлюза. Если у вас ее нет, либо в поле Gateway установлен неверный шлюз, то можно это изменить. Устанавливаем шлюз по-умолчанию:

```
route add default gw 172.16.0.1
```

Если вы не сменили при установке ОС имя сервера или вы хотите его изменить, то сделать это можно следующим образом. Для начала проверим, какой hostname у нас установлен:

```
hostname
```

Допустим мы хотим изменить имя на `server.work`, для этого необходимо выполнить команду:

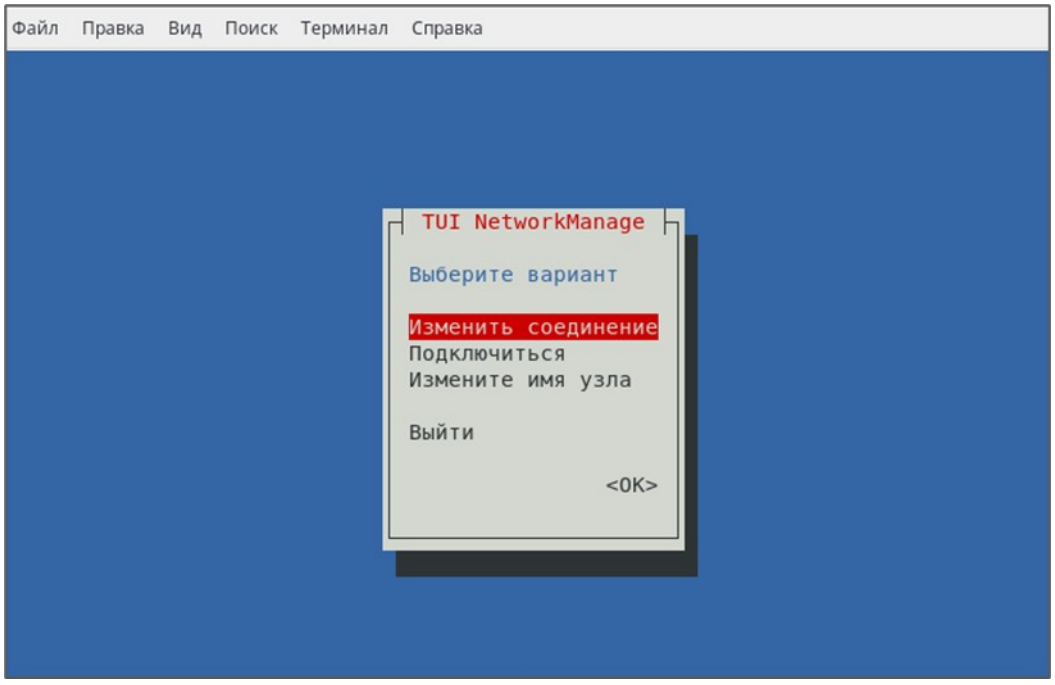
```
hostnamectl set-hostname server.work
```

Для смены имени ПК перезагрузка не требуется.

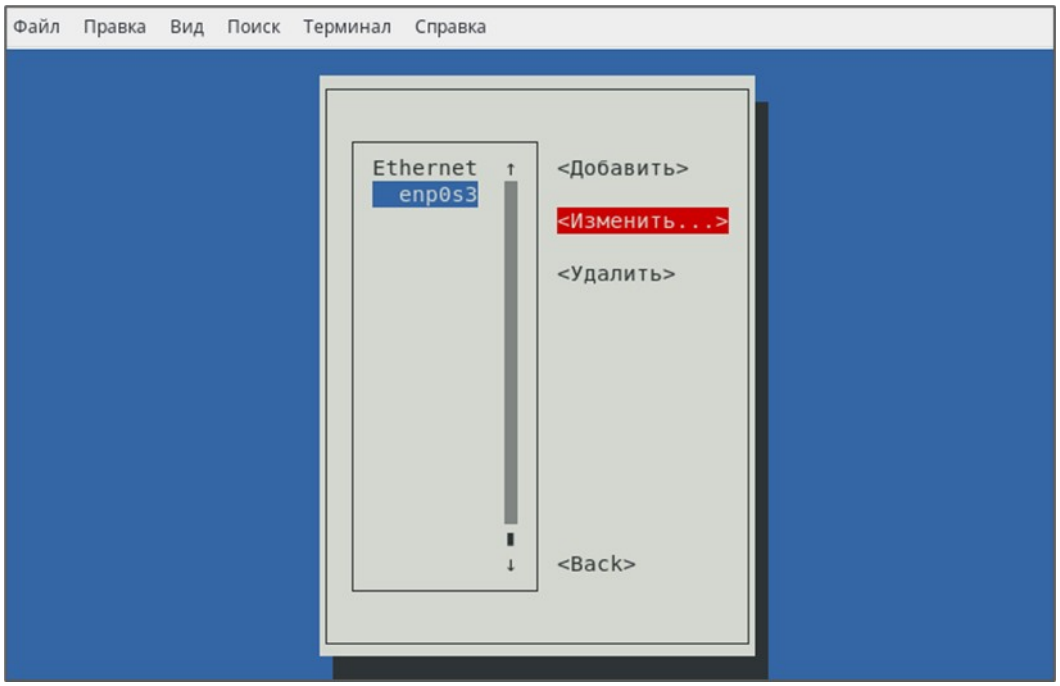
Выполнить настройку сети можно разными способами. Наиболее популярным является способ настройки сети с помощью Network Manager. С его помощью можно выполнить настройки как в графической сессии, так и при помощи специальных утилит, например nmtui и nmcli.

nmtui - утилита для настройки сети с интуитивно-понятным графическим интерфейсом, для ее запуска в терминале необходимо выполнить:

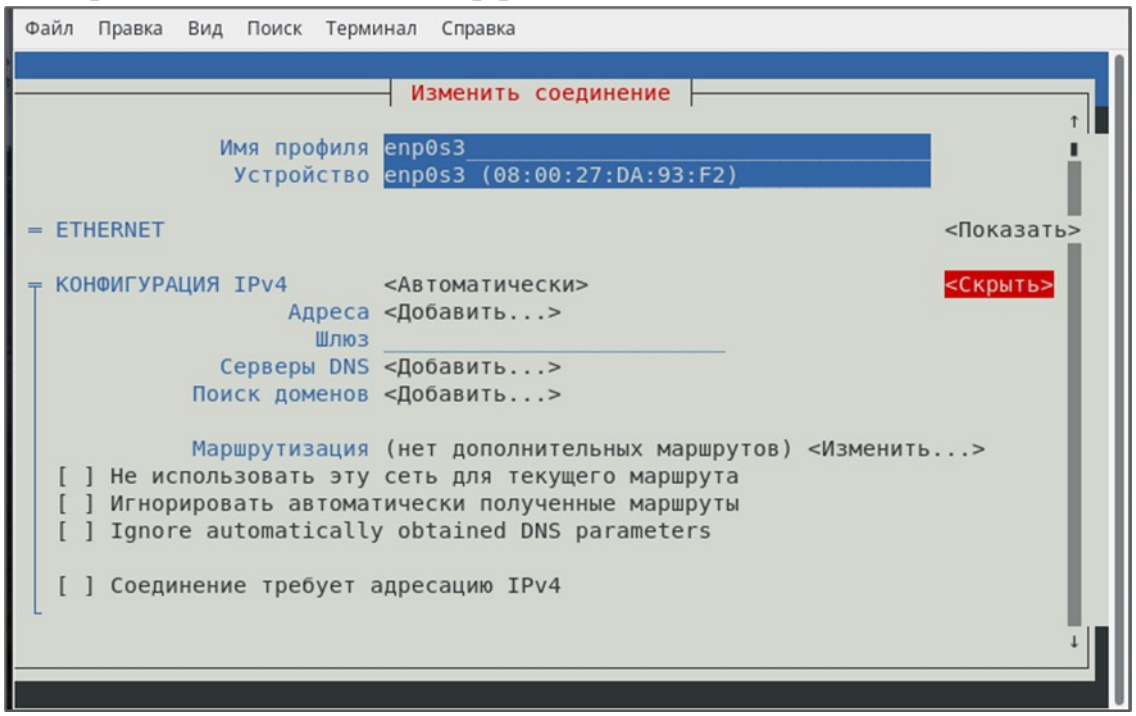
```
nmtui
```



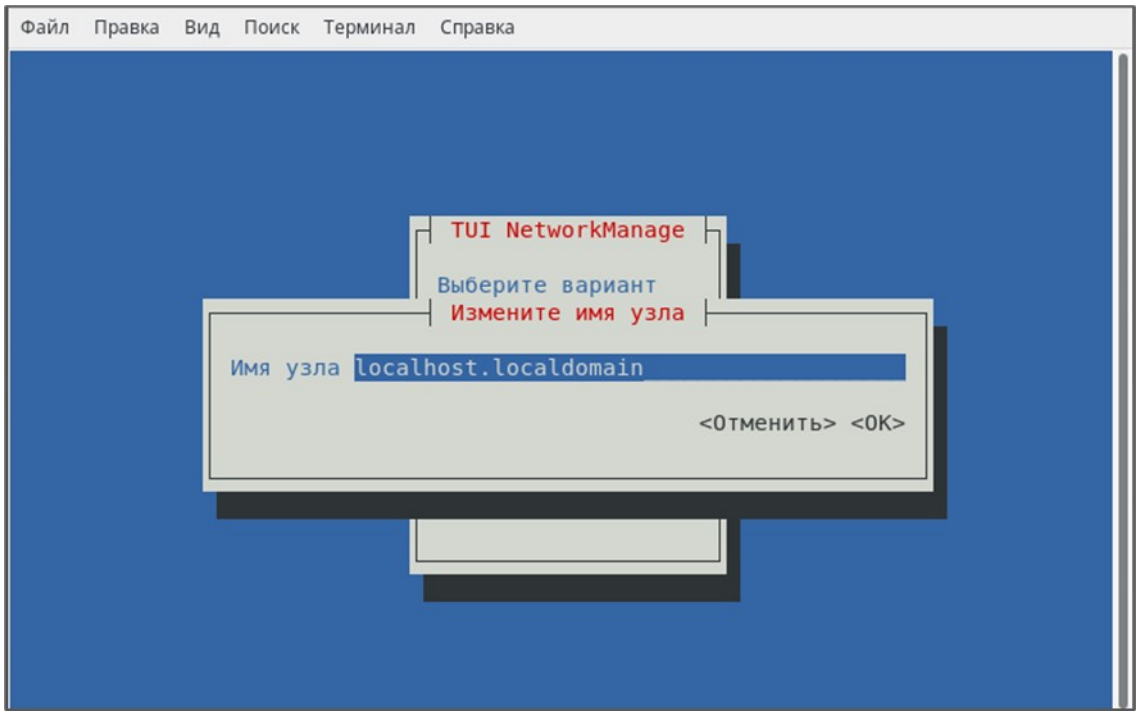
Выберите "Изменить соединение" клавишей **Enter**.



Выберите необходимый интерфейс



Выберите изменений



nmcli - это инструмент командной строки для управления NetworkManager и создания отчетов о состоянии сети. Используется для создания, отображения, редактирования, удаления, активации и деактивации сетевых подключений, а также для контроля и отображения состояния сетевых устройств.

Чтобы узнать название интерфейса введите в командной строке:

```
nmcli con
```

В нашем случае интерфейс будет называться eth0.

```
nmcli con

NAME UUID TYPE DEVICE
eth0 3a2830fe-b8dd-45bd-b33e-387db796c8da ethernet eth0
```

Для того, чтобы указать, что настройки должны получаться от DHCP используется команда:

```
nmcli con modify eth0 ipv4.method auto
```

Для использования ручных настроек необходимо ввести:

```
nmcli con modify eth0 ipv4.method static
```

Для того, чтобы указать статический адрес, используется команда:

```
nmcli con modify eth0 ipv4.address 192.168.0.1/24
```

Для того, чтобы указать dns, используется команда:

```
nmcli connection modify eth0 ipv4.dns 202.131.124.4
```

Для того, чтобы указать шлюз, используется команда:

```
$ nmcli connection modify eth0 IPv4.gateway 192.168.0.1
```

Чтобы настроить статический маршрут для существующего соединения Ethernet, используется команда:

```
nmcli connection modify eth0 +ipv4.routes "192.168.122.0/24
10.10.10.1"
```

Для остановки сетевой службы и её запуска доступны команды `nmcli networking off` и `nmcli networking on`.

Любое сделанное изменение так же требует перезагрузки интерфейса для его применения.

9.2. Netfilter(iptables)

Пакетный фильтр в ядрах Linux 2.4 называется Netfilter, а утилита для его настройки – iptables (домашняя страница в Интернете http://netfilter.org).

Netfilter – первый пакетный фильтр для Linux, который относится к классу фильтров с проверкой состояния (stateful filter)

Примечание: фильтры данного класса являются гибридом шлюза приложений и пакетного фильтра. Такие фильтры анализируют содержимое пакетов, хотя и не так подробно, как шлюзы приложений.

Netfilter хранит информацию о всех соединениях в памяти, позволяет обнаруживать и блокировать сканирование, DoS атаки.

Netfilter – сложнее и надежнее, чем его предшественник Ipchains

Поддержка Netfilter включается при компиляции ядра: для ядер 2.4.x - Networking options -> IP: Netfilter Configuration для ядер 2.6.x - Device drivers -> Networking support -> Networking options -> -> Network packet filtering (replaces ipchains)

Команда iptables добавляет, удаляет или изменяет правила фильтрации пакетов, записанные в специальной таблице ядра. Данные этой таблицы сбрасываются при любой перезагрузке ОС.

Команда iptables позволяет использовать (с некоторыми изменениями) синтаксис утилиты ipfwadm (для ядер Linux 2.0) и ipchains (для ядер 2.2).

Для обеспечения сохранения правил фильтрации в файлах предназначены

утилиты iptables-save и iptables-restore, аналогичные утилитам для ядер 2.2: ipchains-save и ipchains-restore.

Netfilter – часть пакетного фильтра, находящаяся в ядре ОС GNU/Linux, iptables – пользовательская утилита (пользовательская часть пакетного фильтра)

Netfilter поддерживает три вида таблиц для хранения правил, с которыми сверяются все сетевые пакеты:

filter – используется для фильтрации пакетов

nat – отвечает за таблицу трансляции сетевых адресов

mangle – используется для изменения битов качества обслуживания (QoS) в заголовках TCP (применяется редко)

raw — исключения из правил отслеживающих соединения

security — применение MAC (Mandatory Access Control) для сетевых соединений, применяется после таблицы filter

Опция -t (--table) позволяет указать таблицу, с которой будет работать утилита iptables.

-t filter – используется таблица фильтрации (по умолчанию);

-t nat – используется таблица сетевых трансляции адресов

-t mangle - используется таблица специализированных модификаций пакетов и т. д.

Правила фильтрации пакетов записываются в виде цепочек (как и в IPchains)

Каждое правило заканчивается действием (target, целью), обычно, ACCEPT (пропустить) или DROP (отбросить)

Заранее определено три цепочки правил: INPUT, OUTPUT и FORWARD для, соответственно, обработки входящих, исходящих и перенаправленных пакетов.

Если пакет прошел всю цепочку правил, не встретив ни одного правила, которому он удовлетворяет, он подвергается обработке в соответствии с установленной политикой (policy) для данной цепочки : пропустить - ACCEPT или отклонить - DROP Примечание: текущую политику можно определить, выполнив команду iptables -L

Когда появляется входящий пакет, он сначала проверяется на предмет его назначения. Эта процедура называется маршрутизацией (routing).

Если данный пакет предназначен для данного хоста, то он попадает на начало цепочки INPUT, и в случае ее успешного прохождения передается

ожидающему его локальному процессу.

В случае, если пакет предназначается куда-либо на другой интерфейс, он передается (при разрешенном форвардинге в ядре /proc/sys/net/ipv4/ip_forward = 1) на вход цепи FORWARD.

Если форвардинг в ядре не включен, то пакет отклоняется (DROP).

Если пакет принимается (ACCEPT) правилом в цепочке FORWARD или политика это допускает, то он передается на заданный интерфейс.

Программа, работающая на компьютере, может посылать пакеты. Они попадают на вход цепочки OUTPUT. Если пакеты проходят эту цепь, то они покидают систему через заданный интерфейс.

Для манипуляции цепочками правил можно применять следующие опции команды iptables:

-N (--new-chain) - создание новой цепочки;

-X (--delete-chain) - удаление пустой цепочки;

-P (--policy) - установка политики для встроенной цепочки;

-L (--list) - получение списка правил в цепочке;

-F (--flush) - очистка цепочки (сброс правил);

-Z (--zero) - обнуление счетчиков байтов и пакетов во всех правилах цепочки.

Следующие опции команды iptables позволяют управлять правилами в цепочке:

-A (--append) - добавление нового правила в цепочку (по умолчанию последним);

-I (--insert) - вставка в заданную позицию правила в цепочку (по умолчанию первым);

-R (--replace) - замена в цепочке заданного правила;

-D (--delete) - удаление правила в заданной позиции или первого правила, подходящего специфицированному условию

Опция -j (--jump) указывает действие (DROP или ACCEPT), которое должно быть произведено с пакетами, удовлетворяющими данному правилу.

Пример: Эта команда добавляет в цепочку INPUT правило, которое отбрасывает любые icmp пакеты, направленные с loopback интерфейса

```
[root@linux1]# iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
[root@linux1]# iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination
```

```
DROP            icmp  --    linux1                              anywhere
Chain              FORWARD              (policy              ACCEPT)
target         prot  opt  source                            destination
Chain              OUTPUT               (policy              ACCEPT)
target         prot  opt  source                            destination
[root@linux1]#                     ping                       localhost
PING     linux1     (127.0.0.1)     56(84)     bytes     of     data.
---           linux1           ping           statistics           ---
3 packets transmitted, 0 received, 100% packet loss, time 2006ms
```

Пример: удаляем первое правило в цепочке INPUT

```
[root@linux1]# iptables -D INPUT 1                  [root@linux1]#
iptables -L                             Chain INPUT (policy ACCEPT)
target         prot  opt  source                            destination
Chain FORWARD (policy ACCEPT)                                   target
prot   opt   source                                          destination
Chain OUTPUT (policy ACCEPT)                                    target
prot opt source               destination
```

Пример: удалить можно также, явно указав содержимое правила

```
[root@linux1]# iptables -D INPUT -s 127.0.0.1 -p  icmp -j DROP
```

Опция -s или --src применяется для указания адреса отправителя пакета

Опция -d или --dst применяется для указания адреса получателя пакета.

Адреса отправителя и получателя могут быть указаны либо в виде IP адреса, либо в виде доменного имени.

При необходимости указать группу адресов можно привести адрес сети и через косую черту либо маску сети, либо количество битов в сетевой части IP адреса.

Пример: 192.168.1.0/24 или 199.95.207.0/255.255.255.0.

Если в качестве адреса указана конструкция 0/0 - это означает все адреса.

Пример: запрещаем получение всех пакетов от всех хостов

```
[root@linux1 root]# iptables -A INPUT -s 0/0 -j DROP
```

Для инверсии значения опций указания адресов (-s и -d) можно использовать восклицательный знак, устанавливаемый перед адресом.

Пример: запрещаем получение всех пакетов от всех хостов, кроме локальных пакетов

```
[root@linux1 root]# iptables -A INPUT -s ! localhost  -j DROP
```

Для указания протокола следует использовать опцию -p, после которой необходимо указать либо имя, либо номер протокола (/etc/protocols).

Если перед именем или номером протокола указан восклицательный знак, это обозначает все протоколы, кроме указанного.

Указать интерфейс, с которого должны приходить пакеты можно, используя опцию -i (--in-interface).

Опция -o (--out-interface) указывает интерфейс, через который систему покидают исходящие пакеты.

Для указания всех интерфейсов, начинающихся с некоторого префикса, например, eth, следует указать этот префикс, добавив после него +

Пример:

eth+ - все Ethernet интерфейсы.

Перед именем интерфейса допустимо указывать восклицательный знак для указания всех интерфейсов, кроме заданного.

При использовании протоколов TCP или UDP можно указать порт назначения --destination-port или --dport, а также порт источника --source-port или --sport.

В случае фрагментации пакетов второй и следующие фрагменты не будут содержать такую информацию, как номера портов, тип ICMP сообщения.

Опция -f позволяет строить правила для второго и последующего фрагмента, а установка восклицательного знака перед -f (! -f) будет применять правило только для первого фрагмента или нефрагментированных пакетов.

Применение трансляции сетевых адресов вызывает изменения в адресах отправителя или получателя пакетов. При этом происходит запоминание информации об измененном пакете так, чтобы для ответного пакета можно было провести обратное преобразование.

Трансляцию сетевых адресов принято разделять на две категории:

Source NAT (SNAT) - трансляция адреса отправителя.

Destination NAT (DNAT) - трансляция адреса получателя.

Таблица NAT поддерживает три встроенные цепочки

PREROUTING – трансляция адресов до принятия решения о

маршрутизации

POSTROUTING - трансляция адресов после принятия решения о маршрутизации

OUTPUT – трансляция адресов, созданных локальным процессом, до принятия решения о маршрутизации.

Трансляция адреса отправителя SNAT изменяет адрес отправителя пакета и всегда происходит после маршрутизации непосредственно перед тем, как пакет попадает в канал связи.

Маскарадинг (masquerading) - это форма SNAT.

Трансляция адреса назначения DNAT изменяет адрес назначения пакета и происходит до маршрутизации сразу после прихода пакета из канала связи.

Перенаправление портов, прозрачное проксирование и распределение нагрузки на серверы - это формы DNAT.

SNAT, используемая для подмены адреса источника пакета, производится в цепочке POSTROUTING непосредственно перед выходом пакета в линию связи.

Для включения SNAT необходимо

указать цель (target) -j SNAT

опцию --to, после которой необходимо указать адрес или диапазон адресов, которые будут использованы для подмены исходного адреса

опцию -o для указания внешний интерфейс.

Примеры: Изменяем адреса отправителя на 1.2.3.4.

```
 iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4
Измененяем адреса отправителя на диапазон 1.2.3.4, 1.2.3.5 или 1.2.3.6
 iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4-1.2.3.6
Измененяем адреса отправителя на диапазон 1.2.3.4 и порты 1-1023
 iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to 1.2.3.4:1-1023
```

При использовании в цепи POSTROUTING цели MASQUERADE включается режим маскарадинга, являющегося частным случаем SNAT.

В режиме маскарадинга не нужно указывать адрес(а) для подмены адреса отправителя, поскольку адрес отправителя заменяется на адрес внешнего интерфейса.

Пример: адрес внешнего интерфейса eth0 будет присвоен всем исходящим пакетам

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Трансляция адресов получателя DNAT осуществляется в цепи PREROUTING или OUTPUT. При этом требуется указать

цель -j DNAT

опцию --to, которая указывает на какой адрес(а) должен быть подменен адрес получателя

опцию -i для указания входного интерфейса

Пример: Изменяем адреса получателя на 5.6.7.8

```
 iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 5.6.7.8
Измененяем диапазон адресов получателей на 5.6.7.8, 5.6.7.9 or 5.6.7.10.
 iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 5.6.7.8-5.6.7.10
Измененяем адреса получателей для web траффика to 5.6.7.8, port 8080.
 iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0 -j DNAT --to 5.6.7.8:8080
```

Для устройства прозрачного прокси сервера нужно воспользоваться специализированной формой DNAT, называющейся перенаправлением (redirection). Перенаправление можно включить, указав исходный порт, цель REDIRECT и порт, на который будут перенаправлен траффик.

Пример: Пересылка входящего на 80-й порт web траффика на прозрачный прокси squid

```
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

Iptables позволяет отображать диапазон адресов в один адрес

Пример:

```
iptables -t nat -A POSTROUTING -s 192.168.1.1 -o eth1 -j SNAT --to 1.2.3.1 iptables -t nat -A POSTROUTING -s 192.168.1.2 -o eth1 -j SNAT --to 1.2.3.1
```

Можно отображать сеть в сеть

Пример:

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth1 -j SNAT
--to 1.2.3.0/24
```

Для обеспечения правильной работы FTP с сетью, использующей трансляцию адресов, следует использовать загружаемые модули ядра ip_conntrack_ftp.o и ip_nat_ftp.o.

При использовании трансляции адресов все пакеты, исходящие из локальной сети и входящие в сеть, должны проходить через хост, осуществляющий трансляцию адресов.

Если используется трансляция адресов отправителя SNAT, при которой некоторый адрес отображается на неиспользуемые адреса в той же сети, компьютер, осуществляющий NAT, должен отвечать за правильное разрешение ARP, чего проще всего добиться с помощью IP алиасов.

При отображении адресов на совершенно другую сеть следует позаботиться, чтобы компьютер, осуществляющий NAT, мог принимать ответные пакеты. Это достигается с помощью конфигурации таблицы маршрутизации. Если компьютер с NAT является маршрутизатором по умолчанию, то изменений в таблице маршрутизации не требуется.

Если осуществляется трансляция DNAT порта в той же сети, следует убедиться, что все пакеты и ответные пакеты проходят через компьютер с NAT.

Пример: Эта команда осуществляет трансляцию по назначению DNAT для внутренней сети, отображая адрес 1.2.3.4 в адрес внутреннего сервера 192.168.1.1 для всех http пакетов.

```
iptables -t nat -A PREROUTING -d 1.2.3.4 -p tcp --dport 80 -j DNAT
--to 192.168.1.1
Для обеспечения получения ответных пакетов с веб-сервера
необходимо либо перенаправить ответы на него с помощью DNS
сервера, либо использовать SNAT для отображения на адрес сервера.
iptables -t nat -A POSTROUTING -d 192.168.1.1 -s 192.168.1.0/24 \
-p tcp --dport 80 -j SNAT --to 192.168.1.250
В этой команде предполагается, что адрес NAT компьютера
192.168.1.250, а все пакеты по порту web, направляющиеся по адресу
192.168.1.1, отображаются в адрес NAT сервера 192.168.1.250.
```

Netfilter имеет модульную архитектуру, что обеспечивает хорошую расширяемость функциональности пакетного фильтра.

Расширения Netfilter реализуются или в виде дополнительных модулей

ядра или библиотек Iptables.

Расширения Iptables являются разделяемыми библиотеками и находятся в каталоге /lib/iptables или /usr/lib/iptables.

Пример:

```
[root@rhe4             ~]#            ls            /lib/iptables/
libipt_ah.so             libipt_LOG.so             libipt_SAME.so
libipt_CLASSIFY.so      libipt_mac.so              libipt_sctp.so
libipt_connlimit.so    libipt_mark.so              libipt_SNAT.so
libipt_connmark.so     libipt_MARK.so           libipt_standard.so
libipt_CONNMARK.so        libipt_MASQUERADE.so     libipt_state.so
libipt_conntrack.so    libipt_MIRROR.so           libipt_TARPIT.so
libipt_DNAT.so            libipt_multiport.so     libipt_tcpmss.so
libipt_dscp.so           libipt_NETMAP.so         libipt_TCPMSS.so
libipt_DSCP.so            libipt_NOTRACK.so         libipt_tcp.so
libipt_ecn.so             libipt_owner.so           libipt_tos.so
libipt_ECN.so              libipt_physdev.so        libipt_TOS.so
libipt_esp.so             libipt_pkttype.so       libipt_TRACE.so
libipt_helper.so         libipt_realm.so            libipt_ttl.so
libipt_icmp.so            libipt_recent.so          libipt_TTL.so
libipt_iprange.so         libipt_REDIRECT.so        libipt_udp.so
libipt_length.so         libipt_REJECT.so          libipt_ULOG.so
libipt_limit.so      libipt_rpc.so         libipt_unclean.so
```

Модули ядра, расширяющие функциональность Iptables, находятся в каталоге /lib/modules/`uname -r`/kernel/net/ipv4/netfilter.

Пример:

```
root@rhe4                          ~]#                              ls
/lib/modules/2.6.9-5.EL/kernel/net/ipv4/netfilter/
arptable_filter.ko              ipt_addrtype.ko      ipt_NETMAP.ko
arp_tables.ko                  ipt_ah.ko            ipt_NOTRACK.ko
arpt_mangle.ko                    ipt_CLASSIFY.ko      ipt_owner.ko
ip_conntrack_amanda.ko         ipt_comment.ko       ipt_physdev.ko
ip_conntrack_ftp.ko             ipt_conntrack.ko    ipt_pkttype.ko
ip_conntrack_irc.ko             ipt_dscp.ko           ipt_realm.ko
ip_conntrack.ko                ipt_DSCP.ko          ipt_recent.ko
ip_conntrack_proto_sctp.ko   ipt_ecn.ko             ipt_REDIRECT.ko
ip_conntrack_tftp.ko           ipt_ECN.ko            ipt_REJECT.ko
ip_nat_amanda.ko                 ipt_esp.ko            ipt_SAME.ko
ip_nat_ftp.ko                     ipt_helper.ko        ipt_sctp.ko
ip_nat_irc.ko                    ipt_iprange.ko       ipt_state.ko
ip_nat_snmp_basic.ko            ipt_length.ko        ipt_tcpmss.ko
ip_nat_tftp.ko                  ipt_limit.ko         ipt_TCPMSS.ko
ip_queue.ko                       ipt_LOG.ko            ipt_tos.ko
```

```
iptable_filter.ko                ipt_mac.ko              ipt_TOS.ko
iptable_mangle.ko                ipt_mark.ko             ipt_ttl.ko
iptable_nat.ko                   ipt_MARK.ko           ipt_ULOG.ko
iptable_raw.ko                                    ipt_MASQUERADE.ko
ip_tables.ko                ipt_multiport.ko
```

Все расширения можно разделить на две категории:

расширения для составления новых шаблонов фильтрации

расширения, создающие новые цели (target).

