Глава 12

RHCE Административные задачи

ЦЕЛИ СЕРТИФИКАЦИИ

12.01 Автоматизация обслуживания системы
12.02 Настройка отчетов об использовании системы
12.03 Параметры времени работы ядра
12.04 IP-маршруты
12.05. Введение в IPv6
12.06 Связывание и объединение сетевых интерфейсов
12.07 Аутентификация с помощью Kerberos
✓ Двухминутная тренировка
Q & A Самопроверка

Автоматизация технического обслуживания системы является целью экзаменов **RHCSA** и **RHCE**. Для **RHCE** вам нужно знать, как создать сценарий оболочки. Вы изучите несколько примеров сценариев, используемых в **RHEL 7** для автоматизации обслуживания системы. Вы можете автоматизировать эти сценарии по расписанию: ежечасно, ежедневно или даже еженедельно.

Отчеты об использовании системы **Linux** связаны с командой **sar**, которая настроена как задание **cron**. Как только вы определили наиболее используемые системные ресурсы, вы можете настроить систему. Этот процесс начинается с **ядра Linux**, которое легко настраивается. С помощью различных параметров времени выполнения, настроенных в каталоге **/proc/sys**, ядра могут быть настроены в соответствии с потребностями ваших приложений.

Задачи **RHCE** также включают ряд дополнительных сетевых требований. Вам нужно знать, как настроить **статические маршруты и настроить IPv6**. Вы также узнаете, как настроить объединение сетевых интерфейсов для агрегирования полосы пропускания и избыточность каналов из нескольких сетевых интерфейсов. Наконец, вы должны знать, как настроить систему в качестве клиента **Kerberos**.

ВНУТРИ ЭКЗАМЕНА

В этой главе непосредственно рассматриваются семь целей RHCE. Первый - это важный навык для системного администрирования; в частности, чтобы

- Использование сценариев оболочки для автоматизации задач обслуживания системы

Сценарии **shell** объединяют серию команд в одном исполняемом файле. Автоматизированные сценарии обычно запускаются по регулярному расписанию, с которым демон **cron** прекрасно справляется.

Отчет об использовании системы является важным навыком для всех компьютерных специалистов. **RHEL 7** включает пакет **sysstat** для таких отчетов. Связанные задачи:

- Создание и доставка отчетов об использовании системы (**процессор, память, диск и сеть**).

Поскольку это не традиционный сетевой сервис, нет необходимости настраивать брандмауэр. В настоящее время нет логических значений, связанных с **SELinux**.

Некоторые задачи настройки **Linux** могут быть выполнены с помощью параметров времени выполнения ядра. Это стало возможным благодаря виртуальным файлам в каталоге **/proc/sys** командой **sysctl**. Соответствующая цель

- Используйте **/proc/sys** и **sysctl** для изменения и установки параметров времени выполнения ядра

В этой главе также рассматриваются несколько сетевых задач из задач **RHCE**. Настройка статических маршрутов, как описано в следующей цели, является важной задачей в корпоративных сетях:

- Маршрутизация IP-трафика и создание статических маршрутов.

Теперь, когда у нас заканчиваются адреса **IPv4, RHCE для RHEL 7** включает в себя соответствующую задачу:

- Настройте адреса **IPv6** и выполните базовое устранение неполадок **IPv6**

В корпоративной сети обычно объединяют несколько сетевых интерфейсов для повышения отказоустойчивости или повышения пропускной способности. В **RHEL 7** вы можете объединить несколько сетевые интерфейсы либо через связывание интерфейсов, либо через объединение в сеть, что решается следующей задачей:

Используйте объединение в сеть или соединение для настройки агрегированных сетевых соединений между двумя системами **Red Hat Enterprise Linux**

Для достижения конечной цели в этой главе вы научитесь настраивать систему **RHEL 7** в качестве клиента **Kerberos. Kerberos** предоставляет безопасные услуги аутентификации в незащищенных сетях. Для сдачи экзамена **RHCE** вы должны быть в состоянии

- Сконфигурируйте систему для аутентификации с использованием **Kerberos**

Чтобы подготовиться к этим требованиям, вы научитесь настраивать центр распространения ключей **Kerberos (KDC).** Более подробную информацию о **Kerberos** см. В Руководстве по аутентификации на уровне системы **Red Hat**, доступном по адресу **https://access.redhat.com/Documentation/en-US/Red_Hat_Enterprise_Linux/7**.