При работе с протоколами TCP, UDP и ICMP расширения будут использованы автоматически.

При использовании опции -p (--protocol) команды iptables будет загружен модуль расширения для нового шаблона фильтрации. После опции -p указывается имя протокола, а модуль будет загружен автоматически.

При необходимости загрузить модуль расширения явно следует использовать опцию -m команды iptables, после которой следует указать имя модуля.

Для получения помощи по загружаемому расширению следует использовать опцию -h (--help) после опции -p или -m, указывающих на модуль, который предполагается загрузить.

Пример:

```
[root@linux1 root]# iptables -p tcp --help .... TCP v1.2.8
options:
--tcp-flags [!] mask comp      match when TCP flags & mask == comp
(Flags: SYN ACK FIN RST URG PSH ALL NONE)      [!] --syn
match        when        only        SYN        flag        set
(equivalent to --tcp-flags SYN,RST,ACK SYN)   --source-port [!]
port[:port]                                                  --
sport                                                       ...
match source port(s)                                --destination-port
[!] port[:port]                                              --
dport                                                       ...
match destination port(s)                            --tcp-option [!]
number       match if TCP option set
```

Данные расширения загружаются, если указано -p tcp (--protocol tcp).

Опция --tcp-flags загружает расширение, позволяющее обрабатывать пакеты определенного типа, например, ACK или SYN.

Опция --syn предназначена для обработки пакетов SYN.

Опция --sport указывает порт или порты источника пакетов. Порты могут быть указаны именами (/etc/services), либо их номерами.

Опция --dport предназначена для указания порта или портов назначения пакетов. Порты указываются аналогично --sport.

Диапазоны портов удобно задавать двумя именами портов, разделенными символом двоеточие.

Если после имени порта указано двоеточие, то это обозначает данный порт и все порты, номера которых больше.

Если двоеточие стоит перед именем порта, то это обозначает все порты, с номерами меньше, либо равными номеру данного порта.

Опция --tcp-option позволяет указать биты, установленные в заголовке TCP пакета.

Пример:отбрасываются TCP пакеты, в которых установлены флажки SYN и ACK

```
[root@linux1]# iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -j
DROP              [root@linux1]#       iptables       -L       INPUT
Chain               INPUT               (policy              ACCEPT)
target            prot    opt    source                destination
DROP                  icmp    --       linux1              anywhere
DROP                  all       --    !linux1              anywhere
DROP           tcp     --     anywhere     anywhere             tcp
flags:FIN,SYN,RST,PSH,ACK,URG/SYN,ACK
```

Расширения UDP загружаются при включении опции -p udp, а расширения для протокода ICMP - -p icmp.

Для протокола UDP можно указать опции --dport и --sport, имеющие такой же смысл, как и для TCP.

Опция --icmp-type позволяет указать тип пакетов ICMP, используя либо название типа (например, host-unreachable), либо номер типа.

При необходимости указания типа и кода пакета ICMP одновременно, следует указать их через черту, например: 3/3.

Все расширения, не относящиеся к протоколам TCP, UDP или ICMP, следует загружать явно, используя ключ -m.

Модуль multiport позволяет через запятую перечислять порты в адресе отправителя и/или получателя. Используется вместе с опциями --sport, --dport и --ports

Пример:

```
iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p
TCP \ --sport 1024:65535 -m multiport --dport 80,443 -j ACCEPT
```

Модуль mac, загружаемый опцией -m mac (--match mac), позволяет указывать MAC адреса сетевых карт для цепочек INPUT и FORWARD. Используется только с опцией --mac-source.

Опция --mac-source, используемая при загруженном модуле mac предназначена для указания после нее MAC адреса сетевого интерфейса.

Модуль limit используется для ограничения некоторых событий, например, частоты записи в журнал. Может использоваться с опциями --limit и --limit-burst.

Опция --limit с последующим цифровым значением предназначена для указания максимальной средней частоты событий в секунду. Если через черту после числа указано /second, /minute, /hour или /day, то это значит, что, соответственно, указано количество событий за секунду, минуту, час или день. По умолчанию - 3 события в час.

Опция --limit-burst обозначает максимальное количество подряд идущих событий, после которого предел будет достигнут. По умолчанию используется значение 5.

Пример: согласно приведенному ниже правилу информация о перенаправленных на другой интерфейс пакетах будет записана при достижении следующих параметров. Будет записаны первые пять таких пакетов (счетчик limit-burst обнулится), после этого 20 минут (60/3) такие пакеты записываться не будут. Каждые следующие 20 минут, если пакет такого типа приходить не будет, то счетчик limit-burst будет увеличиваться на единицу. Если пакет не приходит в течение 100 минут, то счетчик limit-burst будет восстановлен в первоначальное значение (в нашем случае – 5)

```
[root@linux1 root]# iptables -A FORWARD -m limit -j LOG
```

Модуль owner используется в цепочке OUTPUT и позволяет создавать правила в зависимости от имени какого пользователя отправлен пакет.

Допустимые опции модуля owner:

--uid-owner – соответствие UID

--gid-owner – соответствие GID

--pid-owner – соответствие PID процесса

--sid-owner – соответствие id сессии

--cmd-owner – соответствие имени команды

Модуль state очень полезен для отслеживания состояний сетевых соединений

В модуле state существует одна опция – --state, которая может отследить 4 состояния

NEW – пакет создает новое соединение

ESTABLISHED – пакет принадлежит установленному соединению

RELATED – пакет имеет отношение к соединению, но не является его часть (например, диагностическое ICMP сообщение или ftp data соединение, сопровождающее ftp сессию)

INVALID - пакет, который не может быть идентифицирован

Модуль unclean позволяет осуществлять различные случайные проверки целостности пакета. Unclean – экспериментальный модуль и его не следует использовать на промышленных системах.

Модуль LOG предназначен для журналирования информации о пакетах, удовлетворяющих правилу. Может использоваться с опциями --log-level, --log-prefix, --log-tcp-sequence, --log-tcp-options, --log-ip-options.

Опция --log-level предназначена для указания уровня (приоритет) сообщения для журналирования с помощью службы syslogd (категория (facility) сообщений – kern)

Опция --log-prefix позволяет установить строку префикса длиной до 29 символов, которая будет выводиться перед каждым сообщением.

Для записи номеров TCP пакетов следует использовать опцию --log-tcp-sequence.

Опция --log-tcp-options позволяет записывать опции из заголовков TCP пакетов.

Опция --log-ip-options нужна для записи опций из заголовков IP пакетов.

Модуль REJECT позволяет использовать действие (цель) REJECT в правилах. Это действие действует аналогично DROP, но при этом производится посылка сообщения ICMP 'port unreachable'.

Цель REJECT может использоваться только в цепях INPUT, FORWARD и OUTPUT.

Сообщение ICMP 'port unreachable' не посылается в случаях (RFC1122):

фильтруемый пакет имеет в начале сообщение об ошибке ICMP или неизвестный тип пакета ICMP;

фильтруемый фрагмент пакета не является первым;

для данного адресата производится посылка слишком большого количества пакетов ICMP с сообщениями об ошибках.

Цель REJECT можно использовать с опцией --reject-with, позволяющей

указывать тип ICMP сообщения об ошибке.

Пользователь может создавать свои цепочки правил, отличные от встроенных (INPUT, OUTPUT, FORWARD).

Создание своих цепочек позволяет упростить настройку сложных и/или иерархических правил фильтрации.

Пример:

```
[root@nb ~]# iptables -N ssh_access
[root@nb ~]# iptables -I ssh_access -s 1.2.3.4/32 -j ACCEPT
[root@nb ~]# iptables -I ssh_access -s 9.8.7.6/32 -j ACCEPT
[root@nb ~]# iptables -I ssh_access -s 192.168.1.0/24 -j DROP
[root@nb ~]# iptables -A ssh_access  -j DROP
[root@nb ~]# iptables -I INPUT -p tcp --dport 22 -j ssh_access
[root@nb ~]# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ssh_access  tcp  --  0.0.0.0/0            0.0.0.0/0           tcp
dpt:22
...
Chain ssh_access (1 references)
target     prot opt source               destination
ACCEPT     all  --  127.0.0.1            0.0.0.0/0
DROP       all  --  192.168.1.0/24       0.0.0.0/0
ACCEPT     all  --  9.8.7.6              0.0.0.0/0
ACCEPT     all  --  1.2.3.4              0.0.0.0/0
DROP       all  --  0.0.0.0/0            0.0.0.0/0
```

Примечание: В примере выше создается новая цепочка правил с названием ssh_access. Затем цепочка заполняется правилами фильтрации. Обратите внимание на то, что последнее правило в цепочке добавляется с опцией -A. Затем в цепочке INPUT создается правило для перехода в цепочку при обращении на порт 22.

По соглашению имена пользовательских цепочек правил необходимо указывать в нижнем регистре.

Пакет попадает на обработку в пользовательскую цепочку тогда, когда она указана в качестве действия (цели) в каком-либо правиле, которому удовлетворил данный пакет.

Если пакет проходит всю цепь, определенную пользователем, и не встречает ни одного правила, которому он подходит, он возвращается для обработки в следующее правило той цепи, которую он покинул.

Цель RETURN позволяет немедленно покидать цепочку. При этом, для встроенной цепочки управление передается политике, а для пользовательской

следующему правилу в цепочке, из которой был произведен переход.

Цель QUEUE ставит пакеты в очередь для обработки на пользовательском уровне (user side), а не на уровне ядра (kernel side).

Если в пользовательском пространстве нет процесса, который ожидает данный сетевой пакет, то пакет отбрасывается

Библиотека libipq предоставляет API для создания таких приложений.

Iptables позволяет вести гибкий учет трафика по любым параметрам, например, по сетям и адресам, протоколам, портам и т.д.

Для учета создается требуемая фильтрующая цепочка (или цепочки), которая принимает все пакеты нужного типа, а затем количество пакетов и байт, попавших в эту цепочку, можно узнать командой iptables -vx -L имя_цепочки

Сброс значений байтов и пакетов в цепочке можно сделать командой iptables -Z имя_цепочки

Пример: Создадим три цепочки: цепочку ACCOUNT_IN для подсчета входящего трафика, цепочку ACCOUNT_OUT для подсчета исходящего трафика, цепочку ACCOUNT для подсчета суммарного трафика и напишем простейший вариант вывода статистики по трафику (можно было не создавать дополнительные пользовательские цепочки, а воспользоваться только встроенными цепочками INPUT и OUTPUT, но предлагаемый механизм учета является более универсальным

```
[root@linux1]# iptables -N ACCOUNT [root@linux1]# iptables -A ACCOUNT -j ACCEPT [root@linux1]# iptables -N ACCOUNT_IN [root@linux1]# iptables -A ACCOUNT_IN -j ACCOUNT [root@linux1]# iptables -N ACCOUNT_OUT [root@linux1]# iptables -A ACCOUNT_OUT -j ACCOUNT [root@linux1]# iptables -I INPUT -d 192.168.1.2 -j ACCOUNT_IN [root@linux1]# iptables -I OUTPUT -s 192.168.1.2 -j ACCOUNT_OUT [root@linux1]# iptables -vx -L | grep -v '^Chain' | grep -v 'pkts' |\ sed '1,4d' | awk '{print $2}' | tr '\n' ' '|\ awk '{ print "Total " $1 " In " $2 " Out " $3 }' Total 257460 In 130522 Out 12693
```

Устанавливая сервер для обслуживания клиентских запросов обдумайте вариант защиты сетевых соединений к этому серверу.

Вы можете установить сервер напрямую подключенным к интернету, тогда для его защиты вам необходимо применять локальный брандмауэр, основанный на Netfilter.

Лучшим, с точки зрения безопасности, будет установить на границе вашей сети специализированный брандмауэр для контроля сетевого трафика

всей сети. Такие системы могут, в том числе, использовать в качестве основы Linux. При этом установка централизованного решения не отменяет настройку локальной защиты.

Преимущества специализированных решений:

Поддержка продвинутых механизмов маршрутизации.

Настройка ориентированная на обеспечение безопасности.

Продвинутая поддержка VPN.

В некоторых случаях контентная фильтрация.

Одной из мер защиты может рассматриваться применение прокси серверов для фильтрации и оптимизации трафика.

В RedHat подобных системах в качестве базового решения для фильтрации трафика можно применить службу iptables. Это простая служба, которая при своем старте загружает правила фильтрации трафика из файла.

Конфигурационный файл /etc/sysconfig/iptables содержит правила фильтрации.

Формат конфигурационного файла такой же как и вывод команды iptables-save.

В последних версиях RedHat подобных Linux эта служба не устанавливается по умолчанию. Вместо нее применяется служба firewalld.

Пример:

```
[root@sl0 ~]# cat /etc/sysconfig/iptables
# sample configuration for iptables service
# you can edit this manually or use system-config-firewall
# please do not ask us to add additional ports/services to this
default configuration
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Имеются несколько сценариев, в которых применение службы iptables является нецелесообразным. Например: вы хотите применять разные правила фильтрации трафика для разных интерфейсов или разных сетей. Или возможно

вы захотите делегировать полномочия на управление брандмауэром. Для решения этих задач предназначена служба firewalld, которая является надстройкой над Netfilter.

Основные характеристики службы firewalld:

Динамический управление межсетевым экраном.

Сетевые интерфейсы поделены на зоны. Зоны определяют наборы применяемых правил фильтрации.

Поддержка IPv4 и IPv6, а также Ethernet мостов и IP Set.

Разделение на активную (runtime) и постоянную (permanent) конфигурацию.

Взаимодействие с другими сервисами и приложениями, через D-Bus.

Авторизация пользователей через polkit.

Настройки демона firewalld находятся в XML файлах, за исключением основного конфигурационного файла /etc/firewalld/firewalld.conf.

Пример: Просмотр состояния демона firewalld, и его запуск.

```
[root@sl0 ~]# firewall-cmd --state
not running
[root@sl0 ~]# systemctl enable firewalld
[root@sl0 ~]# systemctl start firewalld
[root@sl0 ~]# firewall-cmd --state
running
```

Пример: получение информации о зоне. И определение интерфейса в зону.

```
[root@sl0 ~]# firewall-cmd --get-default-zone
public
[root@sl0 ~]# firewall-cmd --zone=public --list-services
ssh dhcpv6-client
[root@sl0 ~]# firewall-cmd --get-active-zones
[root@sl0 ~]# firewall-cmd --get-zone-of-interface=eth0
no zone
[root@sl0 ~]#  firewall-cmd --add-interface=eth0 --zone=home
The interface is under control of NetworkManager, setting zone to
'home'.
success
[root@sl0 ~]#  firewall-cmd --get-zone-of-interface=eth0
no zone
[root@sl0 ~]# systemctl restart NetworkManager
[root@sl0 ~]#  firewall-cmd --get-zone-of-interface=eth0
home
```

```
[root@sl0 ~]#   firewall-cmd --add-interface=eth0 --permanent --
zone=work
The interface is under control of NetworkManager, setting zone to
'work'.
success
[root@sl0 ~]#  firewall-cmd --get-zone-of-interface=eth0
work
[root@sl0 ~]#  firewall-cmd --list-services --zone=work
ssh dhcpv6-client
[root@sl0 ~]#  firewall-cmd --list-ports --zone=work
```

Пример: добавление порта TCP в нужную зону.

```
[root@sl0 ~]#   firewall-cmd --add-port=22222/tcp --permanent --
zone=work
success
[root@sl0 ~]#  firewall-cmd --list-ports --zone=work
[root@sl0 ~]#  firewall-cmd --add-port=22222/tcp  --zone=work
success
[root@sl0 ~]#  firewall-cmd --list-ports --zone=work
22222/tcp
[root@sl0 ~]#  firewall-cmd --list-ports --zone=work --permanent
22222/tcp
```

9.3. Tcpdump

tcpdump – инструмент командной строки для анализа пакетов, который позволяет перехватывать сетевые пакеты на основе правил фильтрации, интерпретировать содержимое пакетов и отображать результат в удобочитаемом формате. По умолчанию утилита предустановлена в РЕД ОС.

Однако если по каким-либо причинам утилита отсутствует в системе, перейдите в сеанс пользователя root:

```
su -
```

и установите ее командой:

```
dnf install tcpdump -y
```

Для работы с утилитой необходимы права суперпользователя root или администратора системы (sudo), поэтому здесь и далее команды выполняются с правами суперпользователя root, если не указано иное.

Утилита tcpdump помогает:

диагностировать потерю пакетов;

поверять производительность приложений;

проверять маршрутизацию трафика в многосетевой среде;

отслеживать теги VLAN;

прослушивать трафик.

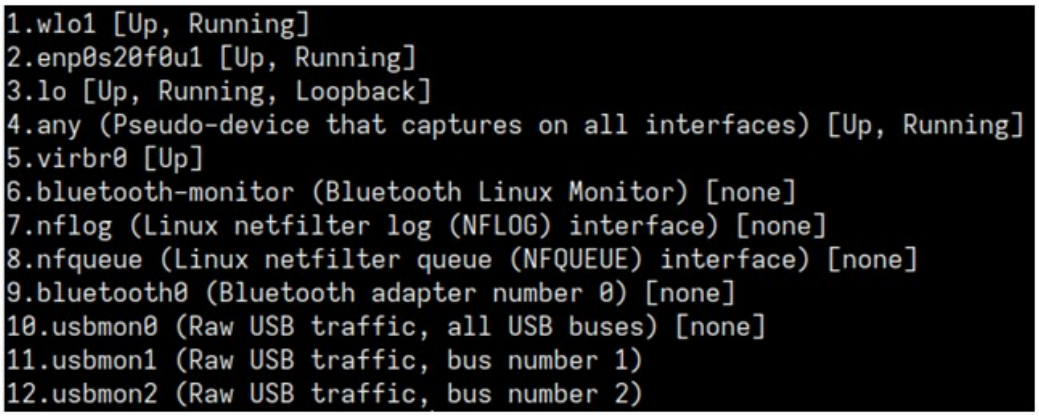
По умолчанию tcpdump прослушивает сетевой интерфейс с наименьшим номером в списке, поэтому сначала необходимо определить, какой интерфейс будет прослушиваться.

Синтаксис команды tcpdump имеет следующий вид:

```
tcpdump [-опции] [фильтры]
```

Для отображения списка сетевых интерфейсов, используемых в системе, выполните команду:

```
tcpdump -D
```



Для прослушивания конкретного сетевого интерфейса необходимо использовать опцию -i с именем интерфейса, передаваемого в качестве параметра. Например, для интерфейса под номером 2 команда будет иметь следующий вид:

```
tcpdump -i enp0s20f0u1
```

Если необходимо захватить трафик со всех сетевых интерфейсов, например, если в системе несколько сетевых подключений, в tcpdump для таких случаев предусмотрен специальный параметр any:

```
tcpdump -i any
```

Описание некоторых опций tcpdump:

Опция	Описание
-a	Преобразовать сетевые и широковещательные адреса в доменные имена.
-e	Отображать данные канального уровня (MAC-адрес, протокол, длина пакета). Помимо IP-адресов будут отображены MAC-адреса компьютеров.

-N	Не добавлять доменное расширение к именам узлов. Например, отобразить net вместо net.library.org.
-n	Отображать IP-адрес вместо имени хоста.
-nn	Отображать номер порта вместо используемого протокола.
-q	Вывести минимум информации. Обычно – имя протокола, отправитель и получатель пакета, используемые порты и объем переданных данных.
-r	Прочесть трафик из файла, если он был предварительно сохранен параметром -w.
-S	Не обрабатывать абсолютные порядковые номера (initial sequence number — ISN) в относительные.
-t	Не отображать метку времени в каждой строке.
-tt	Отображать неформатированную метку времени в каждой строке.
-tttt	Показывать дату и время.
-v	Выводить подробную информацию (TTL; ID; общая длина заголовка, а также его параметры; производить проверку контрольных сумм IP и ICMP-заголовков).
-vv	Выводить ещё более полную информацию, в основном относится к NFS и SMB.
-vvv	Выводить максимально подробную информацию.
-w <файл>	Сохранять данные tcpdump в двоичном формате. Преимущества использования данного способа по сравнению с обычным перенаправлением в файл является высокая скорость записи и возможность чтения подобных данных другими программами, например, snort, но этот файл нельзя прочитать человеку.
-c <число>	Завершит работу tcpdump после получения указанного числа пакетов.

Описанные опции помогают отфильтровывать информативность пакетов, перенаправлять вывод в файл или прочитать уже записанный файл и т.д. Для получения более подробной информации по всем доступным опциям утилиты tcpdump можно воспользоваться командой:

```
man tcpdump
```

Для упрощения поиска пакетов и экономии времени в tcpdump предусмотрены фильтры. Они используются для отсеивания ненужных пакетов. Таким образом можно вывести список пакетов, полученных от какого-либо конкретного IP-адреса или подсети.

Описание фильтров для tcpdump:

host, net, port – для фильтрации по значению соответствующего идентификатора;

src, dst – для фильтрации по конкретному направлению передачи к и/или от идентификатора;

tcp, udp, icmp, ah – для фильтрации по заданному сетевому протоколу.

Примеры использования утилиты

Для просмотра исходящего и входящего трафика по IP-адресу выполните команду с фильтром host и соответствующим аргументом:

```
tcpdump host 192.168.1.1
```

Для обнаружения пакетов, следующих из определенной сети либо в нее, используйте команду с фильтром net:

```
tcpdump net 192.168.1.0/24
```

Для захвата трафика, который поступает только на определенный порт, выполните команду с фильтром port и номером соответствующего порта в качестве аргумента:

```
tcpdump port 22
```

Для отсеивания трафика по отправителю с помощью фильтра src, выполните команду:

```
tcpdump src 192.168.1.1
```

Для отображения входящего трафика с IP-адреса выполните команду:

```
tcpdump dst 192.168.1.1
```

Для поиска конкретного типа трафика используются фильтры tcp, udp, icmp. Например, фильтр icmp поможет вывести на экран весь трафик, передаваемый по протоколу ICMP:

```
tcpdump icmp
```

9.4. Nmap

NMAP — инструмент с открытым исходным кодом для исследования сети и аудита безопасности. Он был разработан для быстрого сканирования больших сетей, но также отлично подходит для отдельных хостов в сети.

Nmap способен определить, какие хосты доступны в сети и какие сервисы они предоставляют, версии операционных систем, используемые типы фильтров пакетов/брандмауэров и десятки других характеристик.

Обычно nmap используется для аудита безопасности, но многие системные и сетевые администраторы находят его полезным для рутинных задач, таких как инвентаризация сети, управление графиками обновления службы и мониторинг работоспособности хостов или служб.

Для установки nmap перейдите в сеанс пользователя root:

```
su -
```

и выполните команду:

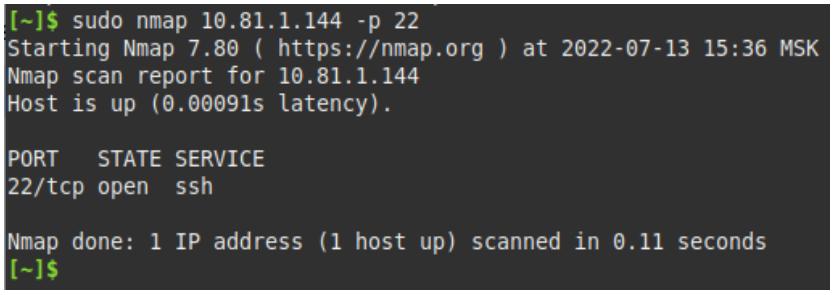
```
dnf install nmap -y
```

Дальнейшая работа с утилитой производится с правами локального пользователя.

Примеры работы

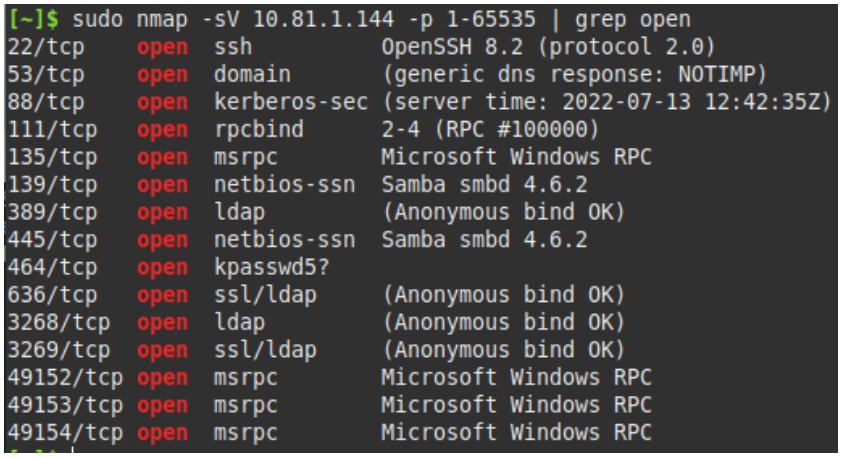
1. Проверка доступности порта:

```
nmap 10.81.1.144 -p 22
```



2. Показать все открытые порты на сетевом узле:

```
nmap -sV 10.81.1.144 -p 1-65535 | grep open
```



Возможен вариант с указанием параметра --open:

```
nmap --open 10.81.1.144
```

```
[~]$ sudo nmap --open 10.81.1.144
Starting Nmap 7.80 ( https://nmap.org )
Nmap scan report for 10.81.1.144
Host is up (0.00062s latency).
Not shown: 985 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
88/tcp    open  kerberos-sec
111/tcp   open  rpcbind
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
```

3. Сканировать сеть методом ping для обнаружения активных узлов:

```
nmap -sn 10.81.81.1/24
```

При этом в выводе команды будут отображаться IP и MAC-адреса.

```
[~]$ sudo nmap -sP 10.81.81.1/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-07-13 16:05 MSK
Nmap scan report for 10.81.81.186
Host is up (0.000055s latency).
MAC Address: 8C:8C:AA:00:26:03 (Unknown)
Nmap scan report for 10.81.81.206
Host is up (0.00017s latency).
MAC Address: 2C:56:DC:4B:22:99 (Asustek Computer)
Nmap scan report for 10.81.81.207
Host is up (0.00027s latency).
MAC Address: 56:6F:94:51:00:01 (Unknown)
Nmap scan report for 10.81.81.210
Host is up (0.00066s latency).
MAC Address: 56:6F:94:51:00:09 (Unknown)
Nmap scan report for 10.81.81.81
Host is up.
```

4. Информация о сетевом узле:

```
nmap -sV 10.81.1.144
```

5. Сканирование определенных портов, в примере указаны 22 (ssh) и 53 (dns):

```
nmap -sV -p 22,53 8.8.8.8
```

```
[~]$ sudo nmap -sV -p 22,53 8.8.8.8
Starting Nmap 7.80 ( https://nmap.org ) at 2022-07-
Nmap scan report for dns.google (8.8.8.8)
Host is up (0.040s latency).

PORT   STATE    SERVICE VERSION
22/tcp filtered ssh
53/tcp open     domain  ISC BIND 9.11.3-1ubuntu1.1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_k
```

Также можно определить диапазон адресов для сканирования:

```
nmap -sV -p 22,53 10.81.1.1-50
```

6. Сканирование всей подсети:

```
nmap 192.168.1.0/24
```

7. Исключение нескольких хостов по их IP-адресам:

```
nmap 192.168.1.0/24 --exclude 192.168.1.10,192.168.1.154
```

8. Для быстрого сканирования используется опция -F:

```
nmap -F 10.81.1.1-10
```

9. Быстрое сканирование открытых портов на всех хостах подсети:

```
nmap -T5 192.168.1.1/24
```

10. Сканирование с определением версии ОС:

```
nmap -O -v --osscan-guess 192.168.1.1
```

11. Сканирование подсети с помощью UDP ping:

```
nmap -PU 192.168.1.1/24
```

12. Сканирование с использованием подмены MAC-адреса, в данном случае одно устройство/приложение маскируется под другое, указывая его MAC-адрес:

```
nmap -v -sT -PN --spoof-mac 0 192.168.1.1
```

В данном примере «0» означает, что nmap выберет случайный MAC-адрес.

9.5. Zenmap

Zenmap — это мультиплатформенный графический интерфейс утилиты nmap и средство просмотра результатов анализа. Zenmap стремится сделать nmap простым в использовании для новичков, а опытным пользователям он предоставляет расширенные возможности.

Для установки zenmap перейдите в сеанс пользователя root:

```
su -
```

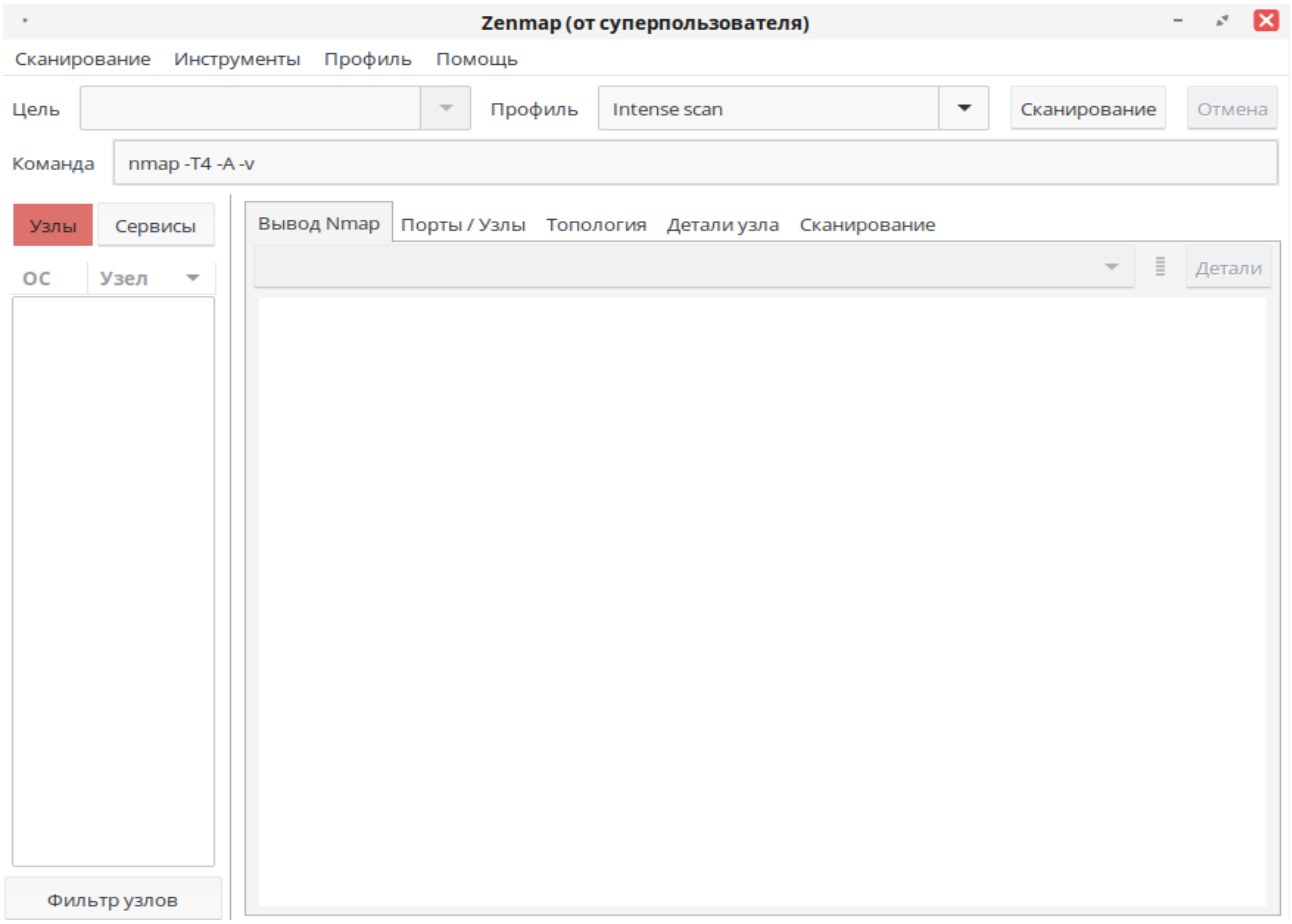
и выполните команду:

```
dnf install nmap-zenmap
```

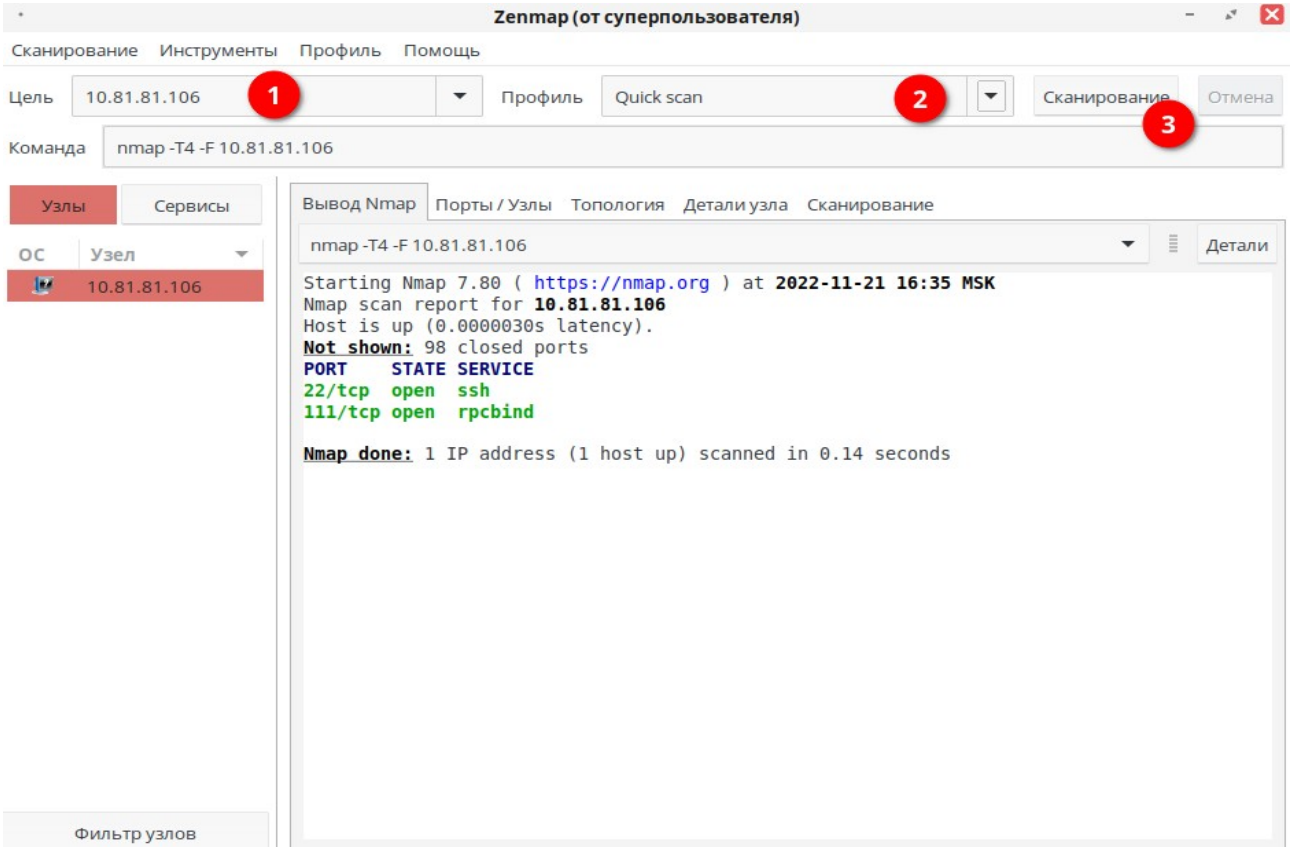
Запуск программы доступен через «Главное меню» - «Интернет» - «Zenmap» или через терминал командой:

```
zenmap
```

Для работы в программе необходимы права суперпользователя root.
Основное окно программы выглядит следующим образом:



Перед началом сканирования необходимо указать в поле «Цель» (1) IP-адрес, имя домена, диапазон IP-адресов или подсеть для сканирования, а во вкладке «Профиль» (2) выбрать профиль сканирования. Запуск сканирования осуществляется нажатием кнопки «Сканирование» (3).



Ниже расположены следующие вкладки:

Вывод Nmap;

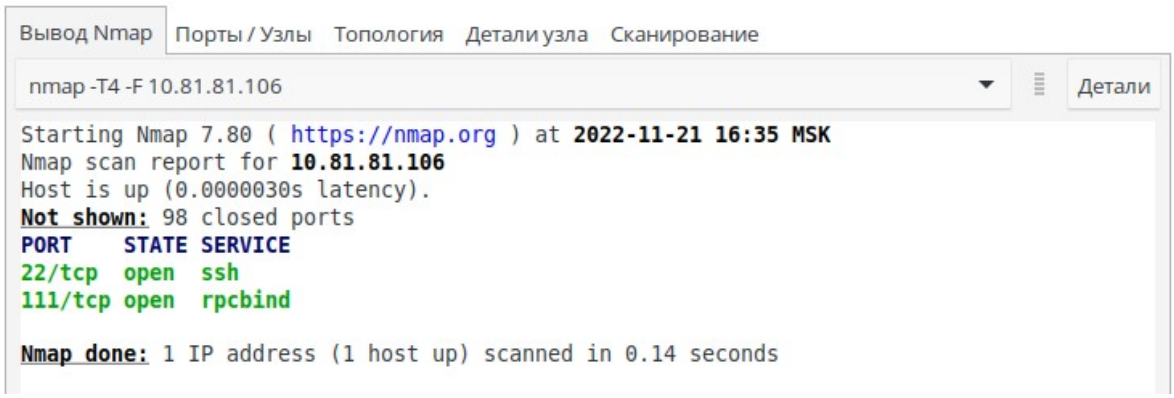
Порты/Узлы;

Топология;

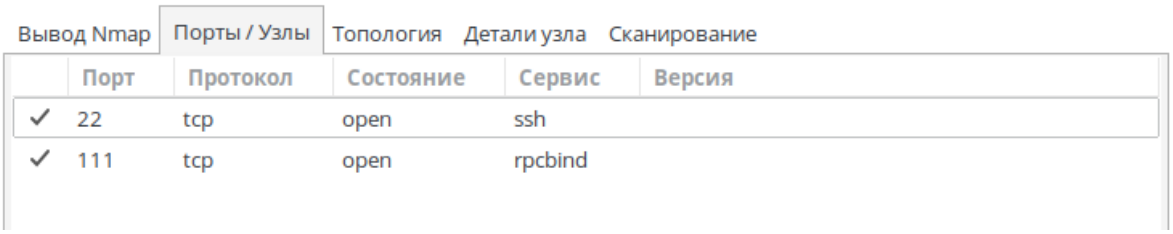
Детали узла;

Сканирование.

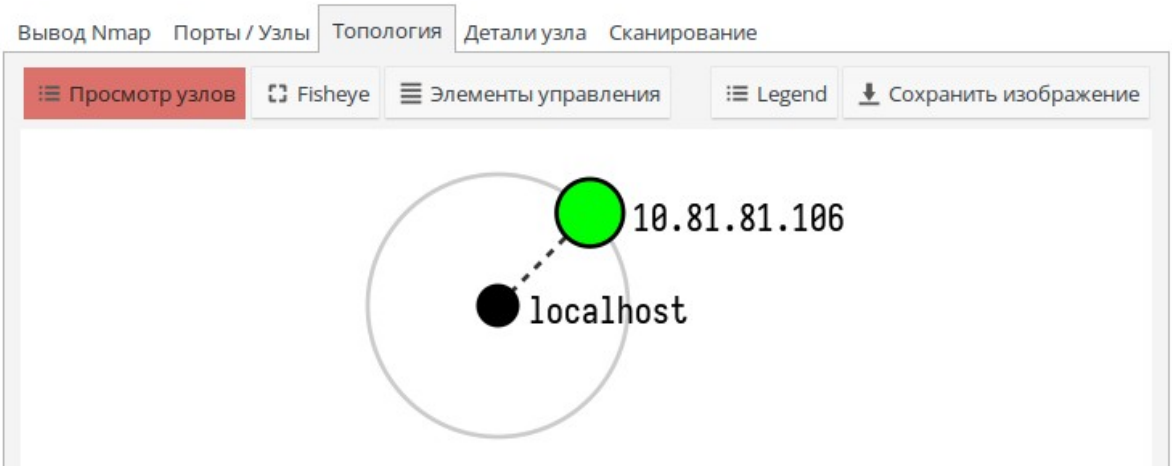
Вкладка «Вывод Nmap» показывает стандартный вывод информации nmap.



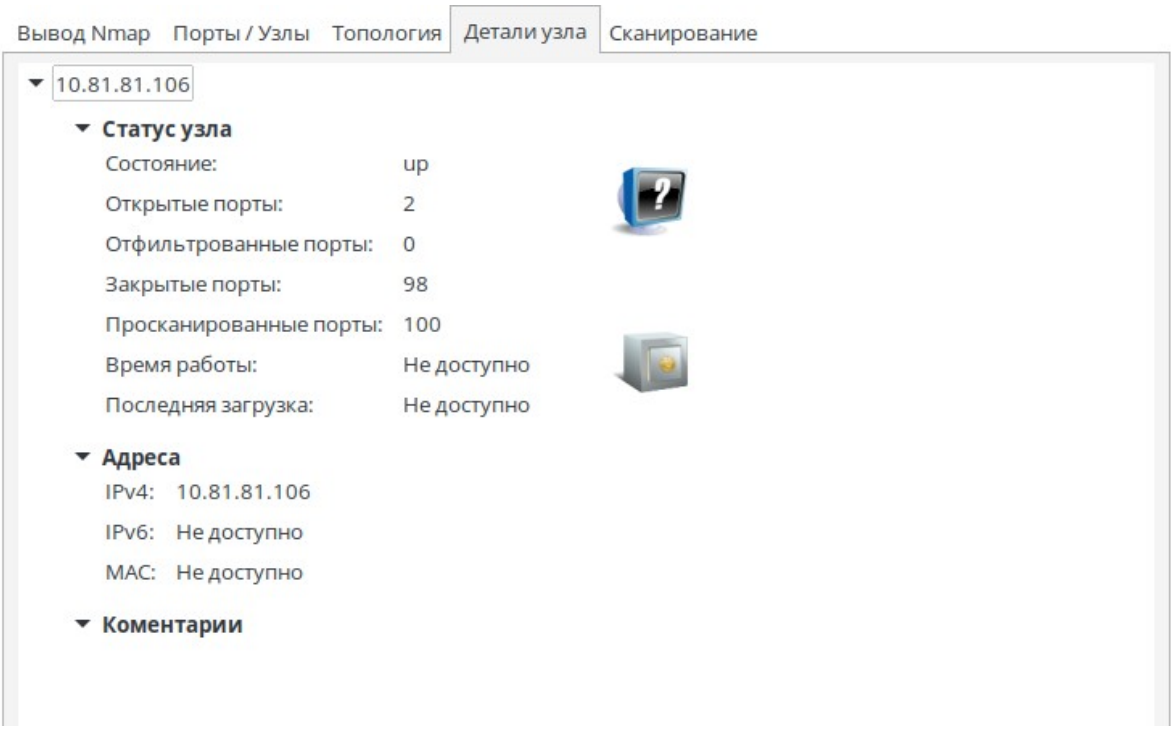
Вкладка «Порты/Узлы» выводит службы или порты с дополнительной информацией, отсортированной по узлам. Если выбран один узел, отображается статус сканированных портов.



Вкладка «Топология» показывает топологию сети цели в графическом режиме.

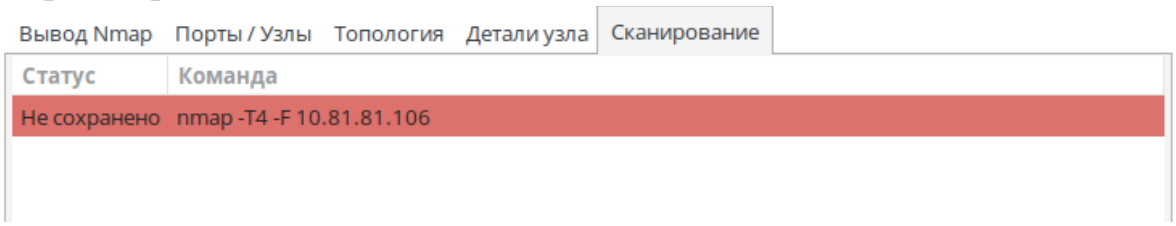


Вкладка «Детали узла» демонстрирует информацию об отсканированном узле, которая включает в себя имя узла, состояние сканированных портов, время безотказной работы, адреса подключения и др.



Вкладка «Сканирование» отображает историю всех выполненных проверок, включая запущенные проверки. Также можно добавить проверки,

импортировав файл кнопкой «Добавить отчет».



В zenmap существует несколько профилей сканирования:

Intense scan – интенсивное сканирование;

Intense scan plus UDP – интенсивное сканирование плюс UDP;

Intense scan, all TCP ports – интенсивное сканирование, со всеми TCP-портами;

Intense scan, no ping – интенсивное сканирование, без пинга;

Ping scan – пинг-сканирование;

Quick scan – быстрое сканирование;

Quick scan plus – быстрое сканирование плюс;

Quick traceroute – быстрая трассировка;

Regular scan – обычное сканирование;

Slow comprehensive scan – медленное комплексное сканирование.

1. Intense scan - этот профиль быстро сканирует наиболее распространенные TCP-порты, а также определяет тип ОС, её службы и версии.

Команда:

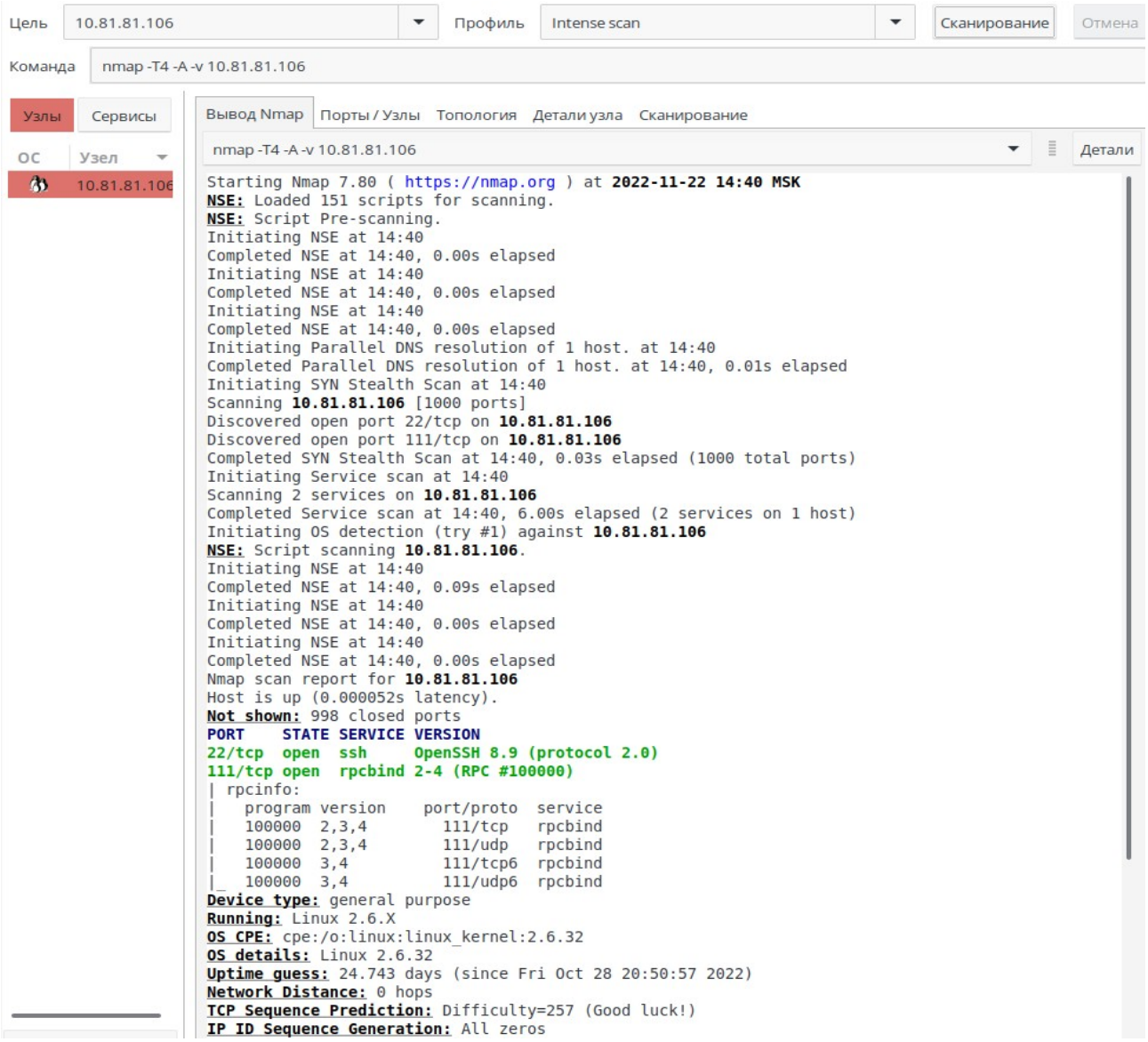
```
nmap -T4 -A -v <цель>
```

где:

-T4 — временной параметр, который находится в диапазоне от 0 до 5, где 0 — самый медленный тип сканирования, а 5 — самый быстрый тип сканирования;

-A — параметр, определяющий тип ОС и ее версии;

-v — параметр, который отображает подробную информацию и детали сканирования, чтобы пользователь знал текущее состояние сканирования.



2. Intense scan plus UDP – работает как обычное интенсивное сканирование, однако дополнительно сканирует порты UDP.

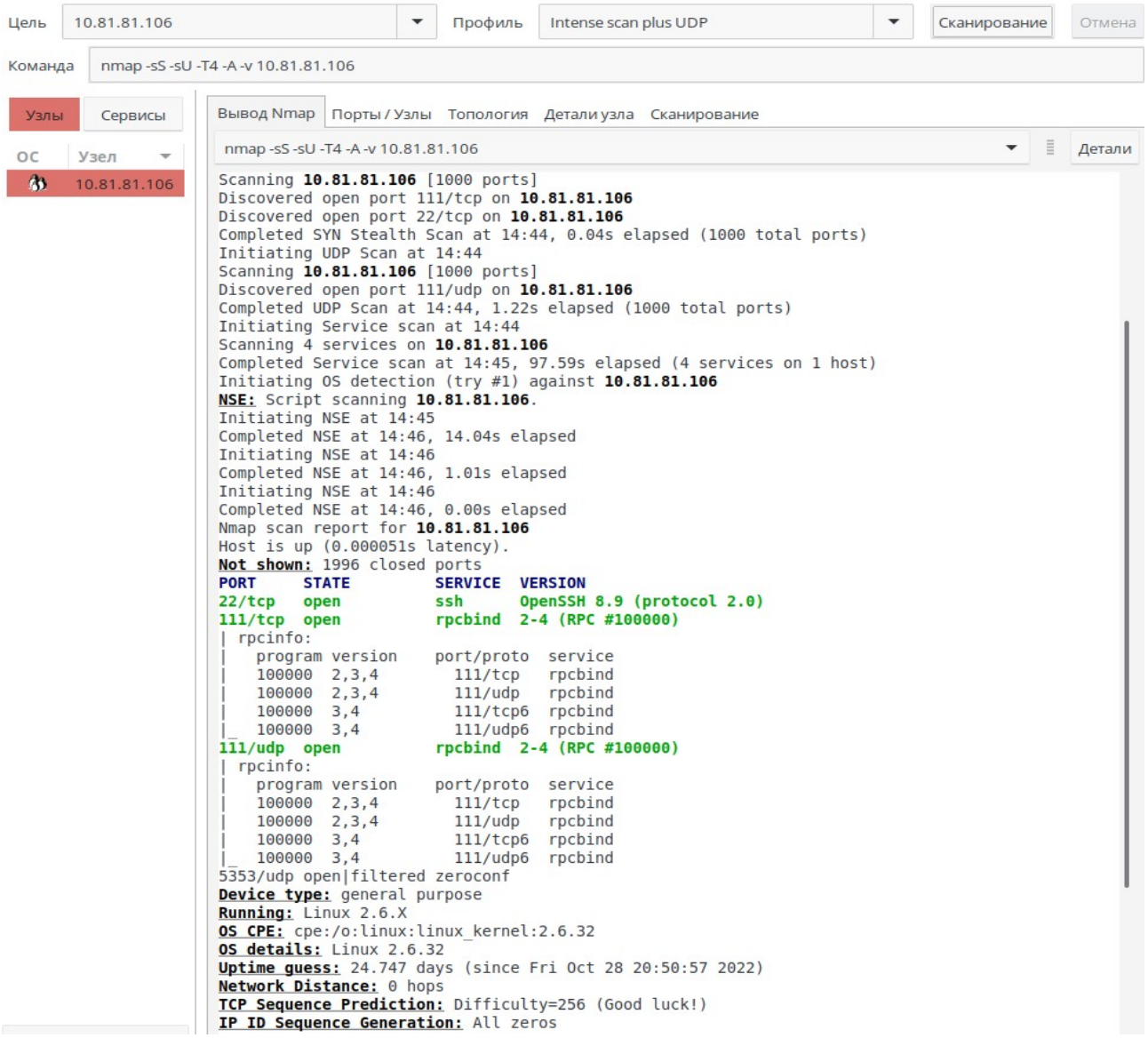
Команда:

```
nmap -sS -sU -T4 -A -v <цель>
```

где:

-sS – указывает nmap сканировать порты TCP с помощью пакетов SYN;

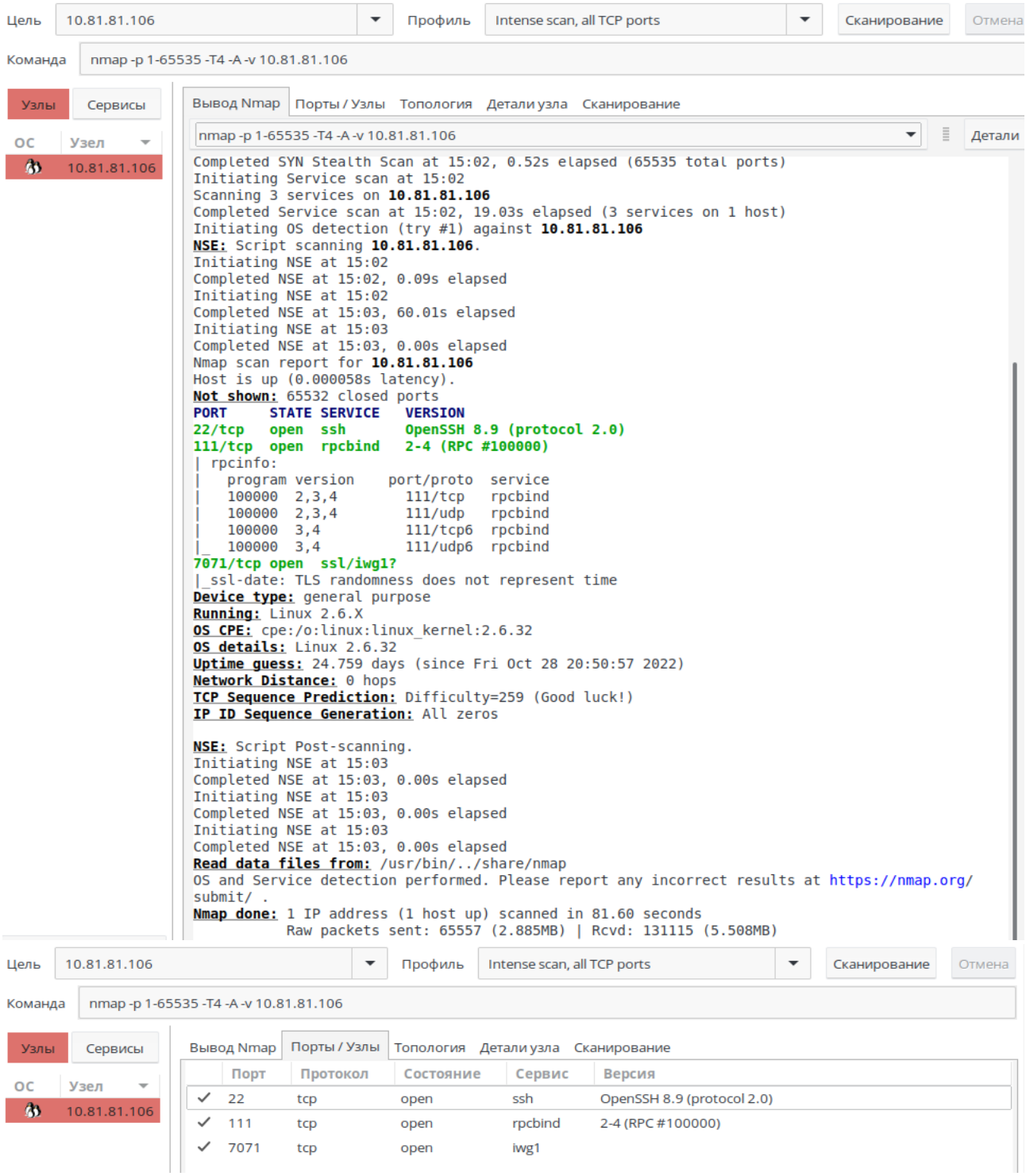
-sU — параметр, отвечающий за сканирование UDP-портов.



3. Intense scan, all TCP ports – поскольку для сканирования всех портов требуется время, nmap обычно сканирует 1000 самых распространённых портов. При Интенсивном сканировании проверка nmap идёт по всем портам от 1 до 65535 (макс.).

Команда:

```
nmap -p 1-65535 -T4 -A -v <цель>
```



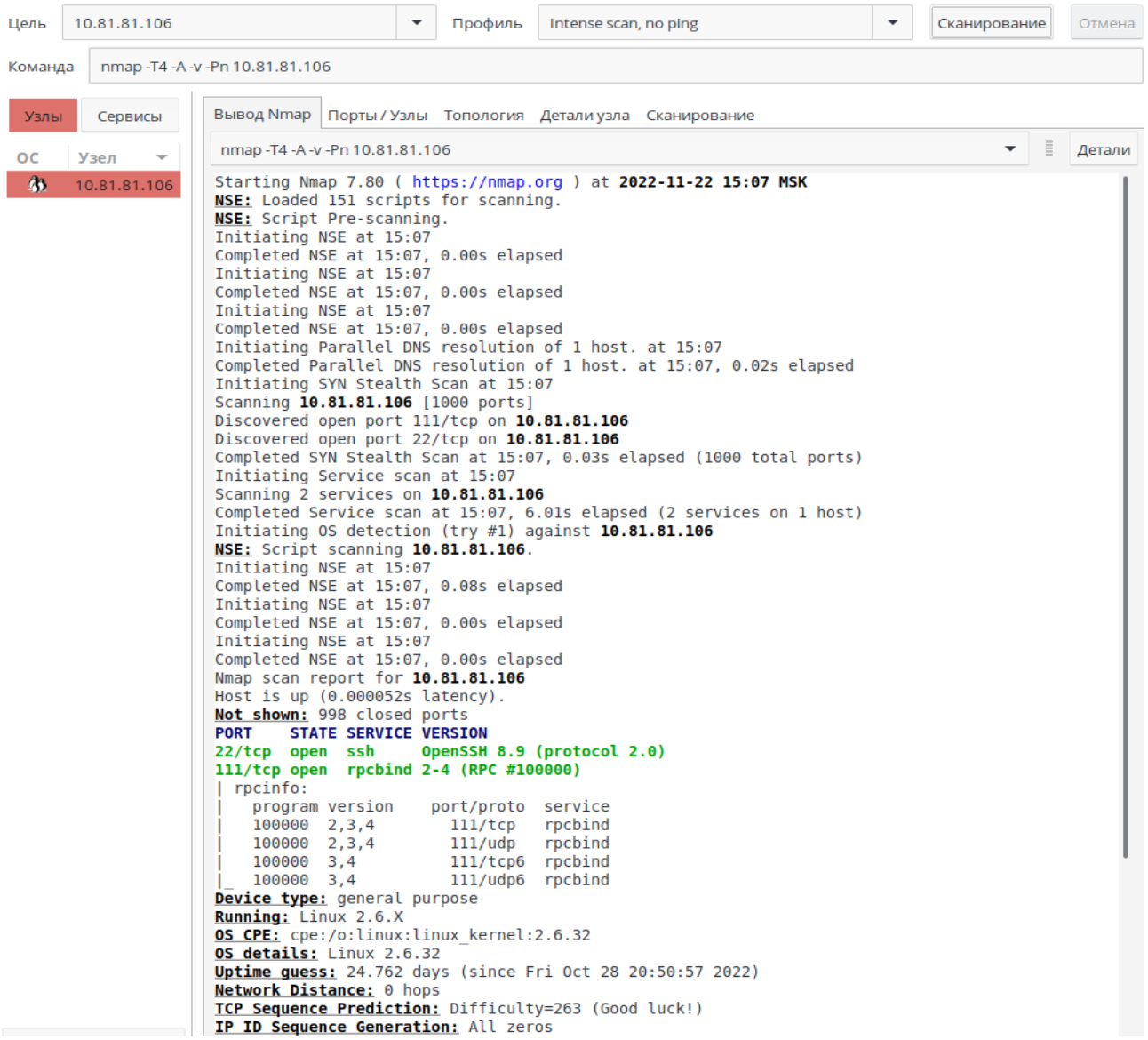
4. Intense scan, no ping – работает точно так же, как и другие интенсивные сканирования. Данный тип сканирования полезен, когда целевой хост блокирует запрос ping, но вы знаете, что он активен.

Команда:

```
nmap -T4 -A -v -Pn <цель>
```

где:

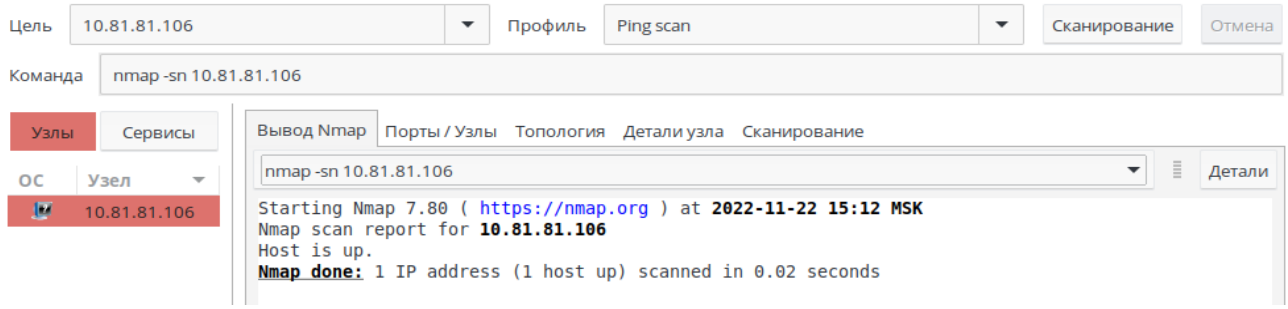
-Pn - параметр предполагает, что хост работает.



5. Ping scan – выполняет только пинг цели, не сканируя порты.

Команда:

```
nmap -sn <цель>
```



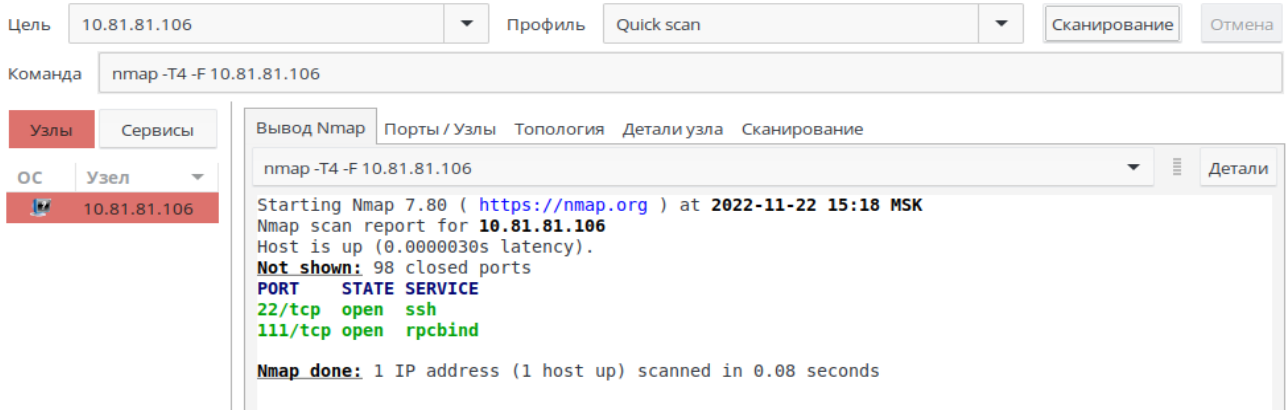
6. Quick scan – сканирует только ограниченное количество портов TCP, т.е. 100 самых распространенных портов TCP.

Команда:

```
nmap -T4 -F <цель>
```

где:

-F — параметр для быстрого сканирования. Вместо того чтобы сканировать все порты, сканируется только несколько портов.



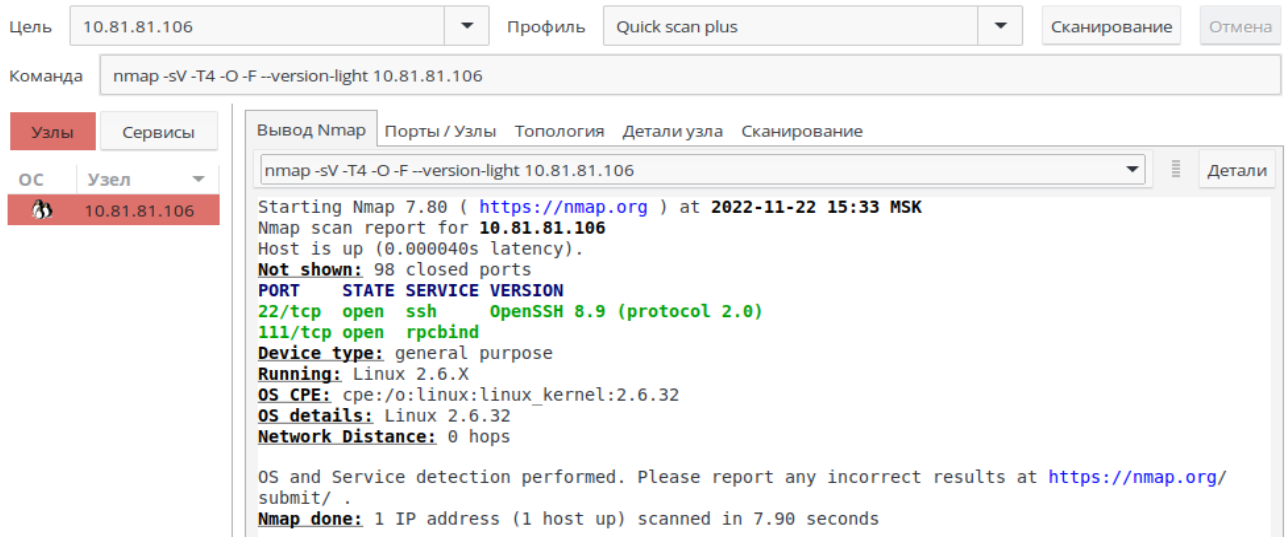
7. Quick scan plus – помимо быстрого сканирования определяет также операционную систему сканируемого хоста и определяет версии служб, за которыми закреплены сканируемые порты.

Команда:

```
nmap -sV -T4 -O -F --version-light <цель>
```

где:

-O — параметр, который определяет тип ОС, а затем выполняет легкое сканирование.



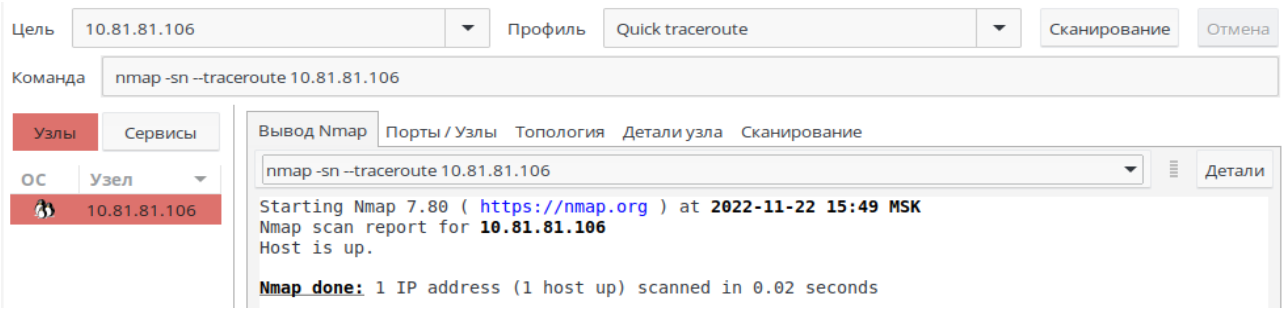
8. Quick traceroute – отслеживает и выполняет пинг всех хостов, определенных в цели.

Команда:

```
nmap -sn --traceroute <цель>
```

где:

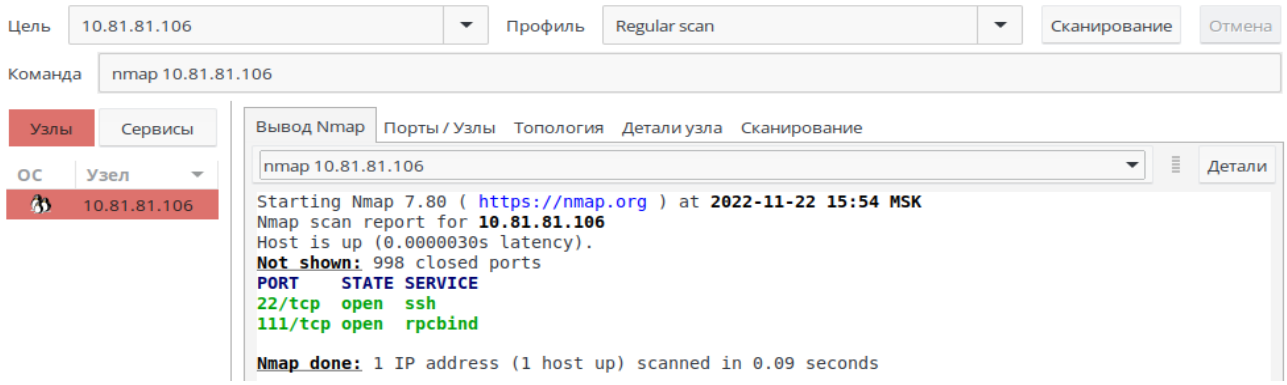
--traceroute — программа, которая записывает маршрут между вашим компьютером и определенным пунктом назначения через Интернет.



9. Regular scan – выполняет сканирование TCP SYN для 1000 наиболее распространенных портов, используя запрос ping для обнаружения хоста.

Команда:

```
nmap <цель>
```



10. Slow comprehensive scan – профиль обладает большим количеством параметров, которые позволяют провести комплексное сканирование сети.

Команда:

```
nmap -sS -sU -T4 -A -v -PE -PP -PS80,443 -PA3389 -PU40125 -PY -g
53 --script "default or (discovery and safe)" <цель>
```

где:

-PE — пинг с использованием ICMP-эхо запросов;

-PP — пинг с использованием запросов временной метки;

-PS<№_порта> - TCP SYN-пинг заданных хостов;

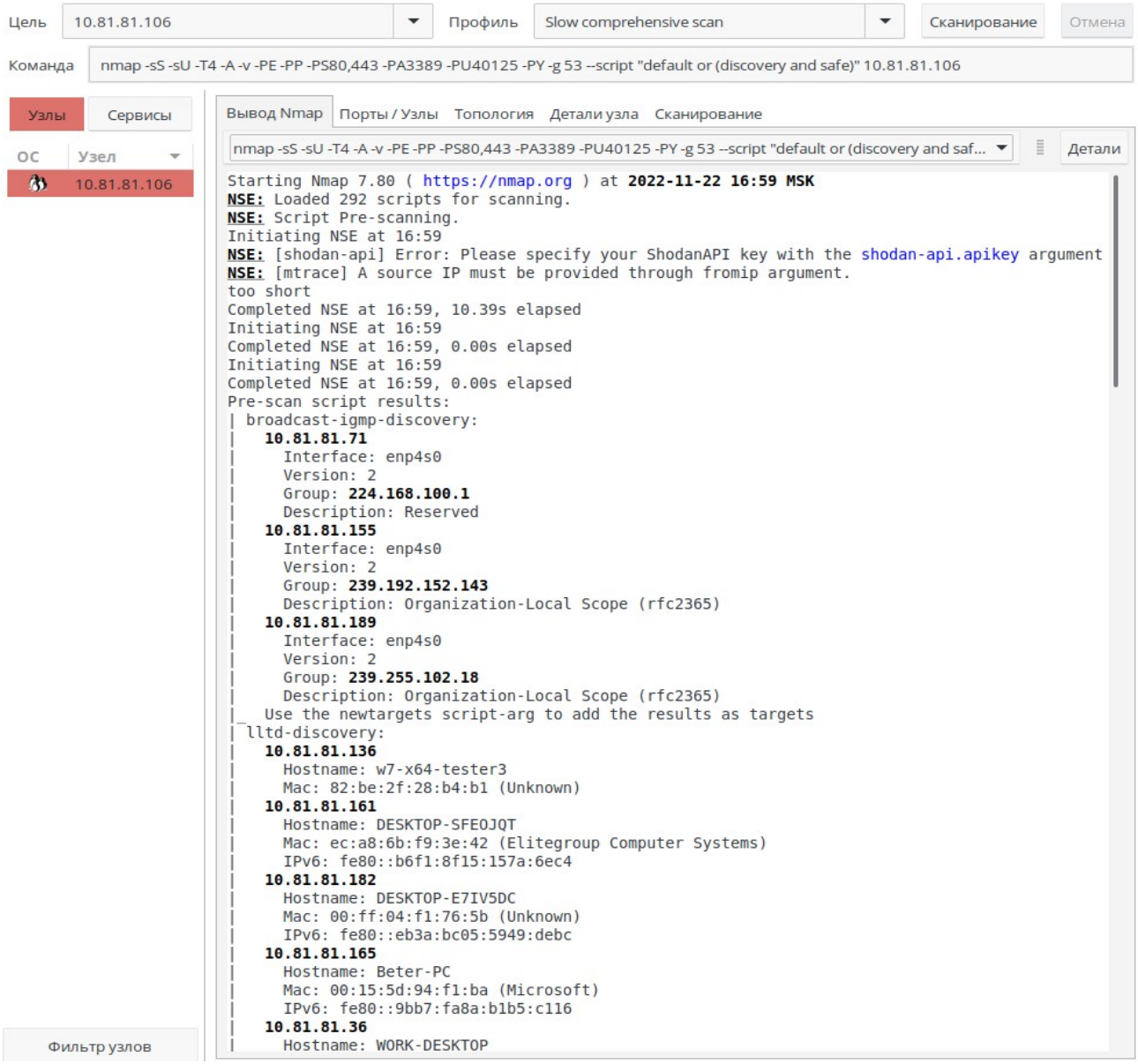
-PA<№_порта> - TCP ACK-пинг заданных хостов;

-PU<№_порта> - UDP-пинг заданных хостов;

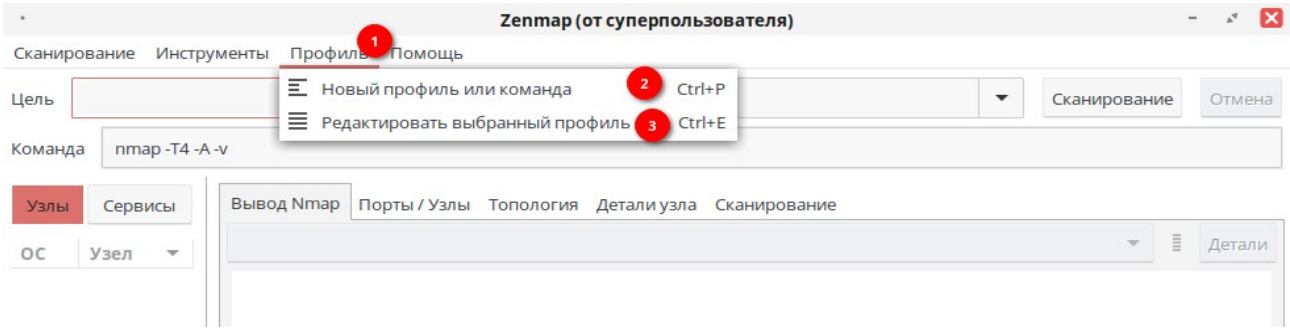
-PY – похоже на сканирование TCP SYN, только вместо этого используется SCTP;

-g – указывается, какой исходный порт вы хотите использовать. Используется при попытке обойти систему обнаружения вторжений (IDS);

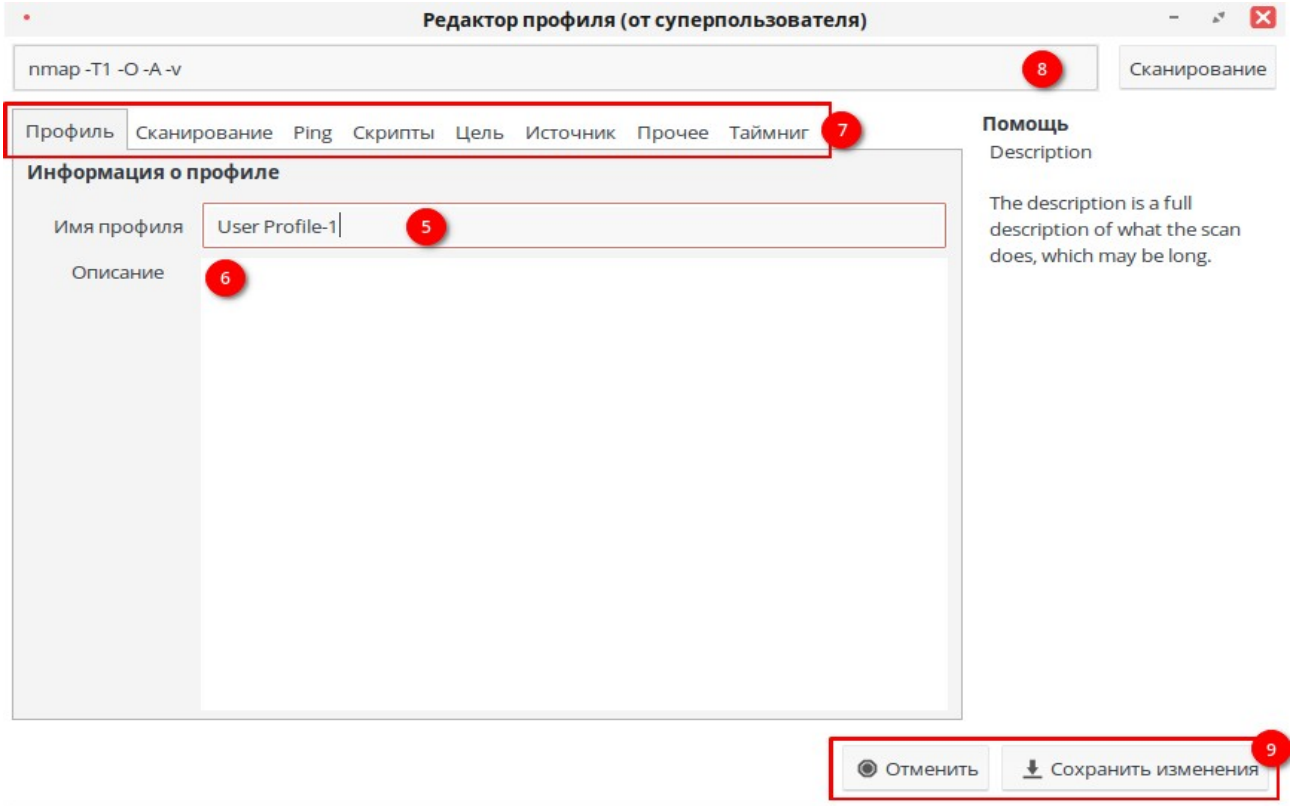
--script – сканирование с использованием скриптов.



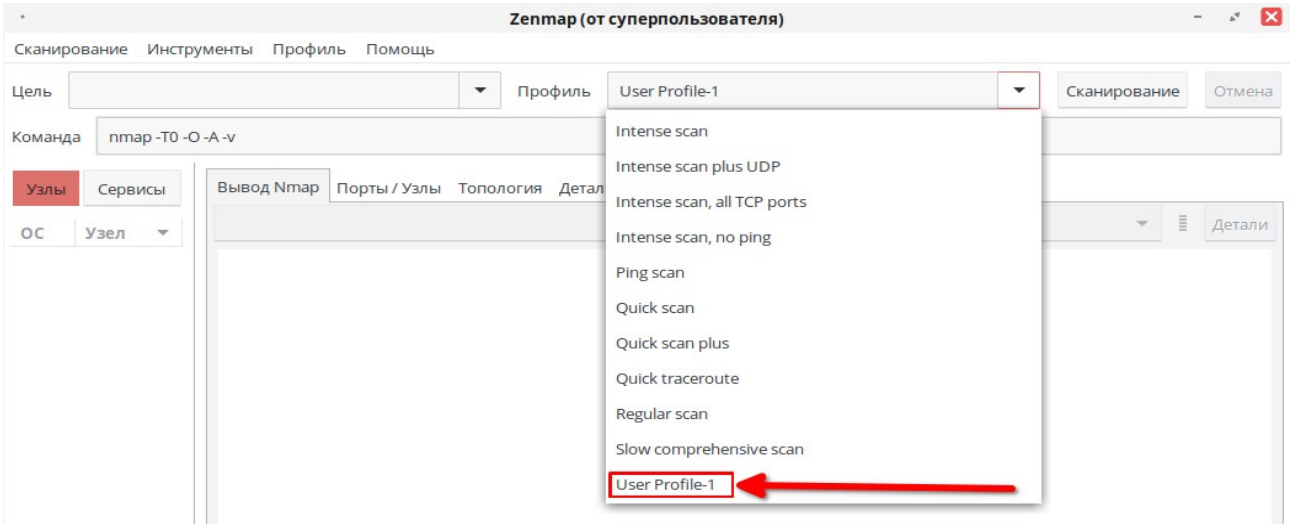
В zenmap есть возможность создавать и редактировать пользовательские профили. Для этого необходимо выбрать во вкладке «Профиль» (1) соответствующий пункт меню: «Новый профиль или команда» (2) или «Редактировать выбранный профиль» (3).



В окне редактора указывается имя профиля (5). Пользователь может при необходимости добавить описание к своему профилю (6). После указания всех необходимых параметров настройки (7) в отдельном поле (8) формируется команда для последующего сканирования сети. В этом же поле можно сразу указать нужную команду для выполнения сканирования без последовательной настройки. После настройки можно сохранить полученный профиль или отменить все изменения (9).



Во вкладке «Профиль» можно выбрать сохранённый профиль сканирования.


Zenmap (от суперпользователя)
Сканирование
Инструменты
Профиль
Помощь
Цель
Профиль
User Profile-1
Сканирование
Отмена
Команда
nmap -T0 -O -A -v
Узлы
Сервисы
ОС
Узел
Вывод Nmap
Порты / Узлы
Топология
Детали
Intense scan
Intense scan plus UDP
Intense scan, all TCP ports
Intense scan, no ping
Ping scan
Quick scan
Quick scan plus
Quick traceroute
Regular scan
Slow comprehensive scan
User Profile-1

Глава 10. Удалённое управление в РЕД ОС

10.1. SSH

SSH — это специальный сетевой протокол, позволяющий получать удаленный доступ к компьютеру с большой степенью безопасности соединения.

В основном, SSH реализован в виде двух приложений — *SSH-сервера* и *SSH-клиента*. При подключении клиент проходит процедуру авторизации у сервера и между ними устанавливается зашифрованное соединение.

По умолчанию сервер и клиент **OpenSSH** уже установлен в РЕД ОС. Однако, при необходимости, его можно установить из терминала командой:

```
sudo dnf install openssh
```

Для остановки и запуска сервера SSH используйте команды:

```
sudo systemctl stop sshd
sudo systemctl start sshd
```

Основной файл конфигурации SSH-сервера — файл **/etc/ssh/sshd_config**, доступный для чтения или редактирования только суперпользователю. После каждого изменения этого файла необходимо перезапустить ssh-сервер для применения таких изменений командой:

```
sudo systemctl restart sshd
```

Сам по себе, неправильно настроенный SSH-сервер — огромная уязвимость в безопасности системы, т. к. у потенциального злоумышленника есть возможность получить практически неограниченный доступ к системе. Помимо этого, у *sshd* есть много дополнительных полезных опций, которые желательно включить для повышения удобства работы и безопасности.

`Port, ListenAddress и AddressFamily`

Эти три параметра определяют, на каких портах и адресах ваш сервер будет ждать входящие соединения. Во-первых, имеет смысл по возможности ограничить семейство обрабатываемых адресов реально используемыми, т. е. если вы используете только IPv4 — отключите IPv6, и наоборот. Сделать это можно при помощи параметра **AddressFamily**, например (для разрешения IPv4 и запрета IPv6):

```
AddressFamily inet
```

Во-вторых, если ваш ssh-сервер имеет выход в интернет, либо находится в незащищённой сети, желательно сменить стандартный порт (22), на котором слушает sshd. Это связано с тем, что многочисленные сетевые сканеры постоянно пытаются соединиться с 22-м портом и, как минимум, получить доступ путем перебора логинов/паролей из своей базы. Даже если у вас и отключена парольная аутентификация — эти попытки сильно засоряют журналы и (в большом количестве) могут негативно повлиять на скорость работы ssh-сервера. Если же вы по какой-либо причине не желаете изменить стандартный порт, вы можете использовать различные внешние утилиты для борьбы брутфорсерами, например *fail2ban*. Задать порт можно как абсолютным значением для всех интерфейсов при помощи директивы **Port**, так и конкретным значением для каждого интерфейса, при помощи директивы **ListenAddress**. Например:

```
Port 2002
```

или

```
ListenAddress 192.168.0.1:2003
ListenAddress 192.168.1.1:2004
```

После того, как был задан порт, в любой из приведённых выше опций, необходимо предупредить систему безопасности операционной системы о внесённых изменениях. Если этого не сделать, то, при попытке перезапуска ssh-сервера, SELinux заблокирует его работу. Для этого используем команду:

```
sudo semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
```

Если задан порт, отличный от стандартного, строка подключения будет иметь вид:

```
ssh -p 2002 user@server
```

Параметр MaxAuthTries прописывается в файле **/etc/ssh/sshd_config** и указывает максимальное число попыток аутентификации, разрешенное для одного соединения. Как только число неудачных попыток превысит половину заданного значения, все последующие попытки будут заноситься в журнал. При достижении максимума сеанс аутентификации разрывается.

LoginGraceTime - Как и предыдущий параметр, указывается в файле настройки ssh-сервера. Он задаёт время ожидания регистрации пользователя в

системе. Если пользователь не успел войти в систему в течение отведенного данной директивой времени, сеанс обрывается.

AllowUsers user - В файле настроек **/etc/ssh/sshd_config** может быть прописан список пользователей, разделённых пробелом, которым можно пользоваться sshd. По умолчанию — не указан = разрешено всем. Т.е. если указан хотя бы один пользователь, ssh доступ к серверу доступен только для него.

AllowGroups users -Список шаблонов имён групп через пробел. Если параметр определён, регистрация в системе разрешается только пользователям, чья главная или вспомогательная группы соответствуют какому-либо из шаблонов. Допустимы только имена групп; числовой идентификатор группы не распознаётся. По умолчанию разрешена регистрация в системе для членов всех групп. Разрешающие/запрещающие (allow/deny) директивы обрабатываются в следующем порядке: DenyUsers AllowUsers DenyGroups AllowGroups.

Учетная запись root часто является целью хакеров. Доступ к учетной записи через root по протоколу SSH на сервере Linux, который работает в сети или в Интернете, может представлять серьезную угрозу безопасности. По умолчанию root-доступ по паролю разрешён. Если есть пользователь, который имеет возможность решать все административные задачи через sudo, то рекомендуется совсем отключить root-доступ по ssh. Отключить эту опцию можно так:

```
PermitRootLogin no
```

Разрешенная по умолчанию парольная аутентификация является практически самым примитивным способом авторизации в sshd. С одной стороны, это упрощает конфигурацию и подключение новых пользователей (пользователю достаточно знать свой системный логин/пароль), с другой стороны пароль всегда можно подобрать, а пользователи часто пренебрегают созданием сложных и длинных паролей. Специальные боты постоянно сканируют доступные из интернета ssh-сервера и пытаются авторизоваться на них путем перебора логинов/паролей из своей базы. Настоятельно не рекомендуется использовать парольную аутентификацию. Отключить ее можно так:

```
PasswordAuthentication no
```

Если по каким-либо причинам вам все-таки хочется использовать парольную аутентификацию — позаботьтесь о том, чтобы никто не мог

авторизоваться с пустым паролем. Для этого задайте директиву **PermitEmptyPasswords**:

```
PermitEmptyPasswords no
```

Наиболее предпочтительным способом авторизации является аутентификация на основе SSH2 RSA-ключей. При таком способе пользователь генерирует на своей стороне пару ключей, из которой один ключ является секретным, а другой публичным. Публичный ключ копируется на сервер и служит для проверки идентичности пользователя. Более подробно про создание пары ключей и способы размещения их на сервере см. в описании SSH-клиента. Включить аутентификацию по публичному ключу можно так:

```
PubkeyAuthentication yes
```

Сервер должен знать, где ему следует искать публичный ключ пользователя. Для этого применяется специальный файл **authorized_keys**.

Можно указать как один общий файл с ключами, так и по файлу на каждого пользователя. Последний способ является более удобным и безопасным, т. к. можно, во-первых, указывать разные комбинации ключей для каждого пользователя, а во-вторых, ограничить доступ к публичному ключу пользователя. Задать файл с ключами можно при помощи директивы **AuthorizedKeysFile**:

```
AuthorizedKeysFile /etc/ssh/authorized_keys
```

для схемы с общим файлом, или

```
AuthorizedKeysFile .ssh/authorized_keys
```

для схемы пользователь — файл.

По умолчанию SSH-клиент ищет ключи в файле **.ssh/authorized_keys**, т. е. по схеме пользователь-файл.

Перейдем к настройке клиента.

Наиболее безопасным считается вход по ключу, и в большинстве случаев на стороне сервера такая возможность включена, так что для её использования никаких прав суперпользователя не требуется. На клиентской машине генерируем ключ:

```
ssh-keygen -t rsa
```

Получаем предложение ввести пароль для защиты файла ключа (оказывается полезным при попадании файла в чужие руки). Если мы

собираемся по SSH выполнять скрипты, то оставляем пароль пустым. Передаём публичный ключ на сервер командой:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@server
```

Всё, можно заходить:

```
ssh user@server
```

Когда ssh работает на нестандартном порту:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub -p port user@server
```

Если возникает ошибка, попробуйте взять параметры в кавычки:

```
ssh-copy-id '-i /home/user/.ssh/id_rsa.pub -p port user@server'
```

Завершающий этап настройки: отключение парольной аутентификации на сервере, см. «Парольная аутентификация».

Копирование файлов по зашифрованному каналу можно выполнять при помощи утилиты SCP. С помощью этой консольной утилиты можно перемещать файлы между удалённой и локальной системой.

Копирование файла на удалённую машину:

```
scp /srcfolder/file1 user@server:/destfolder/file1
```

Копирование файла на локальную машину с удалённой:

```
scp user@server:/srcfolder/file1 /destfolder/file1
```

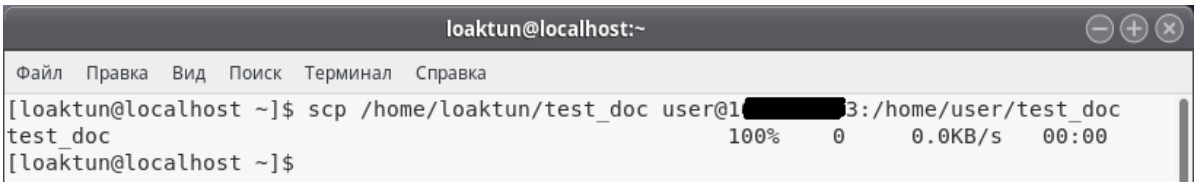
При копировании файлов нужно учитывать, что если на удалённой машине уже существует файл с таким же именем, то после выполнения команды он будет перезаписан.

Доступные опции :

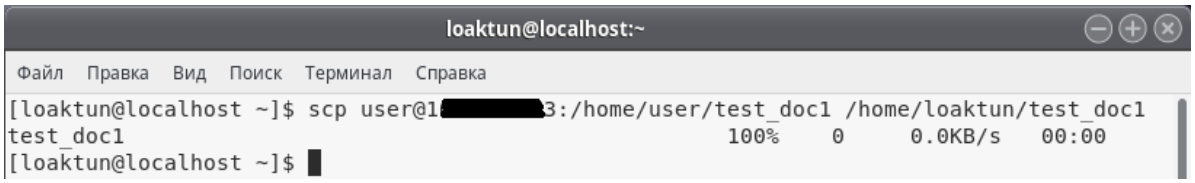
```
C : сжимать данные при их отправке на конечный компьютер.
r : рекурсивно копировать каталоги.
```

Пример.

Копирование на удалённую машину:



Копирование из удалённой системы в локальную:



10.2. Ansible

Ansible — система управления конфигурациями, написанная на Python, с использованием декларативного языка разметки для описания конфигураций. Используется для автоматизации настройки и развертывания программного обеспечения. Обычно используется для управления Linux-узлами, но Windows также поддерживается. Поддерживает работу с сетевыми устройствами, на которых установлен Python версии 2.4 и выше по SSH или WinRM соединению.

Для установки Ansible открываем терминал в режим суперпользователя командой:

```
su -
```

и выполните команду:

```
dnf install ansible
```

В файле /etc/ansible/hosts прописать все хосты, на которые будет распространяться конфигурация. Хосты можно разделить по группам, а так же, если у вас есть домен, то автоматически экспортировать список из домена. Можно прописывать как ip адреса так и имена хостов, если они резолвятся DNS ом в сети. Для теста пропишем 2 хоста.

```
[test] 10.10.1.111 10.10.1.74
```

Подключение к хостам осуществляется по протоколу ssh с помощью rsa ключей. Сгенерировать серверный ключ можно командой ниже. При её выполнении везде нажмите Enter.

```
ssh-keygen -C "$(whoami)@$(hostname)-$(date -I)"
```

Далее нужно распространить ключ на все подключенные хосты. Распространить ключи на хосты можно командой:

```
ssh-copy-id root@server
```

где:

root - это пользователь, от имени которого будут выполняться плейбуки;

server - IP-адрес хоста.

Пример: root@10.10.1.74 .

Пингуем удаленные хосты с помощью Ansible:

```
ansible test -m ping
test-1 | success >> {
    "changed": false, U
    "ping": "pong"
}

test-2 | success >> {
    "changed": false,
    "ping": "pong"
}
```

Где test - это группа хостов, указанная в файле hosts. В результате под каждым хостом должно быть написано `"ping": "pong"`.

Для работы с Ansible обычно используются плейбуки. Плейбуки являются сценариями, выполняемыми на удаленных хостах. Создаем каталог, для хранения наших playbooks, которые пишутся на языке YAML:

```
mkdir /etc/ansible/playbooks
```

Для примера создадим плейбук, устанавливающий программы. В каталоге /etc/ansible/playbooks создаем файл install_programm.yml:

```
touch /etc/ansible/playbooks/install_programm.yml
```

Содержимое файла следующее:

```
---
- hosts: all
tasks:
- name: Install programm
dnf:
name: "{{ packages }}"
vars:
packages:
- unrar
- p7zip
```

В данном примере для установки используется модуль dnf. Применяется массив packages для установки. В него вписываются необходимые для установки программы.

Если хотите, можно проверить, на каких хостах будет происходить работа, командой:

```
ansible-playbook   /etc/ansible/playbooks/install_programm.yml   --
list-host
```

Запустить только что созданный набор инструкций можно следующей командой:

```
ansible-playbook /etc/ansible/playbooks/install_programm.yml
```

С помощью Ansible возможно сразу на всех клиентах выполнить команду bash без создания плейбуков. Просмотрим информацию об использовании оперативной памяти на удаленных хостах:

```
 ansible test -a "free -h"
 test-1 | success | rc=0 >>
               total        used        free      shared    buffers
cached
 Mem:           7.6G        6.4G        1.2G        471M         64M
1.2G
 -/+ buffers/cache:       5.2G       2.4G
 Swap:          4.0G       616M       3.4G

 test-2 | success | rc=0 >>
               total        used        free      shared    buffers
cached
 Mem:           3.9G        3.3G        573M        333M        4.8M
442M
 -/+ buffers/cache:       2.9G       1.0G
 Swap:          4.0G       1.7G       2.3G
```

Пакеты Ansible версии 6.X располагаются в подключаемом репозитории, поэтому порядок обновления несколько отличается от обычного.

Установка и обновление пакетов Ansible до версии 6.X на РЕД ОС 7.3 производится с помощью следующих команд (потребуются права пользователя root):

```
 dnf install ansible6-release    # подключение репозитория
 dnf clean all     # очистка метаданных
 dnf makecache     # загрузка метаданных из всех репозиториев
 dnf install ansible     # установка новой версии ansible 6.4
```

10.3. X2go

X2Go — это программное обеспечение для удаленного рабочего стола с открытым исходным кодом, использующее модифицированный протокол NX 3. X2Go предоставляет удаленный доступ к графическому пользовательскому интерфейсу системы.

В данной статье раскрываются основные моменты по настройке данного варианта удаленного подключения, однако вы так же можете обратиться к официальной документации.

Для установки x2go перейдите в сеанс пользователя root:

```
su -
```

и выполните команду:

```
dnf install x2goserver-xsession x2goserver-fmbindings x2goserver-common x2goserver x2goagent x2goserver-desktopsharing -y
```

Здесь и далее команды будут выполняться от пользователя root, если не указано иное.

Для установки x2go на клиенте выполните команду:

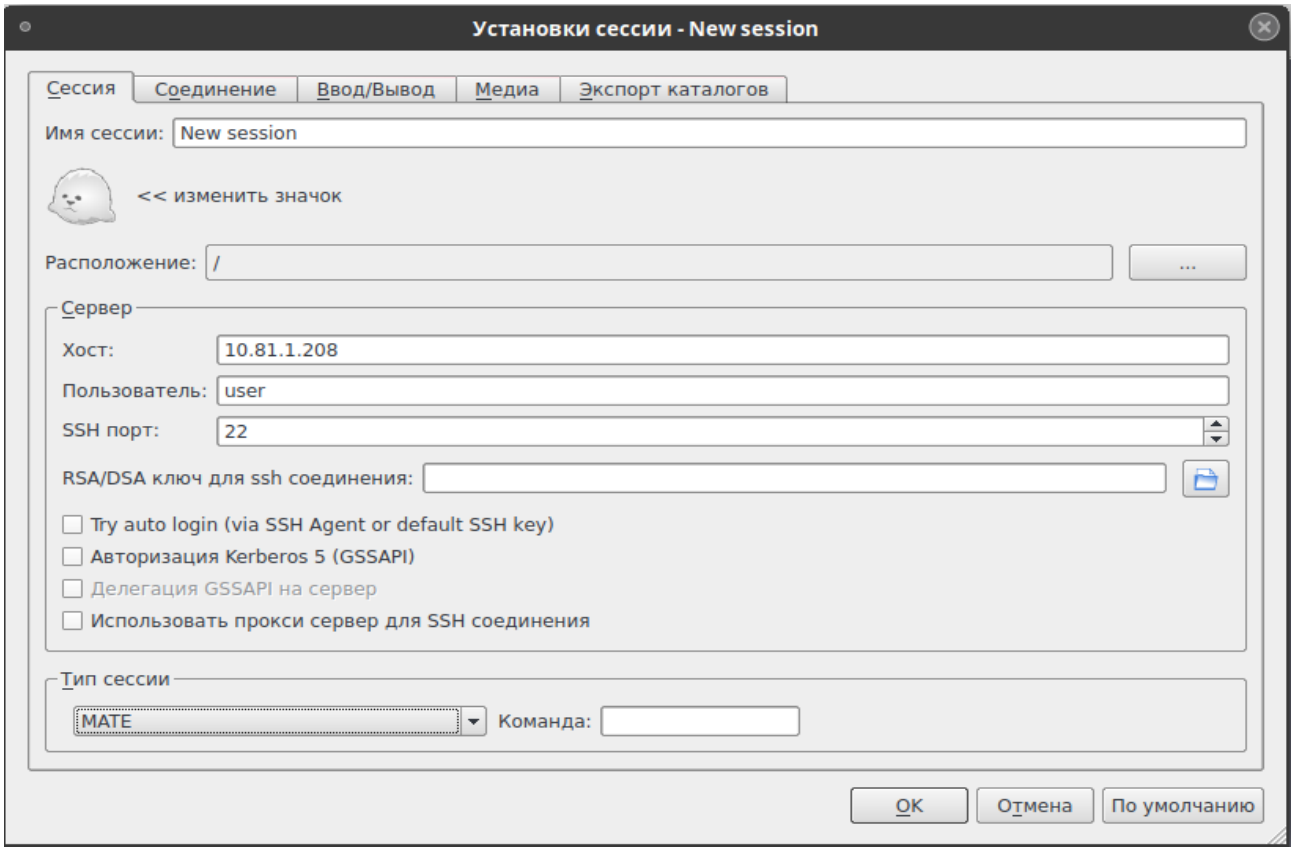
```
dnf install x2goclient -y
```

1. Откройте x2go client из «Главного меню» - «Интернет» - «X2Go Client».
2. После открытия потребуется настроить параметры:

Хост: IP-адрес машины;

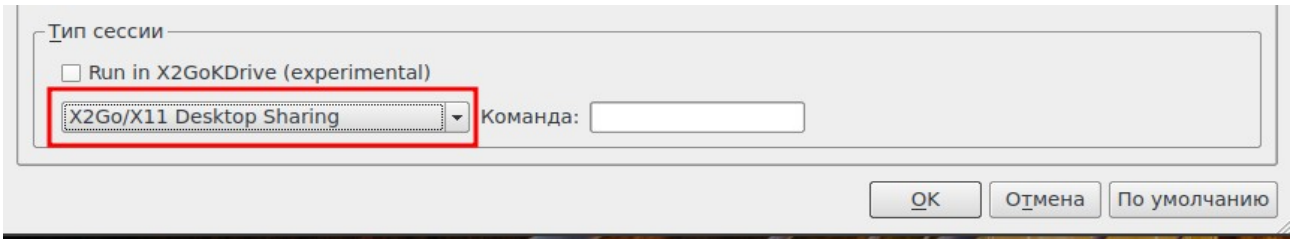
Пользователь: имя учетной записи.

SSH-порт: если не менялся на клиенте, оставить 22.



3. Для создания новых сессий при подключении к серверу требуется выбрать тип сессии Mate. В РЕД ОС 7.3 по-умолчанию используется графическая оболочка Mate. Если же у вас используется другая DE, то тип сессии изменяется в соответствии с нужным требованием.

Если необходимо соединение с активной сессией пользователя, то выберите тип сессии «X2Go/X11 Desktop Sharing».



4. Обратите внимание, что можно настроить автоподключение, указав RSA/DSA ключ через выбор файла ключа.

Для этого необходимо выполнить следующие действия:

На клиенте необходимо сгенерировать ключ SSH командой ssh-keygen (с правами локального пользователя):

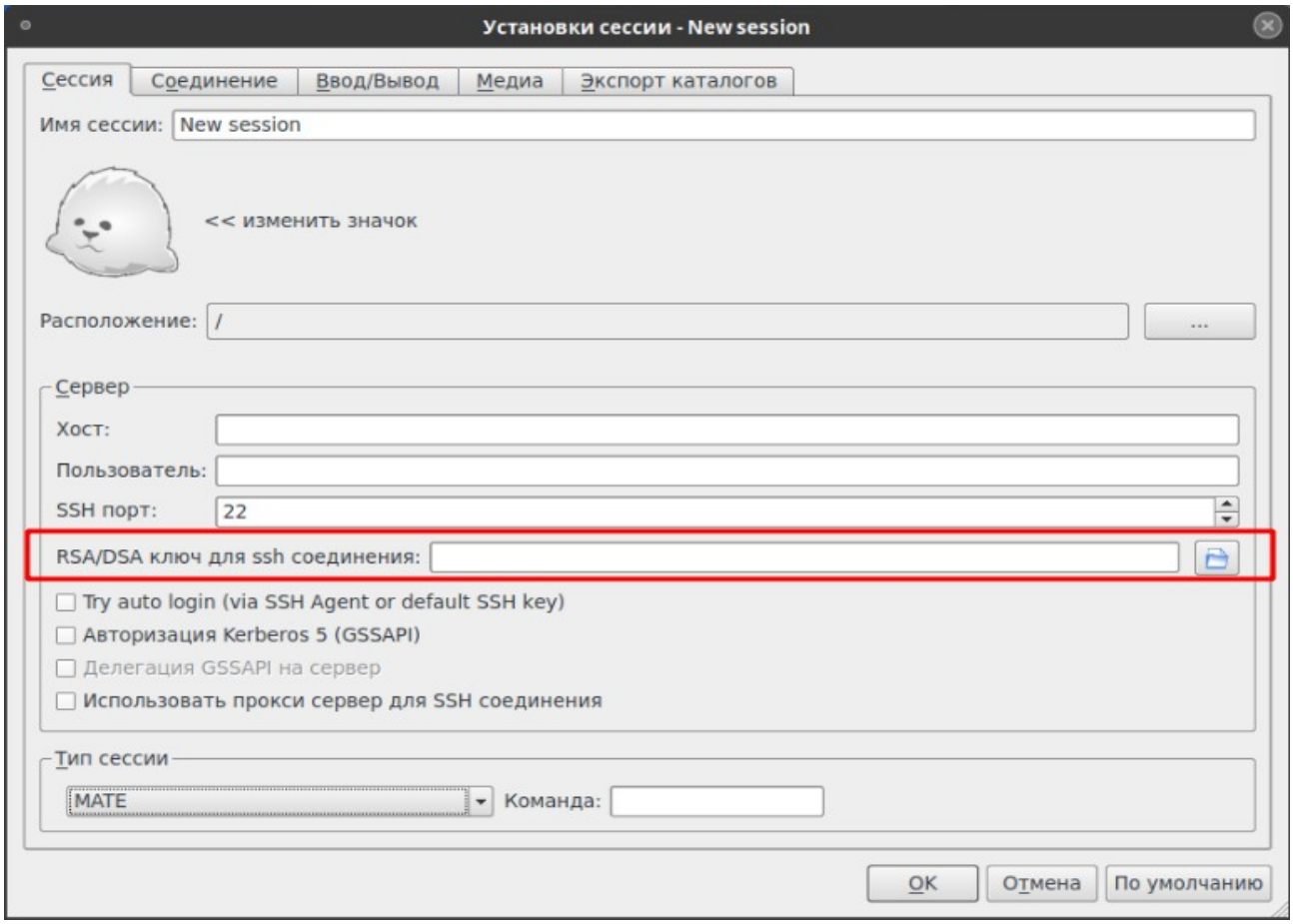
```
ssh-keygen -t rsa
  или
ssh-keygen -t dsa
```

Будет выведено предложение ввести пароль для защиты файла ключа. Если вы планируете выполнять скрипты по SSH, оставьте пароль пустым.

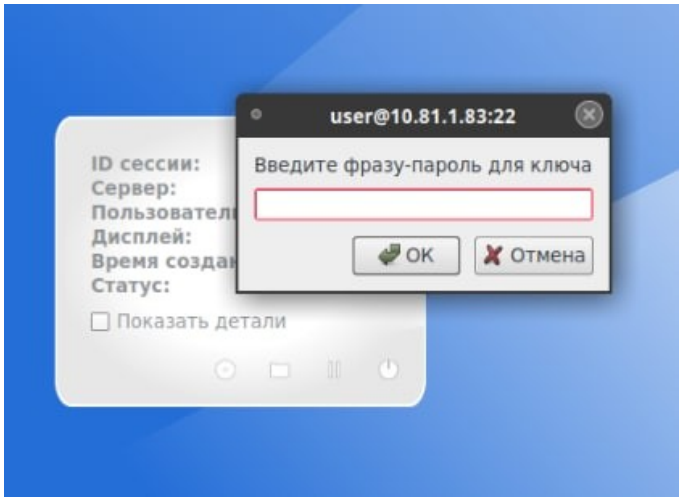
Передайте публичный ключ на сервер командой от локального пользователя:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@server
  или ssh-copy-id -i ~/.ssh/id_dsa.pub user@server
```

В x2go client выберите приватную часть ключа.



При подключении произойдет запрос фразы-пароля для ключа.



Для того чтобы настроить автоматический вход без пароля, необходимо установить флажок в поле «Try auto login (via SSH Agent or default SSH key)».

10.4. FreeRDP

freerdp - клиент с открытым кодом для подключения к удаленному рабочему столу по протоколу RDP.

В РЕД ОС имеется две различные сборки пакета freerdp:

freerdp — не имеющий поддержки Kerberos;

freerdp-krb — пакет с поддержкой Kerberos и GSS API, может оказаться необходимым для работы с Windows Server 2016, 2019 или 2022.

Для установки пакета без поддержки Kerberos выполните команду:

dnf install freerdp

Для установки пакета с поддержкой Kerberos и GSS API выполните команду:

dnf install freerdp-krb

В случае, если выбранная для работы версия сборки не подошла и возникают проблемы с подключением, можно заменить уже установленный пакет на другой.

Описание часто используемых параметров:

Параметр	Описание
/u:	Имя пользователя
/d:	Имя домена
/v:	ip-адрес или имя хоста
/p:	Пароль
/w: /h:	Значения ширины и высоты
/smartcard:	Проброс токенов
/a:printer	Подключение принтера
/cert-ignore	Игнорировать сертификат
/app:	Запуск удаленного приложения
/drive:	Подключение каталога
/sound:	Перенаправление звука
/microphone:	Активация микрофона

Варианты использования freerdp для подключения к удаленному рабочему столу:

Подключение папки

```
xfreerdp /u:bob /d:win.redos /v:10.81.1.196 /cert-ignore
/sec:rdp /p:my_password /drive:folder,/home/bob@win.redos/temp
```

Подключение принтера

Названия подключенных принтеров можно вывести командой lpstat -a.

```
xfreerdp /u:bob /d:win.redos /v:10.81.1.196 /cert-ignore
/sec:rdp /p:my_password /a:printer,"Kyocera-Kyocera-ECOSYS-
P2335dn-(KPDL)"
```

Подключение токена

С помощью pcsc_scan узнаем имя токена, например, Reader 0: Aladdin R.D. JaCarta 00 00 достаточно указать первое вхождение в имени Aladdin.

```
xfreerdp /u:user /p:my_password /smartcard:Aladdin /v:10.81.1.196
```

Перенаправление звука и микрофона в сессию

```
xfreerdp /v:10.81.1.196 /u:user /p:my_password
/sound:sys:oss,dev:1,format:1 /sound:sys:alsa
/microphone:sys:oss,dev:1,format:1 /microphone:sys:alsa
```

где sys:alsa - явное указание какой звуковой поддержки выбрать (alsa/pulse).

Подключение к сессии с явным указанием размеров окна

```
xfreerdp /v:10.81.1.196 /u:user /p:my_password /w:800 /h:600
```

Данное явное указание полезно в тех случаях, когда стандартное разрешение не умещается на мониторе.

Также существует утилита xfreerdp_gui - графическая оболочка для работы с freerdp

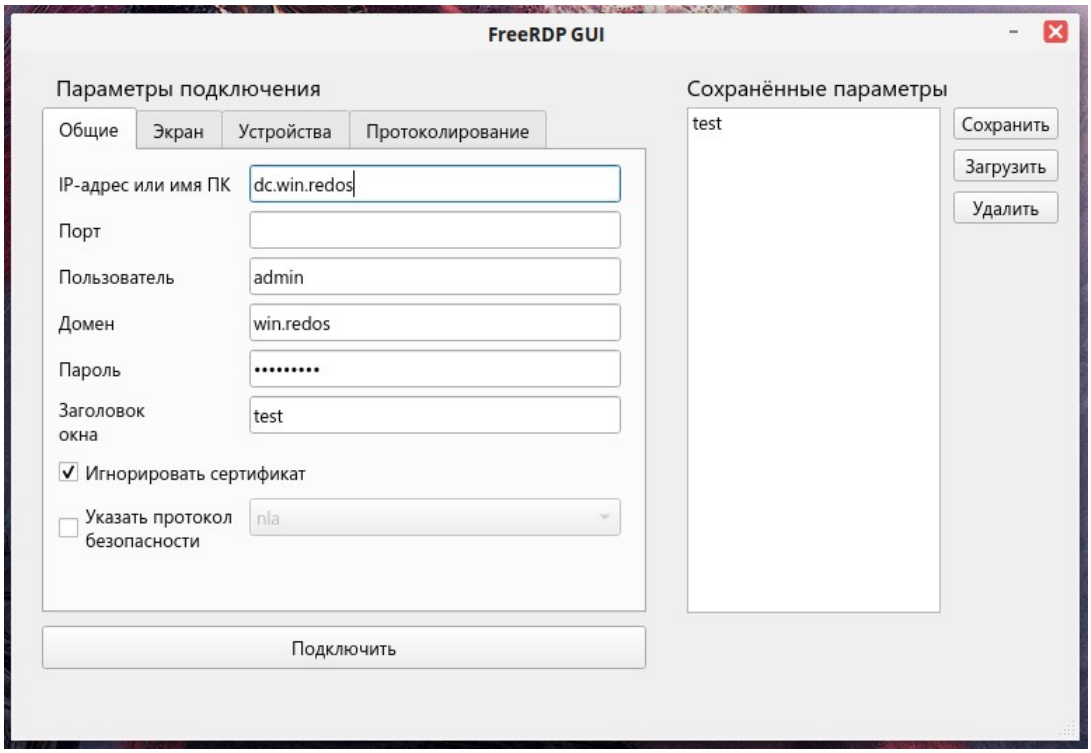
Данная утилита устанавливается в РЕД ОС 7.3 следующей командой:

```
sudo -E dnf install xfreerdp_gui
```

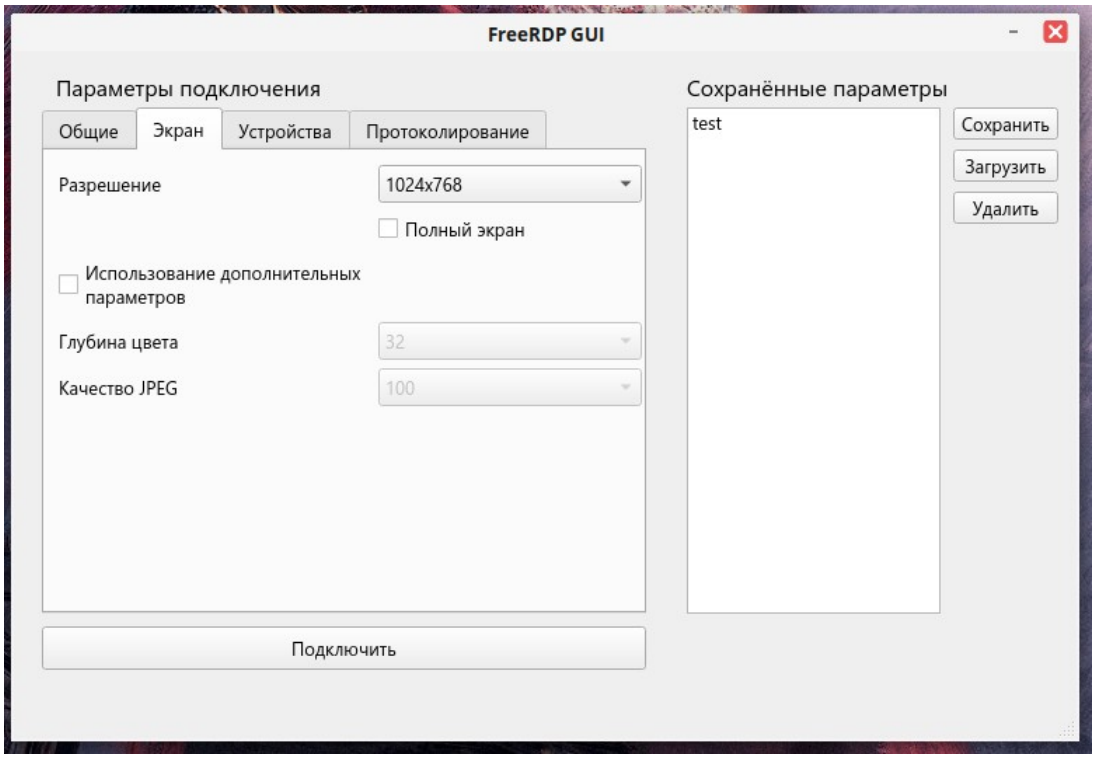
Запуск утилиты доступен из «Главного меню» - «Интернет» - «Оболочка для FreeRDP».

Главное окно утилиты представляет функционал по подключению к xrdp или Windows RDP серверу.

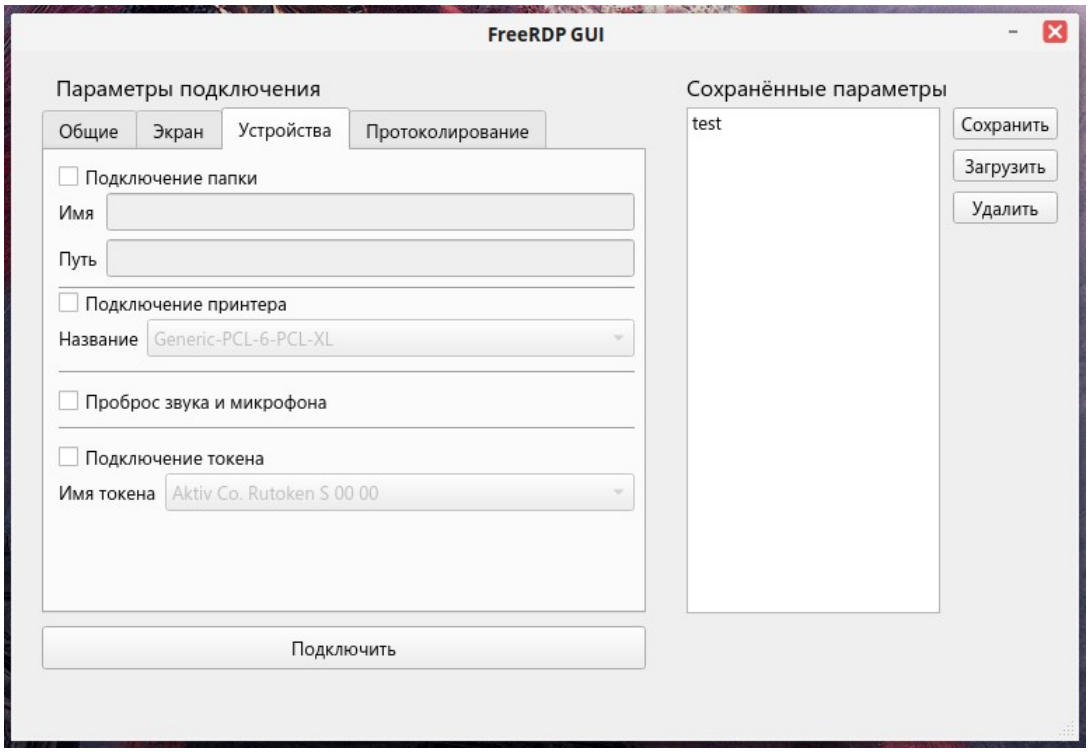
Пользователь может задать для своих целей параметры, представленные на снимке основного окна утилиты.



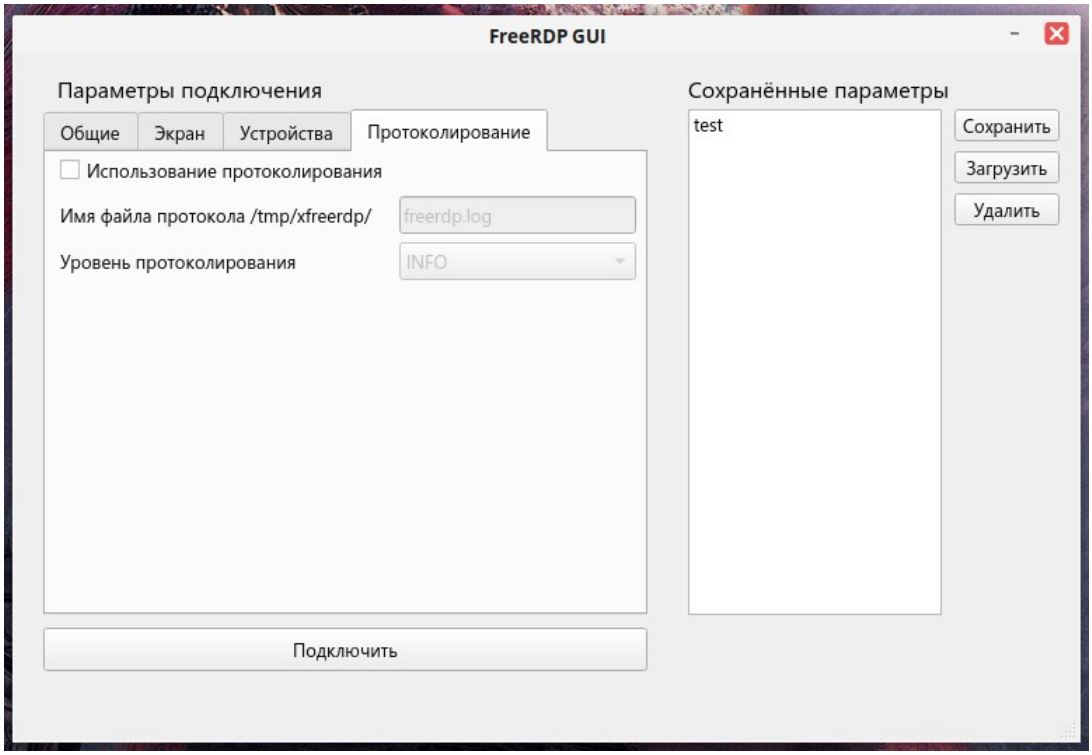
Для выбора необходимого разрешения экрана и дополнительных параметров предусмотрена вкладка «Экран».



Если требуется работа с пробросом папки или устройств, таких как принтер, токен или звук на серверную часть - для этого предусмотрена вкладка «Устройства».



Для настройки протоколирования событий предусмотрена вкладка «Протоколирование».



10.5. Remmina

Remmina — клиент удалённого рабочего стола, имеет лицензию GPLv2+. Обычно используется для подключения к удаленному рабочему столу

Windows (RDP), но также Remmina имеет возможности подключения с использованием следующих протоколов: SSH, VNC, RDP, NX и XDMCP.

Для установки данного приложения выполните команду:

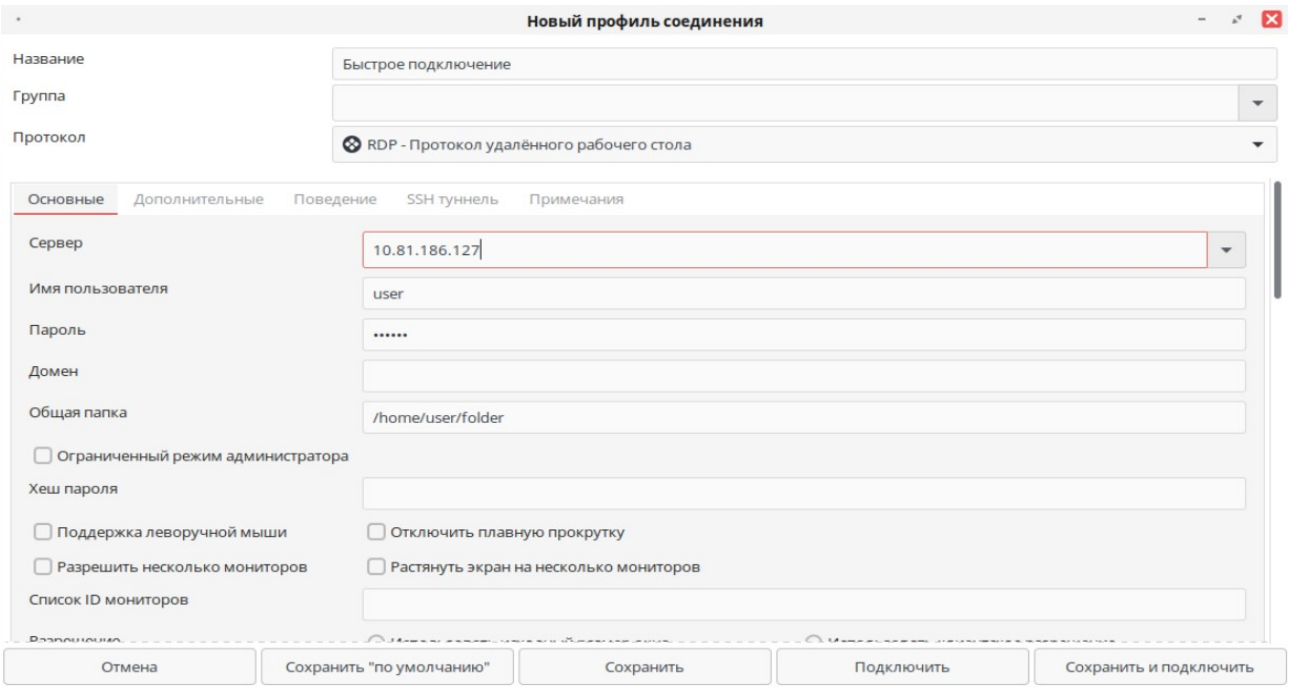
```
dnf install remmina -y
```

При скачивании *Remmina* будут доступны следующие плагины: EXEC, NX, RDP, RDPF, RDPS, SFTP, SPICE, SSH, ST, VNC, VNCI, XDMCP, glibsecret.

Для расширенной работы с приложением можно установить оставшиеся плагины следующей командой (с правами пользователя root):

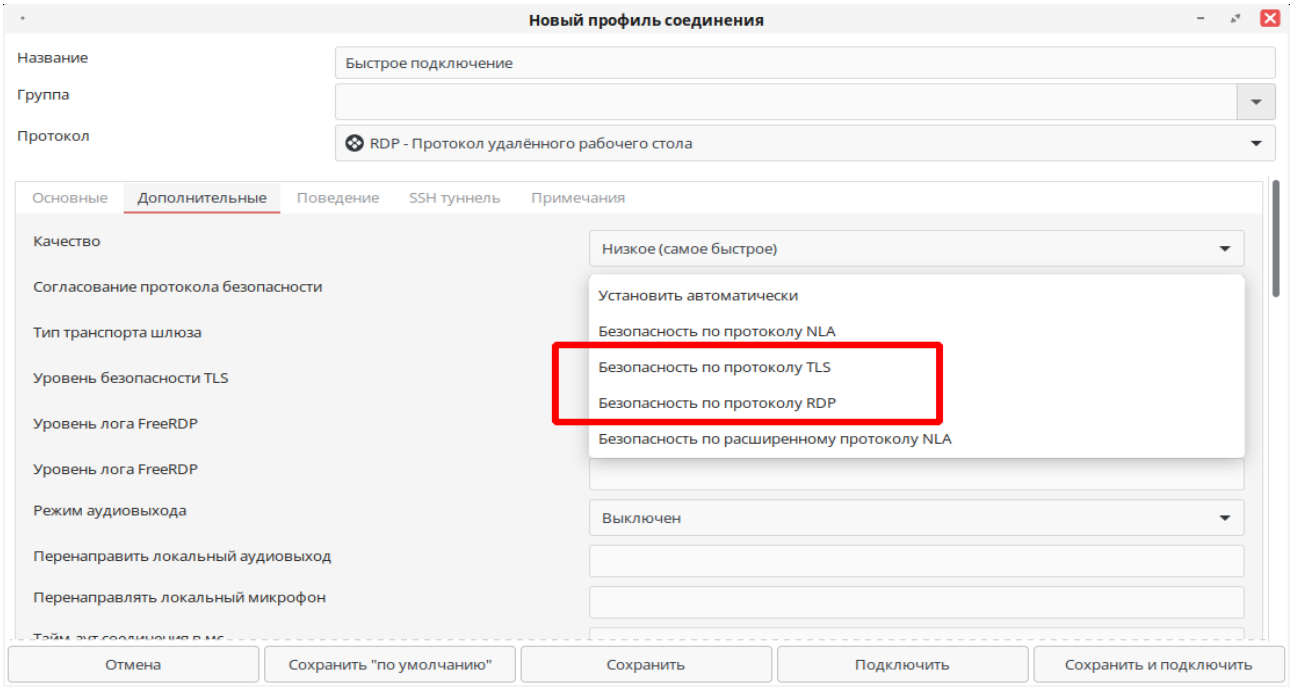
```
dnf install remmina-gnome-session remmina-plugins-kwallet remmina-plugins-spice remmina-plugins-www -y
```

Для установки соединения нажмите на кнопку «Добавить новый профиль соединения». Откроется окно создания соединения, где вам нужно выбрать протокол RDP и заполнить поля «Сервер», «Имя пользователя» и «Пароль».



После заполнения полей нажмите на кнопку «Сохранить» - для сохранения настроек или на кнопку «Сохранить и подключить» - для сохранения настроек и подключения к удаленному рабочему столу.

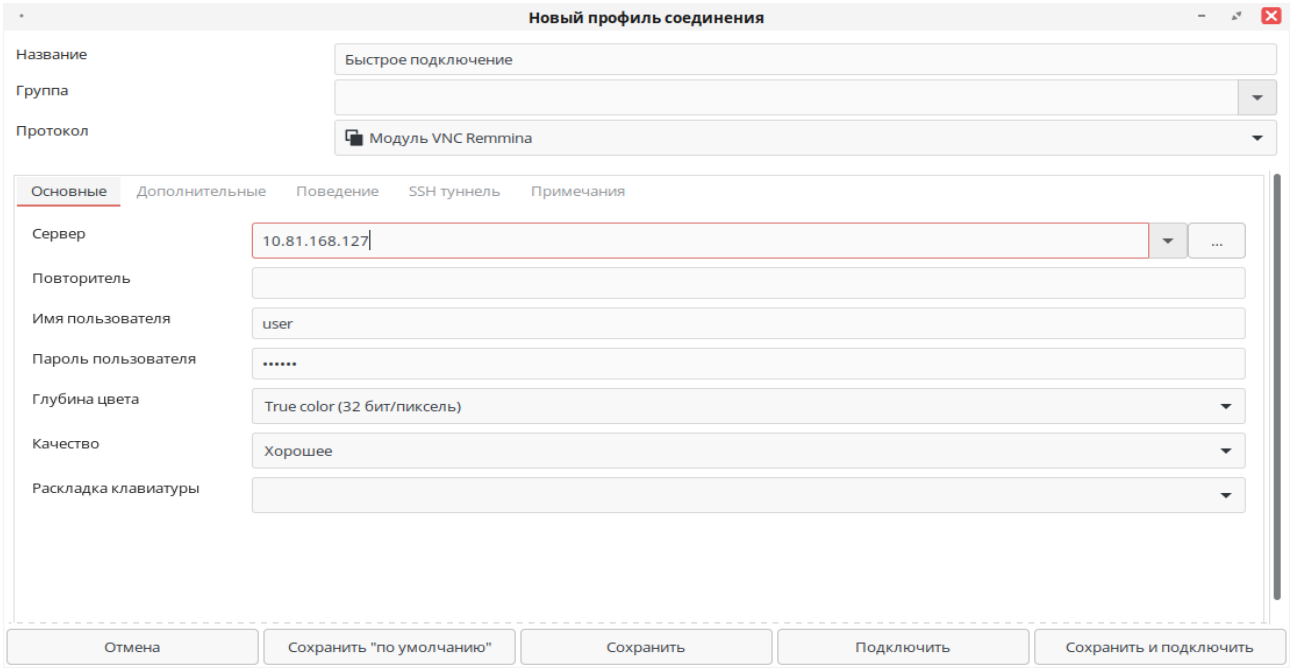
Если возникла проблема с закрытием Remmina при повторном подключении к windows-серверу по протоколу RDP, рекомендуется в настройках соединения на вкладке «Дополнительные» для параметра Согласование протокола безопасности использовать «Безопасность по протоколу TLS» или «Безопасность по протоколу RDP».



Для подключения по протоколу безопасности TLS может понадобится снять чекбокс «Разрешить подключение только с компьютеров, на которых работает удалённый рабочий стол с проверкой подлинности на уровне сети» в дополнительных параметрах windows-сервера.

Для подключения по протоколу безопасности RDP на windows-сервере требуется поменять регистр HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\SecurityLayer на 0.

В новом профиле соединения из выпадающего списка параметра «Протокол» выберите «Модуль VNC Remmina». Также заполните поля «Имя пользователя» и «Пароль». В поле «Сервер» впишите IP-адрес хоста вашей машины и порт VNC.



Для подключения нажмите на кнопку «Подключить» или «Сохранить и подключить».

Глава 11. Система управления контролем доступа SELinux

11.1. Механизм работы политик SELinux.

В большинстве операционных систем имеются средства управления доступом, которые определяют, может ли определенный объект (пользователь или программа) получить доступ к определенному ресурсу.

В системах UNIX® применяется разграничительный контроль доступа (discretionary access control, DAC). Этот метод позволяет ограничить доступ к объектам на основе групп, к которым они принадлежат.

Такое разграничение прав доступа может привести к возникновению ряда проблем из-за того, что программа, в которой может быть обнаружена уязвимость, наследует все права доступа пользователя. Следовательно, она может выполнять действия с тем же уровнем привилегий, какой есть у пользователя (что нежелательно).

Вместо того чтобы определять ограничения подобным образом, более безопасно использовать принцип наименьшего уровня привилегий (principle of least privilege), согласно которому программы могут делать только то, что им необходимо для выполнения своих задач, и не более того.

Linux с улучшенной безопасностью (SELinux) - это реализация принудительного управления доступом (mandatory access control — MAC) в ядре Linux, проверяющего разрешение на выполнение операций после проверки стандартного разграничительного управления доступом DAC.

SELinux создан Агенством Национальной Безопасности и вводит в действие правила для файлов и процессов в системе Linux, для совершаемых над ними действий, основываясь на установленной политике.

При использовании SELinux, файлы, включая директории и устройства являются объектами. Процессы, такие как, выполнение команды пользователем или приложение Mozilla® Firefox®, являются субъектами.

Основное назначение архитектуры MAC - это возможность принудительного назначения административно-установленной политики безопасности над всеми процессами и файлами системы, при этом решение основывается на метках, содержащих множество значимой информации по безопасности. Когда механизм SELinux реализован, он переводит систему в состоянии достаточной защищенности и предоставляет критичную поддержку

приложениям, защищая приложения от взлома или обхода безопасности.

MAC предоставляет строгое разделение приложений и позволяет безопасное исполнение не доверенных приложений. Обладая способностью ограничивать привилегии, связанные с исполнением процессов, MAC ограничивает рамки потенциальной угрозы, таким образом ограничивая взлом уязвимостей в приложениях и системных службах. MAC включает защиту информации от пользователей корректно авторизованных в системе с ограниченными правами также как и от авторизованных пользователей, которые неосознанно исполняют вредоносный код.

В операционных системах Linux с запущенным SELinux, существуют пользователи Linux и пользователи SELinux. Пользователи SELinux - это часть политики SELinux. Пользователи Linux сопоставляются (маппируются) с пользователями SELinux. Для того, чтобы избежать путаницы, в данном руководстве используются два термина "пользователь Linux" и "пользователь SELinux" для различия двух разных понятий.

В дополнение к DAC SELinux (Security Enhanced Linux) предлагает несколько вспомогательных моделей управления доступом:

Type Enforcement (TE): Основной механизм ограничения, используемый в целевых политиках. Позволяет детально, на самом низком уровне управлять разрешениями. Самый гибкий, но и самый трудоемкий для системного администратора механизм.

Role-Based Access Control (RBAC): в этой модели права доступа реализуются в качестве ролей. Ролью называется разрешения на выполнение определенных действий одним или несколькими элементами системы над другими частями системы. По-сути, RBAC является дальнейшим развитием TE.

Multi-Level Security (MLS): многоуровневая модель безопасности, в которой всем объектам системы присваивается определенный уровень доступа. Разрешение или запрет доступа определяется только соотношением этих уровней.

Multi-Category Security(MCS): Расширение MLS, используется в целевой политике для ограничения виртуальных машин и контейнеров через sVirt.

SELinux не является:

антивирусным программным обеспечением.

заменой паролям, межсетевым экранам или другим системам безопасности.

решением безопасности "всё в одном".

SELinux разработан, для усовершенствования существующих решений по безопасности. Даже с запущенным SELinux, необходимо использование практик по безопасности, таких как обновление программного обеспечения последними обновлениями, использование сложных паролей, межсетевых экранов и прочего

Основные понятия SELinux

Сущность (identity) — этот термин схож с понятием "пользователь" в классической схеме доступа. Сущность может иметь такое же название, как и логин пользователя, но в отличие от логина, сущность не меняется после выполнения команды su. Если провести аналогию, то сущность - это конкретный человек, Вася Пупкин, Петя Смирнов и т.д.

Домен (domain) — это список того, что может делать отдельный процесс. Фактически домен - это действия, минимально необходимые одному процессу для выполнения его задачи. По аналогии из реальной жизни, доменом можно назвать набор действий для совершения какой-либо операции.

Роль (role) — это список доменов, которые могут быть использованы. Если некоего домена нет в списке, то роль не может выполнить действия из этого домена. В данном случае можно провести аналогию с должностью. То есть роль - это фактически должность (или должностная инструкция), которая может выполнять определённые наборы операций, или, в понятии SELinux, домены.

Тип (type) — это набор действий (операция) применительно к объекту. Важно понять отличие от домена. Домен относится к процессам, а тип - к объектам, таким как файлы, каталоги, пайпы(pipes), сокеты и т. д.

Уровень (level) — это атрибут многоуровневого управления доступом MLS и MCS. Пространство MLS - это пара уровней, записанных в виде lowlevel-highlevel, если уровни в данной паре отличаются или, если не отличаются, то lowlevel. То есть (s0-s0 то же самое, что и s0). Если дополнительно определены категории, то уровень записывается как sensitivity:category-set. Если категории не определены, то запись выглядит как sensitivity.

Контекст безопасности (context) — это набор всех атрибутов, связанных с объектами и субъектами. Контекст безопасности для субъектов (процессов) состоит из сущности, роли, домена, чувствительности и категории. Обычно используется только сущность-роль-домен(или тип), а, например, целевая политика от Fedora использует только домены и типы.

Переход (transition) — это смена контекста безопасности. Есть два основных типа переходов:

Переход домена процесса — процесс меняет контекст; Например, запускается из-под пользователя некий демон. Selinux, на основе метки исполняемого файла, меняет его контекст.

Переход типа файла — создание файлов в определённых подкаталогах. Например, пользователь создаёт html-страничку в каталоге WEB-сервера. Чтобы WEB-сервер получил доступ к этой страничке, необходимо сменить контекст безопасности файла (WEB-сервер не имеет доступа к контексту пользователя).

Политика (policy) — это набор правил, контролирующих взаимодействие ролей, доменов, типов и т. д. Политики работают на уровне системных вызовов и обрабатываются ядром, но можно реализовать и на уровне приложения. Политики описываются при помощи специального языка описания правил доступа.

В настоящий момент уже разработано несколько готовых политик безопасности, которые можно использовать по умолчанию на серверах и на домашних компьютерах. Всё, что требуется от системного администратора - выбрать используемую политику и перезагрузить компьютер с включённым SELinux.

В среднем, политика безопасности SELinux для всей системы содержит более ста тысяч правил, так что её создание и отладка занимает значительное время.

Наиболее распространены следующие политики:

Целевая (targered). Эта политика разработана компанией Red Hat и является наиболее используемой;

Минимальная (minimum). Является модификацией целевой политики, в которой только выбранные процессы защищаются.

Многоуровневая (MLS). Позволяет обеспечивать уровни безопасности и может использоваться госструктурами для хранения информации различных уровней секретности;

Строгая (strict). Этот вариант политики подразумевает правило "Что не разрешено, то запрещено".

SELinux имеет три основных режим работы, при этом по умолчанию установлен режим Enforcing. Это довольно жесткий режим, и в случае необходимости он может быть изменен на более удобный для конечного

пользователя.

Enforcing: Режим по-умолчанию. При выборе этого режима все действия, которые каким-то образом нарушают текущую политику безопасности, будут блокироваться, а попытка нарушения будет зафиксирована в журнале.

Permissive: В случае использования этого режима, информация о всех действиях, которые нарушают текущую политику безопасности, будут зафиксированы в журнале, но сами действия не будут заблокированы.

Disabled: Полное отключение системы принудительного контроля доступа.

Команда sestatus показывает состояние SELinux.

Пример:

```
[root@sl0 ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          error (Success)
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

Также вы можете узнать статус SELinux при помощи команды getenforce.

Пример:

```
[root@sl0 ~]# getenforce
Enforcing
```

Команда setenforce позволяет быстро переключаться между режимами Enforcing и Permissive, изменения вступают в силу без перезагрузки. Но если вы включаете или отключаете SELinux, требуется перезагрузка, ведь нужно заново устанавливать метки безопасности в файловой системе.

Для установки режима по-умолчанию, который будет применяться при каждой загрузке системы, задайте значение строки SELINUX= в файле /etc/selinux/config, задав один из режимов — enforcing, permissive или disabled.

Пример:

```
[root@sl0 ~]# grep SELINUX=[a-z] /etc/selinux/config
SELINUX=enforced
```

Параметр SELINUXTYPE= в файле /etc/selinux/config указывает тип применяемой политики.

Когда целевая политика targeted используется, процессы, которые являются целевыми, запускаются в ограниченном домене, остальные процессы запускаются в неограниченном домене. Например, по умолчанию пользователи, прошедшие авторизацию, работают в домене unconfined_t и системные процессы запущенные init-ом запускаются в домене initrc_t - оба домена неограниченные.

Неограниченные домены (так же, как и ограниченные) - это субъекты для операций выполнения и записи в память. По умолчанию, субъекты запущенные в неограниченном домене не могут выделить память для записи и запустить ее. Это уменьшает степень угрозы атаки переполнения буфера buffer overflow attacks. Эти проверки памяти отключаются установкой Булевых переключателей, что позволяет изменять политику SELinux "на ходу". Настройка Булевых значений рассматривается позже.

Почти каждая сетевая служба ограничена. Также большинство процессов, которые запускаются в Linux с привилегиями пользователя root и выполняют задачи для пользователей, такие как приложение passwd, ограничены. Когда процесс ограничен, он запускается в своём собственном домене, например процесс httpd запускается в домене httpd_t. Если ограниченный процесс скомпрометирован атакующим, в зависимости от конфигурации SELinux, доступ атакующего к ресурсам и вред, который он может нанести ограничен.

Неограниченные (unconfined) процессы выполняются в неограниченных (unconfined) доменах, программы запускаемые init выполняются в неограниченном unconfined `initrc_t` домене, неограниченные процессы ядра запускаются в домене `kernel_t`. Для неограниченных процессов правила политики SELinux также применяются, но правила политики существуют для разрешения практически всех доступов для процессов, запущенных в неограниченных доменах. Процессы запущенные в неограниченных доменах откатываются к использованию только правил DAC. Если неограниченный процесс скомпрометирован, SELinux не ограничивает атакующего от получения доступа к системным ресурсам и информации, но, конечно, правила DAC всё равно используются. SELinux это улучшение механизмов безопасности над DAC, но SELinux не заменяет его.

В целевых (targeted) политиках предоставление доступа основано на анализе меток. Политика проверяет может ли домен процесса (субъекта)

получить доступ к ресурсу (объекту). SELinux перехватывает системные вызовы и разрешает доступ, если политика это позволяет.

Информация о метках находится в контексте.

Основная опция получения информации о контекстах -Z.

Пример: Мы проверим контекст файлов, которые использует демон apache (httpd). И контекст процесса веб сервера. В политиках имеется разрешение для домена httpd_t получать доступ к типам httpd_sys_content_t. Чтобы процесс получил домен httpd_t программа имеет тип httpd_exec_t.

```
[root@sl0 devel]# ls -Z /var/www/html/index.html
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0
/var/www/html/index.html
[root@sl0 devel]# ls -Z /usr/sbin/httpd
-rwxr-xr-x.    root    root    system_u:object_r:httpd_exec_t:s0
/usr/sbin/httpd
[root@sl0 devel]# ps -eZ | grep httpd | head -1
system_u:system_r:httpd_t:s0     1100 ?        00:00:00 httpd
```

В примере выше мы не видим самой политики. Политика представляет из себя бинарный файл, который во время старта системы загружается в ядро. Правила написания политик мы обсудим ниже.

Переключатели позволяют изменять части политики SELinux во время работы (без перезапуска и остановки), не обладая глубоким пониманием создания политики SELinux. Это позволяет вносить изменения, такие как: разрешение доступа службам к файловым системам NFS, без перезагрузки или рекомпиляции политики SELinux.

Для получения списка переключателей, объяснения, за что отвечает каждый переключатель, включен или выключен, необходимо выполнить команду semanage boolean -l от имени пользователя root.

Пример:

```
[root@sl0 devel]# semanage boolean -l | egrep '(SELi|
httpd.*connect_db)'
SELinux boolean                State  Default Description
httpd_can_network_connect_db   (off  ,  off)  Allow HTTPD scripts
and modules to connect to databases over the network.
```

Команда getsebool -a выводит список переключателей, показывает выключены они или нет, но не даёт описания, за что они отвечают.

Пример:

```
[root@sl0 devel]# getsebool -a | grep httpd.*connect_db
httpd_can_network_connect_db --> off
```

Для получения статуса одного конкретного Булева значения (переключателя) boolean-name используется команда getsebool boolean-name

Пример:

```
[root@sl0 devel]# getsebool httpd_can_network_connect_db
httpd_can_network_connect_db --> off
```

Команда setsebool boolean-name x переводит переключатели в состояние включено или выключено, где boolean-name - название переключателя, а x - on для включения или off для выключения.

Пример:

```
[root@sl0 devel]# setsebool httpd_can_network_connect_db on
[root@sl0 devel]# getsebool httpd_can_network_connect_db
httpd_can_network_connect_db --> on
```

Чтобы некоторый ограниченный процесс получил доступ к файлу, последний, в свою очередь, должен иметь нужный тип.

Команда chcon устанавливает временный контекст.

Пример: Создаем собственный ресурс и получаем запрет доступа. После установки правильного контекста доступ появляется.

```
[root@sl0 myweb]# pwd
/myweb
[root@sl0 myweb]# ls -Z
-rw-r--r--.    root    root    unconfined_u:object_r:default_t:s0
index.html
[root@sl0 myweb]# cat /etc/httpd/conf.d/mypage.conf
Alias /mypage /myweb
<Directory /myweb>
    AllowOverride None
    Require all granted
</Directory>
[root@sl0   myweb]#   wget   http://127.0.0.1/mypage/index.html   -O
/dev/null
--2017-08-01 21:01:26--  http://127.0.0.1/mypage/index.html
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
```

```
2017-08-01 21:01:26 ERROR 403: Forbidden.
[root@sl0 myweb]# chcon -R  -t httpd_sys_content_t /myweb
[root@sl0   myweb]#   wget   http://127.0.0.1/mypage/index.html   -O
/dev/null
--2017-08-01 21:03:13--  http://127.0.0.1/mypage/index.html
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 115877 (113K) [text/html]
Saving to: ‘/dev/null’
100%[======================================>] 115,877      --.-K/s
in 0s
2017-08-01 21:03:13 (609 MB/s) - ‘/dev/null’ saved [115877/115877]
```

Команда restorecon восстанавливает контекст по умолчанию.

Пример: Восстановление контекста приводит к запрету доступа.

```
[root@sl0 myweb]# restorecon -R /myweb
[root@sl0   myweb]#   wget   http://127.0.0.1/mypage/index.html   -O
/dev/null
--2017-08-01 21:07:12--  http://127.0.0.1/mypage/index.html
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2017-08-01 21:07:12 ERROR 403: Forbidden.
[root@sl0 myweb]# ls -Z
-rw-r--r--.    root    root    unconfined_u:object_r:default_t:s0
index.html
```

Команда semanage fcontext изменяет контекст SELinux для файлов. При использовании целевой политики targeted, изменение вносимые данной командой, добавляются в файл /etc/selinux/targeted/contexts/files/file_contexts, если изменения вносятся для существующих файлов, то они добавляются в файл file_contexts, или добавляютсяв файл file_contexts.local для новых файлов и каталогов, например при создании каталога /web/.setfiles, использующаяся при маркирование файловой системы и restorecon, использующаяся для восстановления контекста SELinux по умолчанию, читают эти файлы. Это значит, что изменения вносимые командой semanage fcontext постоянно, даже если файловая система будет перемаркирована. Политика SELinux контролирует возможность пользователей изменять контекст файлов.

Для внесения изменений в контекст SELinux изменений, которые сохранятся при перемаркировании файловой системы надо:

Выполнить команду semanage fcontext -a options file-name|directory-name Помните, что необходимо использовать полные пути к файлам и каталогам.

Выполнить команду restorecon -v file-name|directory-name для применения

изменений контекста.

Пример: Применение контекста к каталогу.

```
[root@sl0 myweb]# ls -dZ /myweb
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /myweb
[root@sl0 myweb]# ls -Z /myweb
-rw-r--r--.    root    root    unconfined_u:object_r:default_t:s0
index.html
[root@sl0  myweb]#  semanage  fcontext  -a  -t  httpd_sys_content_t
/myweb
[root@sl0 myweb]# ls -Z /myweb
-rw-r--r--.    root    root    unconfined_u:object_r:default_t:s0
index.html
[root@sl0 myweb]# ls -dZ /myweb
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /myweb
[root@sl0 myweb]# restorecon -Rv /myweb/
restorecon             reset             /myweb             context
unconfined_u:object_r:default_t:s0-
>unconfined_u:object_r:httpd_sys_content_t:s0
[root@sl0 myweb]# ls -dZ /myweb
drwxr-xr-x. root root unconfined_u:object_r:httpd_sys_content_t:s0
/myweb
[root@sl0 myweb]# ls -Z /myweb
-rw-r--r--.    root    root    unconfined_u:object_r:default_t:s0
index.html
[root@sl0                          myweb]#                          cat
/etc/selinux/targeted/contexts/files/file_contexts.local
# This file is auto-generated by libsemanage
# Do not edit directly.
/myweb    system_u:object_r:httpd_sys_content_t:s0
[root@sl0  myweb]#  semanage  fcontext  -a  -t  httpd_sys_content_t
'/myweb(/.*)?'
[root@sl0 myweb]# restorecon -Rv /myweb/
restorecon         reset         /myweb/index.html         context
unconfined_u:object_r:default_t:s0-
>unconfined_u:object_r:httpd_sys_content_t:s0
[root@sl0                          myweb]#                          cat
/etc/selinux/targeted/contexts/files/file_contexts.local
# This file is auto-generated by libsemanage
# Do not edit directly.
/myweb    system_u:object_r:httpd_sys_content_t:s0
/myweb(/.*)?    system_u:object_r:httpd_sys_content_t:s0
[root@sl0 myweb]# ls -Z /myweb/
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0
index.html
[root@sl0   myweb]#   wget   http://127.0.0.1/mypage/index.html   -O
/dev/null
--2017-08-01 21:25:44--  http://127.0.0.1/mypage/index.html
```

```
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 115877 (113K) [text/html]
Saving to: ‘/dev/null’
100%[======================================>] 115,877      --.-K/s
in 0s
2017-08-01    21:25:44    (1.33    GB/s)    -    ‘/dev/null’    saved
[115877/115877]
```

Удаление перманентной маркировки файлов производится командой semanage fcontext -d options file-name|directory-name

Пример: Удаление лишнего правила маркировки.

```
[root@sl0  myweb]#  semanage  fcontext  -d  -t  httpd_sys_content_t
/myweb
[root@sl0                          myweb]#                          cat
/etc/selinux/targeted/contexts/files/file_contexts.local
# This file is auto-generated by libsemanage
# Do not edit directly.
/myweb(/.*)?    system_u:object_r:httpd_sys_content_t:s0
[root@sl0 myweb]# restorecon -Rv /myweb
[root@sl0 myweb]# ls -Zd /myweb
drwxr-xr-x. root root unconfined_u:object_r:httpd_sys_content_t:s0
/myweb
```

11.2. Язык описания правил доступа.

SELinux это гибкий и мощный инструмент обеспечения безопасности. Основой для принятия решений в SELinux служит политика. Политику по умолчанию создают люди, которые поддерживают дистрибутив Линукса. Основу для этой политики создают разработчики программного обеспечения.

Разработчики софта и дистрибутивов, как правило, придерживаются следующих целей:

Программы должны работать. Пользы от программы, которая абсолютна защищена и при этом не может работать, нет никакой.

Обеспечение безопасности основано на общих принципах и соображениях. Это означает, что в конкретно вашей установленной системе, не учитываются все нюансы функционирования даже стандартных пакетов.

В каждом дистрибутиве имеется набор предусмотренного этим дистрибутивом пакетов. Если вы используете какой-то пакет не предусмотренный разработчиками, то для него не будет политики. В этом случае пакет или не работает или попадает в категорию неограниченных, т. е.

не защищается с помощью SELinux.

Исходя из выше изложенного можем сделать вывод, что рано или поздно у вас может возникнуть потребность в создании собственной политики. Основными причинами для этого могут служить следующие соображения:

Вы используете стандартный ограниченный сервис не так как это предусматривал разработчик. В этом случае вам придется расширять стандартную политику своими правилами.

Вы используете пакет, которого нет в стандартной поставке, или создаете собственный пакет. В таком случае вам потребуется создать свой модуль политики SELinux для данного программного обеспечения.

Вы, после оценки, существующих дефолтных политик пришли к выводу, что ограничения в ней слишком либеральные и необходимо ужесточение политик.

Перед созданием собственных политик, нужно понимать, что имеется два подхода для создания политик.

Реактивный. В этом случае вы решаете какую-то конкретную проблему, связанную с неправильной работой ограниченной программы. Как результат решения проблемы вы создаете политику SELinux и загружаете ее.

Проактивный. Вы создаете собственный ограниченный сервис и у вас имеются представления о том, что и как этот сервис должен делать. Результатом этих представлений будет созданная вами политика.

Конечно же имеется и гибридный подход, в котором, например, сначала создается политика, применяется, тестируется, на основе тестов модифицируется и снова применяется, анализируется и т. д.

Модуль SELinux это бинарный файл. Создание этого файла основано на использовании трех текстовых файлов. Не все три файла требуется создавать для модуля.

Файл .te предназначен для описания используемых типов и описания правил.

Файл .fc описывает маркировку файлом этим модулем.

Файл .if создает интерфейс взаимодействия с модулем, на который могут ссылаться другие модули. В файле описываются макросы.

На сайте https://selinuxproject.org/page/RefpolicyWriteModule можно посмотреть принципы создания модулей.

Каждый модуль .te может состоять из четырех составных частей:

Заголовок. Обязательная часть, которая задает уникальное имя модуля и

версию. Версия позволяет отслеживать изменения в модуле, а так же ядру понимать, что модуль изменился и надо загрузить его новую версию.

Описание типов. Эти строки описывают все возможные типы или атрибуты нашего ограниченного сервиса. Необязательный компонент, если у сервиса нет собственных типов.

Политики. Описание правил работы сервиса. Необязательный если нет особых правил.

Макросы. Используют политики определенные в других модулях (через файл .if). Необязательный.

Пример:Модуль некого приложения с названием myapp. В файле определяется название модуля — myapp и его версия — 1.0.0. Затем идет декларирование типов, как напрямую строками типа type, так и макросами. Последняя часть — политика. Политика описывается напрямую и через макросы.

```
[root@sl0 ~]# cat /usr/share/doc/selinux-policy/example.te
policy_module(myapp,1.0.0)
########################################
#
# Declarations
#
type myapp_t;
type myapp_exec_t;
domain_type(myapp_t)
domain_entry_file(myapp_t, myapp_exec_t)
type myapp_log_t;
logging_log_file(myapp_log_t)
type myapp_tmp_t;
files_tmp_file(myapp_tmp_t)
########################################
#
# Myapp local policy
#
allow myapp_t myapp_log_t:file { read_file_perms append_file_perms
};
allow myapp_t myapp_tmp_t:file manage_file_perms;
files_tmp_filetrans(myapp_t,myapp_tmp_t,file)
```

Описание макросов можно посмотреть в каталоге /usr/share/selinux/devel/include/, если установлен пакет selinux-policy-devel

Файл .fc определяет контексты, которые будут использованы для маркировки файлов, необходимых для работы сервиса.

Пример: Назначение контекста на файл программы.

```
[root@sl0 ~]# cat /usr/share/doc/selinux-policy/example.fc
# myapp executable will have:
# label: system_u:object_r:myapp_exec_t
# MLS sensitivity: s0
# MCS categories: <none>
/usr/sbin/myapp -- gen_context(system_u:object_r:myapp_exec_t,s0)
```

В файле .if описываются макросы, которые другие модули могут использовать для получения доступа к сервису.

Пример: Описывается два макроса: myapp_domtrans, который дает право переходить в домен myapp_t и myapp_read_log, который дает возможность просматривать журналы приложения. Обратите внимание, что в каждом макросе явно описываются используемые типы.

```
[root@sl0 ~]# cat /usr/share/doc/selinux-policy/example.if
## <summary>Myapp example policy</summary>
## <desc>
##  <p>
##    More descriptive text about myapp.  The desc
##    tag can also use p, ul, and ol
##    html tags for formatting.
##  </p>
##  <p>
##    This policy supports the following myapp features:
##    <ul>
##    <li>Feature A</li>
##    <li>Feature B</li>
##    <li>Feature C</li>
##    </ul>
##  </p>
## </desc>
#
########################################
## <summary>
##  Execute a domain transition to run myapp.
## </summary>
## <param name="domain">
##  <summary>
##  Domain allowed to transition.
##  </summary>
## </param>
#
interface(`myapp_domtrans',`
    gen_require(`
      type myapp_t, myapp_exec_t;
```

```
    ')
    domtrans_pattern($1,myapp_exec_t,myapp_t)
')
########################################
## <summary>
##  Read myapp log files.
## </summary>
## <param name="domain">
##  <summary>
##  Domain allowed to read the log files.
##  </summary>
## </param>
#
interface(`myapp_read_log',`
    gen_require(`
      type myapp_log_t;
    ')
    logging_search_logs($1)
    allow $1 myapp_log_t:file read_file_perms;
')
```

После создания модуля. Его необходимо скомпилировать и установить. Процедура установки модуля следующая:

Создаете каталог.

В это каталог копируете файлы .te, .fc и .if

Выполняете команду

make -f /usr/share/selinux/devel/Makefile

Устанавливаете полученный модуль (файл .pp) в ядро командой semodule -i файл.pp

Включаем модуль командой semodule -e модуль

Выполняем перемаркировку файлов командой restoreconn

Пример:Установка нового модуля.

```
[root@sl0 ~]# mkdir myapp
[root@sl0 ~]# cp /usr/share/doc/selinux-policy/example.* myapp/
[root@sl0 ~]# cd myapp/
[root@sl0 myapp]# rename example myapp example.*
[root@sl0 myapp]# ls
myapp.fc  myapp.if  myapp.te
[root@sl0 myapp]# make -f /usr/share/selinux/devel/Makefile
Compiling targeted myapp module
/usr/bin/checkmodule:      loading   policy   configuration   from
tmp/myapp.tmp
/usr/bin/checkmodule:  policy configuration loaded
/usr/bin/checkmodule:   writing binary representation (version 17)
```

```
to tmp/myapp.mod
Creating targeted myapp.pp policy package
rm tmp/myapp.mod.fc tmp/myapp.mod
[root@sl0 myapp]# semodule -i myapp.pp
[root@sl0 myapp]# semodule -e myapp
```

Удаление ненужного модуля производится командой semodule -d модуль

Пример:

```
[root@sl0 myapp]# semodule -l | grep myapp
myapp     1.0.0
[root@sl0 myapp]# semodule -d myapp
[root@sl0 myapp]# semodule -l | grep myapp
```

При использовании SELinux производится аудит событий. В результате вы можете увидеть, как блокируется доступ к тем или иным функциям ОС. События записываются в журнал /var/log/audit/audit.log.

Выбрав из журнала интересующие вас события мы можем сформировать политику, которая устранит блокировки. Для этого можно воспользоваться командой audit2allow, которая анализирует события типа denied и на их основе формирует предлагаемую политику.

Пример: Формирование политики на основе анализа событий по метке httpd_t.

```
[root@sl0 myapp]# grep httpd_t /var/log/audit/audit.log | audit2allow
#============= httpd_t ==============
#!!!! WARNING: 'default_t' is a base type.
#!!!! The file '/myweb/index.html' is mislabeled on your system.
#!!!! Fix with $ restorecon -R -v /myweb/index.html
allow httpd_t default_t:file getattr;
#!!!! The file '/myweb/index.html' is mislabeled on your system.
#!!!! Fix with $ restorecon -R -v /myweb/index.html
#!!!! This avc can be allowed using one of the these booleans:
#                httpd_enable_homedirs,   httpd_read_user_content, httpd_unified
allow httpd_t httpd_user_content_t:file getattr;
```

В общем случае применять такие политики ПЛОХАЯ ИДЕЯ!!!

Лучше проанализировать этот вывод и принять решение что необходимо сделать. Для подробного анализа событий можно воспользоваться командой audit2why, которая показывает почему возникла ошибка и предлагает возможные решения.

Пример: Разбор событий командой audit2why

```
[root@sl0 myapp]# grep httpd_t /var/log/audit/audit.log| grep
user_content | head -1 | audit2why
type=AVC msg=audit(1501603329.870:89): avc: denied { getattr }
for pid=2702 comm="httpd" path="/myweb/index.html" dev="dm-0"
ino=67663060 scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:httpd_user_content_t:s0 tclass=file
    Was caused by:
    One of the following booleans was set incorrectly.
    Description:
    Allow httpd to read home directories
    Allow access by executing:
    # setsebool -P httpd_enable_homedirs 1
    Description:
    Allow httpd to read user content
    Allow access by executing:
    # setsebool -P httpd_read_user_content 1
    Description:
    Unify HTTPD handling of all content files.
    Allow access by executing:
    # setsebool -P httpd_unified 1
```

Анализ подразумевает что вы отвечаете на следующие вопросы:

Мешает ли блокировка правильной работе ограниченного сервиса?

Если сервис работает некорректно, то в чем причина?

Неправильная политика?

Неправильная маркировка файлов?

Некорректные действия пользователей?

По результатам анализа мы можем принять следующие решения:

Принять события и ничего не делать. Это означает, что вы не просто блокируете доступ, но и отслеживаете блокировку. Например, чтобы отслеживать атаки на службу.

Модифицировать существующую политику, если решили, что политика не верна.

Создать новый модуль и установить его. Не создавайте модули автоматически, это может привести к бреши в системе безопасности. В примере выше создавать политику с правилом allow httpd_t default_t:file getattr не лучшее решение. Правильным решением будет установить нужную маркировку файлов.

Запретить аудит блокировки таких событий. Тогда вместо allow необходимо создать модуль с параметром dontaudit или использовать опцию -D

в команде audit2allow

11.3. Реализация других форм контроля доступа с помощью SELinux.

При работе с пользователями использование только меток для процессов и файлов является недостаточным. Причин несколько:

Как правило мы не можем заранее предусмотреть какие программы и файлы будет использовать конкретный пользователь.

Если мы знаем какую-то программу, которую может использовать пользователь, например nano, то не можем с достаточной долей уверенности сказать, как пользователь будет использовать эту программу. Что он будет редактировать?

Запуская одну и ту же программу разным пользователям надо разрешать разный тип доступа. Задачи у каждого пользователя свои.

В целевой политике процесс получает метку от файла программы, но это совершенно не подходит под принцип разделения доступа на основе пользовательской информации.

Для решения задач по управлению доступом пользователей мы можем воспользоваться системой RBAC (Role Based Access Control). В этой системе каждый пользователь Unix связывается с SELinux пользователем. SELinux пользователь получает список своих ролей.

Роль определяет полномочия ее обладателя. Управление ролями настраивается посредством политики.

По умолчанию система RBAC не используется. Все пользователи являются неограниченными, или, другими словами используют систему DAC.

Пример: Пользователю root назначен пользователь unconfined_u (неограниченный) у которого назначена роль unconfined_r. Метка домена пользователя unconfined_t.

```
[root@sl0 myapp]# id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@sl0 myapp]# ps -Z
LABEL                                PID TTY          TIME CMD
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  1055  ttyS0
00:00:07 bash
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  3074  ttyS0
00:00:00 ps
```

Связь Unix и SELinux пользователей можно посмотреть командой semanage login -l. Эта команда так же показывает SELinux пользователя по

умолчанию, это назначение используется для Unix пользователей, которым не привязан SELinux пользователь.

Пример: SELinux пользователь по умолчанию unconfined_u

```
[root@sl0 myapp]# semanage login -l
Login  Name                SELinux  User              MLS/MCS  Range
Service
__default__           unconfined_u         s0-s0:c0.c1023       *
root                  unconfined_u         s0-s0:c0.c1023       *
system_u              system_u             s0-s0:c0.c1023       *
```

Назначенные роли SELinux пользователям можно выяснить командой semanage user -l.

Пример: В примере пользователю staff_u назначены роли staff_r sysadm_r system_r unconfined_r.

```
[root@sl0 myapp]# semanage user -l
                Labeling   MLS/       MLS/
SELinux   User          Prefix            MCS   Level     MCS   Range
SELinux Roles
guest_u                   user              s0                   s0
guest_r
root                   user           s0              s0-s0:c0.c1023
staff_r sysadm_r system_r unconfined_r
staff_u                user           s0              s0-s0:c0.c1023
staff_r sysadm_r system_r unconfined_r
sysadm_u              user           s0              s0-s0:c0.c1023
sysadm_r
system_u              user           s0              s0-s0:c0.c1023
system_r unconfined_r
unconfined_u       user            s0              s0-s0:c0.c1023
system_r unconfined_r
user_u                     user              s0                   s0
user_r
xguest_u                  user              s0                   s0
xguest_r
```

Политика для каждой роли определяется в соответствующем модуле. Команда semanage module -l показывает список модуле.

Пример:

```
[root@sl0 myapp]# semanage module -l | grep staff
staff                   100       pp
```

В SELinux определено несколько стандартных ролей.

Каждая роль имеет индивидуальный модуль.

Вы можете использовать эти роли для выполнения стандартных системных действий.

Если существующие роли не удовлетворяют ваших потребностей, то вы можете создать свою собственную роль. Для этого вам потребуется:

Создать модуль SELinux с описанием политики для вашей роли. При создании модуля вы можете воспользоваться макросами-шаблонами

userdom_unpriv_user_template(myrole): макрос для определения роли похожей на user_r и staff_r.

userdom_admin_user_template(myrole): макрос для определения роли похожей на sysadm_r.

Внести изменения в файлы в каталоге /etc/selinux/targeted/contexts:

default_type — контекст по умолчанию для роли

default_contexts — поведение SELinux программ для выбора контекста пользователю.

Создать Unix пользователя с желаемой привязкой к SELinux пользователю можно командой useeradd с опцией -Z.

Если вы хотите сопоставить сущетсвующих пользователей, то нужно использовать команду semanage login -a.

Пример: Создание Unix пользователя с привязкой к SELinux пользователю.

```
[root@sl0 myapp]# useradd -Z staff_u staffuser
[root@sl0 myapp]# passwd staffuser
Changing password for user staffuser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@sl0 myapp]# gpasswd -a staffuser wheel
Adding user staffuser to group wheel
[root@sl0 myapp]# semanage login -l
Login  Name                SELinux  User             MLS/MCS  Range
Service
__default__           unconfined_u         s0-s0:c0.c1023       *
root                  unconfined_u         s0-s0:c0.c1023       *
staffuser             staff_u              s0-s0:c0.c1023       *
system_u              system_u             s0-s0:c0.c1023       *
```

Пример: Проверка возможностей ограниченного пользователя.

```
[root@sl0 myapp]# ssh staffuser@127.0.0.1
staffuser@127.0.0.1's password:
[staffuser@sl0 ~]$ id -Z
staff_u:staff_r:staff_t:s0-s0:c0.c1023
[staffuser@sl0 ~]$ ps -Z
LABEL                             PID TTY          TIME CMD
staff_u:staff_r:staff_t:s0-s0:c0.c1023 4059 pts/0 00:00:00 bash
staff_u:staff_r:staff_t:s0-s0:c0.c1023 4080 pts/0 00:00:00 ps
[staffuser@sl0 ~]$ su -
Password: Правильный_пароль
su: Authentication failure
[staffuser@sl0 ~]$ sudo su -
[sudo] password for staffuser:
su:  avc.c:74:  avc_context_to_sid_raw:  Assertion  `avc_running'
failed.
[staffuser@sl0 ~]$ sudo -i
-bash: /root/.bash_profile: Permission denied
-bash-4.2# id -Z
staff_u:staff_r:staff_t:s0-s0:c0.c1023
bash-4.2# passwd root
passwd: SELinux denying access due to security policy.
bash-4.2# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=30.5 ms
#
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 1 received, 50% packet loss, time 1001ms
rtt min/avg/max/mdev = 30.590/30.590/30.590/0.000 ms
bash-4.2# wget http://www.google.com -O /dev/null
--2017-08-02 20:53:56--  http://www.google.com/
Resolving  www.google.com  (www.google.com)...  173.194.221.103,
173.194.221.105, 173.194.221.106, ...
Connecting       to       www.google.com       (www.google.com)|
173.194.221.103|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location:                                  http://www.google.ru/?
gfe_rd=cr&ei=lfWBWbe4DsyG7gTYn5u4Aw [following]
--2017-08-02       20:53:56--              http://www.google.ru/?
gfe_rd=cr&ei=lfWBWbe4DsyG7gTYn5u4Aw
Resolving   www.google.ru   (www.google.ru)...   173.194.222.94,
2a00:1450:4010:c0b::5e
Connecting to www.google.ru (www.google.ru)|173.194.222.94|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘/dev/null’
    [ <=>                                  ] 10,960      --.-K/s
```

```
in 0.001s
2017-08-02 20:53:56 (7.81 MB/s) - ‘/dev/null’ saved [10960]
-bash-4.2# logout
```

Переключение на новую роль

```
[staffuser@sl0 ~]$ sudo -r sysadm_r -i
[root@sl0 ~]# id -Z
staff_u:sysadm_r:sysadm_t:s0-s0:c0.c1023
[root@sl0 ~]# passwd root
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 7 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@sl0 ~]# ping 8.8.8.8 -c1
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=30.8 ms
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 30.802/30.802/30.802/0.000 ms
[root@sl0 ~]# wget http://www.google.com -O /dev/null
--2017-08-02 20:53:31--  http://www.google.com/
Resolving   www.google.com   (www.google.com)...   173.194.222.105,
173.194.222.103, 173.194.222.147, ...
Connecting        to        www.google.com        (www.google.com)|
173.194.222.105|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location:                                    http://www.google.ru/?
gfe_rd=cr&ei=fPWBWZPSE6rG7gTF0re4Dg [following]
--2017-08-02        20:53:31--                http://www.google.ru/?
gfe_rd=cr&ei=fPWBWZPSE6rG7gTF0re4Dg
Resolving    www.google.ru    (www.google.ru)...    173.194.221.94,
2a00:1450:4010:c0b::5e
Connecting  to  www.google.ru  (www.google.ru)|173.194.221.94|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘/dev/null’
    [ <=>                                    ] 10,944      --.-K/s
in 0s
2017-08-02 20:53:31 (98.7 MB/s) - ‘/dev/null’ saved [10944]
[root@sl0 ~]# logout
```

Для сопоставления существующего Unix пользователя с SELinux пользователем можно использовать команду semanage login -m:

```
[root@sl0  myapp]#  semanage  login  -m  -s  "user_u"  -r  "s0"
```

```
__default__
[root@sl0 myapp]# semanage login -l
Login Name                SELinux User              MLS/MCS Range
Service
__default__           user_u                s0                    *
root                  unconfined_u          s0-s0:c0.c1023        *
staffuser             staff_u               s0-s0:c0.c1023        *
system_u              system_u              s0-s0:c0.c1023        *
```

Системы MLS или MCS используют, в дополнение к меткам, еще и числовой анализ при предоставлении доступа.

Метка чувствительность (sensitivity) задает уровень доступа. Общая идея при использовании чувствительности: некий уровень может прочитать информацию уровнем ниже своего, но не может ее изменить; на своем уровне пользователь может читать и изменять данные; на уровень выше своего может только изменять (предоставлять) информацию, но не читать.

Использование категорий показывает какой диапазон категорий может получить доступ к другим категориям.

Глава 12. Процесс загрузки и выключения системы

12.1. Последовательность процесса загрузки.

При включении питания компьютера автоматически запускаются программы, находящиеся в ПЗУ.

Исторически, начиная с первых IBM совместимых компьютерах, использовался BIOS (Basic Input/Output Services – Базовые службы ввода-вывода).

Микросхема ПЗУ в таких компьютерах выполняется по технологии CMOS (Complemenary Metal-Oxyde Semiconductor – Комплементарный метало-оксидный полупроводник).

Примечание: Особенностью этой технологии является то, что программа, сохраняемая в таком ПЗУ может быть настроена без необходимости полного перепрограммирования ПЗУ.

Программа, которая позволяет настраивать поведение BIOS называется CMOS Setup.

В 1990-х Intel разработал новый стандарт для компьютеров с архитектурой Itanium. Изначальное название — Intel Boot Initiative (Загрузочная инициатива Intel), позже было переименовано в EFI.

Основной мотивацией разработки EFI было преодоление ограничений BIOS: 16-битный исполняемый код, адресуемая память 1 Мбайт, проблемы с одновременной инициализацией нескольких устройств, ограничения на размер дисков.

В 2005 году Intel внесла эту спецификацию в UEFI Forum, который теперь ответственен за развитие и продвижение EFI. EFI был переименован в Unified EFI (UEFI).

Одна из программ, входящих в BIOS/UEFI называется POST (Power On Self Test – Проверка самого компьютера при включении питания). Эта программа выполняет начальную проверку конфигурации компьютера и при наличии проблем выдает специальные сигналы (обычно звуковые).

После удачного завершения программы POST запускается другая программа в BIOS, которая называется Bootstrap Loader (аппаратный загрузчик). Она предназначена для обращения к загрузочному устройству, указанному с помощью CMOS Setup в BIOS, поиска на нем загрузочного сектора и, при удачном поиске, загрузке его.

Если загрузка осуществляется с магнитного диска, то программа загрузки находится в первом по порядку секторе. Для жестких магнитных дисков загрузочный раздел еще называют MBR (Master Boot Record – Главная загрузочная запись).

Аппаратный загрузчик опознает программу загрузки в загрузочном разделе по сигнатуре ее последних двух байт (510 и 511 байты начиная с 0 байта – в шестнадцатеричном виде, соответственно, байты `0x1FE` и `0x1FF`). Сигнатура 16 бит, составленная этими байтами, является числом `0xAA55`. Именно по этой сигнатуре опознается загрузочный сектор, который и загружается аппаратным загрузчиком в ОЗУ.

Если программный загрузчик в MBR отсутствует, то его поиск осуществляется в первом секторе раздела, помеченного в таблице разделов жесткого диска, как активный.

Размер секторов на магнитном диске – 512 байт, поэтому программа, находящаяся в загрузочном секторе, проста и не может выполнять сложных действий.

Примечание: В некоторых загрузчиках (не в GNU/Linux!) программный загрузчик в MBR выдает на экран список разделов, с которых возможно продолжение загрузки.

Для преодоления этого ограничения загрузчики операционных систем разбивают на несколько частей.

Загрузчики, чаще всего применяемые в GNU/Linux – LILO и GRUB, состоят из двух частей.

Одна из них, называемая начальным загрузчиком, помещается либо в MBR, либо в первый сектор активного раздела. Задачей начального загрузчика является поиск и загрузка второй части загрузчика.

Вторая часть загрузчика представляют собой более сложную программу, которая может загрузить ядро Linux (или другое ядро) и выполнить необходимые предварительные действия для загрузки операционной системы (например, создание электронного диска с примитивным образом корневой файловой системы).

Загрузка с UEFI реализована несколько иначе.

С GPT-раздела с идентификатором EF00 (ESP, EFI system partition) и файловой системой FAT32 по умолчанию загружается и запускается файл \efi\boot\boot[название архитектуры].efi, например: \efi\boot\bootx64.efi.

Следующий этап либо запуск загрузчика, либо прямая загрузка ядра Linux.

Также в большинстве реализаций UEFI возможна загрузка в режиме совместимости с диска с разметкой MBR.

При использовании UEFI возможна также Secure Boot, смысл которой заключается в проверке цифровых подписей ядра и модулей ядра.

Secure Boot одна из причин почему может не загрузиться Linux, например когда вы скомпилируете новые модули ядра самостоятельно, но не подпишите их.

Загрузчики, используемые в GNU/Linux, способны передавать ядру Linux различные конфигурационные параметры и дополнительные команды.

Примечание: В частности, ядру Linux может быть передана команда считать из файловой системы на электронном диске специальный командный файл, который чаще всего называется `linuxrc`. *В этом командном файле можно указать список дополнительных действий, которые необходимо выполнить при инициализации системы GNU/Linux.*

После загрузки ядра и его инициализации, если в параметрах, переданных ядру, не указано иное, производится запуск программы `/sbin/init` (или в новых ОС GNU/Linux systemd).

Ее конфигурационный файл `/etc/inittab` .

Процесс, соответствующий этой программе всегда имеет PID равный 1.

Эта программа осуществляет дальнейшую инициализацию операционной системы GNU/Linux, зависящую от выбранного режима работы (например, однопользовательский или многопользовательский режим).

В Linux и UNIX подобных системах такие режимы работы называются уровнями исполнения (runlevels).

В зависимости от выбранного уровня исполнения команда `/sbin/init` исполняет тот или иной набор специальных командных файлов, называемых инициализационными скриптами.

Именно инициализационные скрипты определяют какие программы будут работать в фоновом режиме на том или ином уровне исполнения, то есть какова будет функциональность системы.

Пример: однопользовательский режим применяется для обслуживания системы и при работе в нем нет необходимости в запуске сетевых приложений. Подробнее вопросы инициализации системы на различных уровнях исполнения будут рассмотрены позже в этой главе.

Одна из программ, которые запускает процесс `/sbin/init` — это программа `getty` или ее разновидность.

Эта программа выводит приглашение войти в сеанс (`login:`). Если пользователь вводит что-либо в ответ на это приглашение, то эта строка интерпретируется как имя пользователя, которое передается программе `login` как аргумент.

Далее выводится приглашение ввести пароль (`password:`), который проверяется в базе данных паролей (например, в `/etc/shadow` – в зависимости от настроек системы).

Если аутентификация производится успешно, то запускается оболочка, указанная для пользователя в файле `/etc/passwd`, осуществляя, таким образом, вход в сеанс пользователя.

Перед выводом приглашения `login:` на экран печатается содержимое файла `/etc/issue` (или `/etc/issue.net`, если подключение осуществляется через сеть). А после входа в сеанс на экран выводится содержимое файла `/etc/motd` (Message Of The Day – Сообщение дня).

12.2. Загрузчик GRUB2.

Загрузчик GRUB (GRand Unified Bootloader) является современной альтернативой LILO.

В отличие от загрузчика LILO, который использует данные о точном расположении нужного образа ядра на магнитном носителе, загрузчик GRUB обращается к файловой системе на этом носителе для поиска файла ядра.

Примечание: Загрузчик LILO обращается к заданному номеру сектора диска, считывая определенное количество байт, а загрузчик GRUB обращается к файлу в файловой системе. Загрузчик GRUB распознает различные типы файловых систем, например, ext2, ext3, ReiserFS, XFS, JFS. GRUB способен работать с большими дисками, имеющими более 1024 цилиндров. При этом обеспечивается загрузка ядер ОС с разделов, расположенных в любом месте диска.

GRUB способен загружать ядра разнообразных операционных систем, например, GNU/Linux, GNU/HURD, FreeBSD и других.

Для операционных систем, загрузка ядер которых не поддерживается, загрузчик GRUB обеспечивает цепную загрузку (chainload) , то есть передачу управления загрузчику другой операционной системы.

GRUB поддерживает автоматическую декомпрессию загружаемых файлов, сжатых в формате zip.

GRUB обладает встроенной командной оболочкой, позволяющей гибко управлять процессом загрузки, а также автоматизировать его, задавая в

конфигурационном файле загрузчика последовательности команд, которые должны быть выполнены в процессе загрузки.

Загрузчик GRUB поддерживает загрузку через локальную сеть посредством протокола TFTP (Trivial File Transfer Protocol) и поддерживает управление загрузкой с удаленного последовательного терминала.

В современных GNU/Linux как правило применяется загрузчик GRUB2.

GRUB2 — дальнейшее развитие проекта GRUB.

Основное новшество — возможность описать загрузку ОС без привязки к физическому расположению устройств.

В GRUB2 изменился формат файла конфигурации.

Конфигурационный файл — /boot/grub2/grub.cfg

Настоятельно не рекомендуется изменять файл конфигурации вручную. Вместо этого имеется специальная команда grub2-mkconfig (иногда grub-mkconfig)

grub2-mkconfig — создает конфигурацию на основе заготовок в каталоге /etc/grub.d/, содержимого каталога /boot и /etc/default/grub

Если необходимо поменять параметры загрузки ядер Linux, то изменения необходимо вносить в файл /etc/default/grub

Дополнительные пункты меню загрузки определяются в файле /etc/grub.d/40_custom

12.3. Остановка и перезагрузка системы.

Для немедленной остановки или перезагрузки системы можно использовать команды, соответственно, `/sbin/init 0` или `/sbin/init 6`. Однако для этого удобнее использовать команды `halt` для остановки или `reboot` для перезагрузки.

Команда `halt`

вносит в файл `/var/log/wtmp` запись о том, что система была остановлена в это время.

Далее для останова вызывается команда `shutdown -h now`, останавливающая систему.

Опция `-f` (force) команды `halt` , заставляет систему остановиться без вызова `shutdown`.

Если команда `halt` вызвана с опцией `-n`, то перед остановом не будет произведена операция сброса содержимого кэша на диск.

При использовании команды `halt -d` запись в файл `/var/log/wtmp`

произведена не будет.

Остановка системы с последующим отключением питания будет произведена в результате выполнения команд `halt -p` или `poweroff`.

Примечание: Обычно команды `poweroff` *и* `reboot` *реализованы в виде символических ссылок на файл команды* `/sbin/halt`.

Основной командой для безопасной остановки или перезагрузки системы является `/sbin/shutdown`.

С помощью нее можно осуществлять как немедленную, так и отложенную остановку системы.

Эта команда посылает пользователям предупреждение о том, что система останавливается. Процессам, работающим в этот момент, посылается сигнал `SIGTERM`, получив который приложения могут корректно завершить свою работу.

Для останова системы команда `shutdown` посылает сигнал процессу `init` для перехода на 0-й или 6-й уровень исполнения при вызове с опцией, соответственно, `-h` или `-r`.

Пример:

```
/sbin/shutdown -h now
```

Примечание: Данная команда осуществит немедленную остановку системы, так как в качестве времени останова системы указан параметр `now`. *Если же необходимо остановить или перезагрузить систему в заданное время, то его следует указать в качестве аргумента:*

```
/sbin/shutdown -r 17:00 'System will be rebooted at 17:00!'
```

В этом примере перезагрузка (`-r`*) системы будет произведена в 17:00, причем пользователи будут оповещены об этом с помощью строки сообщения, указанной в качестве аргумента.*

Вместо использования точного указания времени можно указывать время задержки перед остановом. Если задержка измеряется секундами, то количество секунд следует указать после опции `-t`.

Количество минут задается опцией `+n` где n количество минут

Пример:

```
/sbin/shutdown -h +10
```

Примечание: В данном случае останов будет предпринят через 10 минут. Реально между этими двумя вариантами задания задержки существует разница: после опции `-t` *задается время задержки в секундах до того, как* `shutdown` *передаст сигнал* `init` *для перехода на другой уровень исполнения. Если же используется указание либо времени, либо задержки в минутах, то при этом реальное действие самой команды* `shutdown` *будет произведено с заданной задержкой. При этом пользователи, вошедшие в сеанс могут продолжать работать до начала останова, но новые сеансы не будут открыты.*

В таблице, приведенной ниже, указаны часто используемые опции команды `shutdown`.

Опция	Назначение
`-c`	Отменить начавшийся останов системы.
`-f`	создает файл `/fastboot`, наличие которого позволяет не проверять файловую систему при загрузке.
`-F`	создает файл `/forcefsck`, наличие которого вынуждает проверять файловую систему при загрузке.
`-h`	Остановка системы.
`-r`	Перезагрузить систему.
`-k`	Послать пользователям сообщение, но не останавливать систему.

В системах с systemd выключением и перезагрузкой системы управляет сам systemd, командой systemctl:

systemctl reboot — перезагрузка;

systemctl halt — остановка;

systemctl poweroff — выключение питания.

Глава 13. Лабораторные работы

13.1. Лабораторные работы к главе 1.

1. Напишите сценарий `scr1.sh` так, чтобы в нем определялась переменная `V1` и выводилось ее значение.
2. Запустите сценарий, указывая оболочку явно.
3. Измените скрипт для неявного вызова оболочки.
4. Перепишите сценарий `scr1.sh` таким образом, чтобы из него вызывался сценарий `scr2.sh`, который и печатал бы значение переменной `V1`.
5. Перепишите сценарий `scr1.sh` таким образом, чтобы значение переменной `V1` считывалось бы из файла `scr1rc`.
6. Требуется проверить относится ли файл, заданный в качестве аргумента, каталогом или же обычным файлом. Если верно последнее, то сценарий должен выводить имя файла и его размер. В случае, если размер файла превышает 1Кб, то размер должен выводиться в килобайтах. Если размер превышает мегабайт - в мегабайтах.
7. Изучите любой сценарий запуска в `/etc/init.d` . Как в этом сценарии используется команда `case` ?
8. Напишите сценарий, анализирующий список пользователей, находящихся в настоящий момент в системе. В скрипте список пользователей должен быть преобразован в одну строку. В случае, если имеется хотя-бы один сеанс `root` должно выдаваться предупреждающее сообщение. Анализ должен быть осуществлен с помощью команды `case` .
9. Напишите сценарий, позволяющий в цикле создавать в текущем каталоге символические ссылки на файлы, заданные как его аргументы. Причем для каждого файла должно запрашиваться подтверждение на создание ссылки. В этом сценарии должна использоваться команда `for`.
10. Измените предыдущий сценарий так, чтобы для организации цикла использовались бы `while` или `until`.

13.2. Лабораторные работы к главе 2.

1. Определите состояние службы sshd.
2. Определите в какую цель по умолчанию загружается система. Как изменить эту цель?
3. Установите цель загрузки по умолчанию в multi-user.target. Перезагрузите

систему.

4. Прервите загрузку в меню выбора ОС. И, через встроенную оболочку GRUB, загрузите систему в цель basic.target. Можете ли вы взаимодействовать с компьютером?
5. Какая команда показывает только текущее состояние сервиса?
6. Проверьте нормально ли работает система в целом. Если состояние системы не соответствует нормальному (running), то как определить в чем причина?
7. Остановите и запретите запуск службы firewalld.

13.3. Лабораторные работы к главе 3.

Предполагается что вы выполняете задание на виртуальной машине, в которую вы предварительно добавили новый диск размером 20 ГБ.

1. Проверьте текущее разбиение дисков на разделы. Ваш новый диск должен иметь имя /dev/sdb. В дальнейших пунктах будут использоваться именно это имя, если оно отличаются от вашего - используйте полученные вами значения.
2. Выполните команду `sfdisk -s` . Узнайте, что она выдает.
3. На каком разделе вашего диска имеется область подкачки `swap`?
4. В интерактивном режиме команды `fdisk` получите информацию о таблице разделов на одном из ваших новых дисков.
5. Создайте 2 раздела размером по 5 ГБ и один раздел размером 10 ГБ на диске /dev/sdb с помощью команды `fdisk`.
6. Определите код типа раздела для физического тома системы Linux LVM (Logical Volume Manager).
7. Какие могут быть созданы типы разделов FAT, каковы их идентификаторы?

Внимание! При ошибочных действиях все данные на диске могут быть утрачены! При создании раздела на диске, где размещена корневая файловая система, возможно потребуется перезагрузка после создания раздела.

8. Создайте на созданном разделе /dev/sdb1 файловую систему `XFS`.
9. Создайте на разделе /dev/sdb2 файловую систему `EXT2`. Отличается ли информация, выдаваемая командой создания файловой системы, от той, которую выдавала предыдущая команда?
10. Создайте на разделе /dev/sdb3 файловую систему `EXT4` с предварительной

проверкой поверхности диска на плохие блоки. Файловая система должна иметь размер блока `2048` байт.

11. Определите, где находится первая копия суперблока в только что созданной файловой системе.
12. Проверьте целостность только что файловой системы.
13. Выполните команду проверки этой же файловой системы так, чтобы была выполнена ее полная проверка.
14. Командой blkid определите UUID-ы разделов /dev/sdb1 и /dev/sdb2.
15. Создайте запись в /etc/fstab для автоматического монтирования раздела /dev/sdb1 по UUID в каталог /storage1.
16. Автоматическое монтирование раздела /dev/sdb2 по UUID в каталог /storage2 опишите в виде юнита systemd.
17. Изучите для чего применяются опции `user`, `users` и `owner` команды `mount`. В чем их отличие?
18. Получите информацию о структуре разделов на ваших дисках.
19. Какие файловые системы на созданы на разделах дисков.
20. Выведите информацию об использовании дискового пространства на всех смонтированных блочных устройствах в системе.
21. Получите информацию об использовании дискового пространства домашним каталогом.

13.4. Лабораторные работы к главе 4.

1. Добавьте ДВА новых физических диска в вашу виртуальную машину, размер каждого диска — 8 Гб
2. Проверьте наличие в системе физических дисков — lsblk.
3. Создайте LVM диски sdc и sdd, инициализируя устройства как PV командой pvcreate.
4. Проверьте корректность создания LVM дисков командами pvscan и pvs.
5. Создайте группу с именем gr1 из двух физических томов sdc и sdd.
6. Проверьте создание группы командами pvdisplay и vgdisplay
7. Создайте два логических тома lv1 и lv2, принадлежащих группе gr1: размер первого должен значительно превышать размер физического диска (ориентируйтесь на 12 Гб) и размер второго 3 ГБ
8. Проверьте создание томов командами lvscan и lvdisplay.
9. Создайте файловые системы логических томов, ext4 для lv1 и ext2 для lv2
10. Создайте каталоги /disk1 и /disk2 и смонтируйте в них ранее созданные

логические тома.

11. Создайте файл bigfile и забейте его нулями до размера, примерно 5Гб: sudo dd if=/dev/zero of=/disk1/bigfile bs=1G count=5
12. Проверьте работу подключенных дисков командой df -h
13. Настройте автоматическое монтирование дисков при старте системы одним из способов
14. Добавьте новый диск, размером 8 Гб в виртуальную машину.
15. Проверьте наличие в системе физических дисков — lsblk.
16. Произведите замену диска sdc на диск sde:
    - инициализируйте устройство sde как PV;
    - добавьте диск sde в группу gr1 командой vgextend;
    - перенесите данные с одного из физических томов на новый том и извлеките данный физический том из группы томов
    - перед извлечением физического диска из системы обязательно проверьте, а при необходимости измените параметры автомонтирования (после извлечения физического диска и перезагрузки, система переименует устройства, при этом UUID останутся неизменными
    - извлеките физический диск из виртуальной машины, проверьте корректность работы системы.
17. Увеличьте размер логического тома lv2 так, чтобы ему выделить все свободное пространство группы томов gr1: sudo lvextend -l +100%FREE /dev/gr1/lv2
18. Добавьте новый физический диск в вашу виртуальную машину, размер диска — 8 Гб
19. Отформатируйте диск в формате LUKS, задав пароль.
20. Откройте LUKS – том для редактирования.
21. Создайте файловую систему ext4 на данном разделе.
22. Смонтируйте раздел, предварительно создав точку монтирования
23. Проверьте создание шифрованного диска
24. Проверьте самостоятельно, что теперь можно работать с шифрованным диском lock1, как с обычным диском.
25. Закройте раздел LUKS lock1:
    - размонтируйте файловую систему
    - полностью закройте lock1 командой cryptsetup luksClose;
    - проверьте состояние lock1
26. Создайте парольную фразу в виде бинарного файла размером 256 байт:

sudo dd if=/dev/random of=/my.key bs=1 count=256 Примечание: вместо этого можно в качестве парольной фразы использовать любой файл, например картинку в формате .png

27. Создайте ключ для новой парольной фразы в одном из слотов командой cryptsetup luksAddKey
28. Проверьте доступность раздела, используя команду luksOpen, и смонтируйте раздел:
    • по парольной фразе в виде файла;
    • по парольной фразе вводимой с клавиатуры;
29. Отредактируйте файлы /etc/crypttab и /etc/fstab для автоматического монтирования зашифрованного диска во время загрузки. Монтирование осуществляется по парольной фразе вводимой с клавиатуры.
30. Перезагрузите ОС и проверьте, что зашифрованный раздел монтируется автоматически.

13.5. Лабораторные работы к главе 5.

1. Создайте пустой файл и сожмите его `gzip` . Уменьшился ли размер этого файла? Почему?
2. Получите в домашнем каталоге сжатые `gzip`, xz и `bzip2` копии файла `/bin/mount` . Исходный файл не должен быть изменен. Сравните результаты сжатия.
3. Проверьте целостность сжатых файлов.
4. Как с помощью `bzip2` сделать сжатую копию некоторого файла с расширением, характерным для `gzip` ?

13.6. Лабораторные работы к главе 6.

1. Создайте файл, содержащий команду, позволяющую послать суперпользователю электронное письмо, содержащее сведения о средней загрузке системы.
2. Выполните команду `at` так, чтобы предыдущий файл был исполнен через 3 дня.
3. Получите список заданий `at` .
4. Каждое утро в 10:30 требуется получать в виде электронного письма список из десяти пользователей, входивших в сеанс в последнее время. Создайте соответствующую таблицу `cron` для обычного пользователя.
5. Будучи суперпользователем выведите таблицу, которая создана в

предыдущем пункте.

6. От имени суперпользователя отредактируйте эту таблицу так, чтобы создавалось еще одно электронное письмо с информацией о пользователях, находящихся в сеансе в 9:30 утра каждого рабочего дня.
7. Проверьте установлен ли у вас пакет anacron, если не установлен, то установите.
8. Изучите как anacron запускает ежедневные, еженедельные и ежемесячные задания в вашей системе.

13.7. Лабораторные работы к главе 7.

1. Получите список модулей, загруженных в ядро в вашей системе.
2. Узнайте, кто является автором модуля `e1000e`.
3. Определите зависимости модуля `e1000e` от других модулей.
4. Для какого оборудования предназначен модуль `igb`?
5. Получите список PCI устройств.
6. Какой модуль ядра используется для работы с видеоадаптером.
7. Получите список подключенных USB устройств.
8. Создайте правило udev, которое создает символьную ссылку для подключаемого USB Flash диска с именем flashka.

13.8. Лабораторные работы к главе 8.

1. Создайте в конфигурации демона r`syslogd` строку, которая заставит его записывать сообщения от источника `auth` с уровнем важности не ниже `info` в файл `/var/log/mylog` . Не забудьте перезапустить службу, чтобы изменения вступили в силу.
2. Проверьте, записываются ли в этот файл сообщения при входе в сеанс и выходе из него пользователей.
3. Создайте аналогичный журнал `/var/log/mylogpriv` для записи сообщений от источника `authpriv` . Проверьте его работоспособность. Записываются ли в него те же сообщения, что и в предыдущий журнал?
4. Посмотрите с помощью journalctl журнал загрузки системы.
5. Настройте systemd вести постоянные журналы.
6. Создайте настройки ротации для журналов `/var/log/mylog` и `/var/log/mylogpriv`. Файлы должны ротироваться ежедневно. Ротация первой копии должна осуществляться два раза. Ротируемые копии должны сжиматься утилитой `gzip`.

7. Для файла `/var/log/mylog` установите в качестве дополнительного условия ротации достижение им размера 10 Кб.

13.9. Лабораторные работы к главе 9.

1. Установите пакет iptables-services.
2. Сохраните имеющиеся правила фильтрации и NAT в конфигурационном файле сервиса iptables.
3. Очистите все цепочки и запустите сервис iptables.
4. Остановите сервис iptables. И запретите его запуск.
5. Если сервис firewalld еще не установлен, то установите и запустите его.
6. Посмотрите какие имеются зоны. Какие интерфейсы в этих зонах и какие службы могут сейчас работать.
7. Переопределите интерфейс eth0 в зону work. Какие теперь службы разрешены?
8. Дайте разрешение для всех служб и портов, которые вам необходимы.
9. Настройте NAT маскарадинг.

13.10. Лабораторные работы к главе 10.

1. Импортируйте новую виртуальную машину с РЕД ОС
2. Выполните удаленное подключение с 1 ВМ на вторую, используя команду ssh.
3. Создайте файл с содержимым каталога /etc и передайте его с первой ВМ на вторую, используя команду scp.
4. Выполните аутентификацию по ключам, подключаясь с первой ВМ на вторую, используя утилиту ssh.
5. Задайте нестандартный порт для SSH, и запретите использование пользователя root на одной из ВМ. Выполните подключение к данной ВМ.
6. Выполните одиночную команду date при помощи Ansible с сервера (первая ВМ) для всех клиентах (вторая ВМ).
7. Выполните подключение к удаленному рабочему столу Windows, используя утилиту Remmina
8. Выполните подключение к удаленному рабочему столу RedOS, используя утилиту X2go

13.11. Лабораторные работы к главе 11.

1. Проверьте состояние SELinux. Если оно выключено, то включите его или переведите в режим enforcing.
2. Установите веб-сервер apache и проанализируйте как промаркированы файлы этой службы.
3. Создайте файл /var/www/html/index.html и проверьте какой контекст установлен на этот файл. Убедитесь, что веб-сервер показывает стартовую страницу.
4. Создайте алиас для каталога /mywebpage, поместите в этот каталог файл index.html и проверьте доступна ли эта страница. На данном этапе страница не должна быть доступна.
5. Сделайте временную перемаркировку файлов для веб-сервера и проверьте доступ к странице алиаса.
6. Восстановите маркировку по умолчанию и создайте постоянное правило маркировки. Примените эти правила к каталогу /mywebpage. Убедитесь, что правила маркировки отработали правильно и доступ к алиасу восстановился.
7. Разрешите сетевое подключение веб-сервера к базам данных.
8. Создайте собственный модуль для SELinux.
9. Скомпилируйте и установите модуль.
10. Включите модуль.
11. Удалите свой модуль.
12. Проанализируйте журнал аудита и примите решение по устранению отказов в доступе. Стоит ли их принять, проигнорировать или исправить политики или настройки политик. Если необходимо внесите нужные изменения.
13. Создайте нового Unix пользователя и сопоставьте его с SELinux пользователем staff_u. Проверьте какие операции может выполнять данный пользователь.
14. Создайте пользователя webadm_u и проверьте его привилегии.
15. Измените SELinux пользователя по умолчанию на user_u. Проверьте действует ли изменение на обычного и суперпользователя.

13.12. Лабораторные работы к главе 12.

1. Отредактируйте содержимое файла `/etc/issue` и проверьте вывод на экран его содержимого, выйдя из сеанса.

2. Отредактируйте файл `/etc/motd` и проверьте, выводится ли на экран его содержимое при входе в сеанс.
3. Изучите содержимое файла конфигурации GRUB2. Какие пункты меню загрузки там определены? Как описывается раздел, с которого производится загрузка ядра? Как устанавливается раздел для корневого каталога?
4. Выполните команду grub2-mkconfig и сравните вывод с файлом /boot/grub2/grub.cfg.
5. Как описывается загрузка консольного порта в файле /etc/default/grub?
6. Через встроенную оболочку GRUB задайте параметр загрузки ядра init=/bin/bash. Запустите систему. Что произошло? Получите список всех процессов. Загрузите систему в обычном режиме.