ЦЕЛЬ СЕРТИФИКАЦИИ 12.01

Автоматизировать обслуживание системы

Как обсуждалось в **главе 9, RHEL 7** включает в себя стандартные сценарии обслуживания системы, запланированные с помощью файлов конфигурации **/etc/crontab** и **/etc/anacrontab**, а также различные файлы в каталогах **/etc/cron.*.** В этой главе вы проанализируете некоторые сценарии и некоторые связанные внутренние команды **bash**. Тогда у вас появятся навыки, необходимые для создания основных административных сценариев.

Стандартные административные сценарии

Просмотрите сценарии в каталоге **/etc/cron.daily**, начиная с **rhsmd**, части **Red Hat Subscription Manager**. Он регистрирует информацию о текущем статусе прав системы. Этот скрипт имеет две строки. Обычно строки, начинающиеся с символа **хеша (#),** являются

комментариями. Первая строка начинается с **«shebang» (#!)**, За которым следует **/bin/sh**, которая является стандартной первой строкой для скриптов **bash**. На **RHEL 7**, поскольку **/bin/sh** символически связан с **/bin/bash**, он говорит **RHEL 7** интерпретировать команды, следующие за оболочкой **bash**:

#!/bin/sh

!!!!!
Некоторые дистрибутивы Linux (не Red Hat) связывают команду /bin/sh с оболочкой, отличной от bash. Если в сценарии не указан #!/bin/bash**, он не может быть передан в другие дистрибутивы.**
!!!!

Во второй строке запускается команда **rhsmd**, которая записывает все результаты в **syslog**:

/usr/libexec/rhsmd -s

Далее, проверьте содержимое каталога **/etc/cron.daily**. Несколько более сложный скрипт **logrotate**. Копия скрипта показана на **рисунке 12-1**.

РИСУНОК 12-1 Скрипт ротации файлов логов

```
[root@server1 ~]# cat /etc/cron.daily/logrotate
#!/bin/sh

/usr/sbin/logrotate /etc/logrotate.conf
EXITVALUE=$?
if [ $EXITVALUE != 0 ]; then
    /usr/bin/logger -t logrotate "ALERT exited abnormally with [$EXITVALUE]"
fi
exit 0
[root@server1 ~]#
```

Сценарий начинается с **shebang** и пути интерпретатора программы, который будет анализировать остальную часть сценария:

#!/bin/sh

Следующая строка в файле выполняется автоматически. Команда **logrotate** выполняет ротацию журналов, как определено в файле **/etc/logrotate.conf**, описанном в **главе 9**:

/usr/sbin/logrotate /etc/logrotate.conf

Следующая строка назначает значение выхода, возвращаемое последней командой, переменной с именем **EXITVALUE**:

EXITVALUE = $?

Если команда **logrotate** успешна, **EXITVALUE** устанавливается **в 0**.
Следующая команда **if** запускает условный оператор. Последовательность символов **«!=»** Означает «**не равно**». Следовательно, следующее, если условное истинно, когда значение **EXITVALUE** отличается **от 0**:

if [$EXITVALUE != 0]; then

Если **EXITVALUE не равно 0, bash** выполняет команды внутри условного **оператора if,** который сообщает администратору о проблеме со сценарием **logrotate** или связанными файлами журнала.

/usr/bin/logger -t logrotate "ALERT exited abnormally with [$EXITVALUE] "

Следующая **команда fi** завершает условный оператор. Последняя директива **возвращает 0**, что указывает на успех:

exit 0

С этим введением в скрипты вы готовы изучить некоторые переменные и команды **bash**.

Переменные Bash

Вы можете использовать **переменные в Bash** для хранения данных. Хотя обычно имена переменных пишутся заглавными буквами, имя переменной нельзя начинать с цифры.

В следующем примере показано, как можно назначить переменную из командной строки:

 # today=4

Будьте осторожны при назначении переменной, чтобы не добавлять пробелы вокруг одинакового (=) символа. Чтобы отобразить значение переменной, используйте команду **echo** и добавьте знак доллара перед переменной:

 # echo $today
4

Вы также можете добавить фигурные скобки вокруг имени переменной, чтобы избежать неоднозначных выражений. Например, без фигурных скобок следующая команда получит значение переменной **$todayth**, а не **$today**:

 # echo "Today is the ${today}th of June"
Today is the 4th of June

Вы можете использовать переменные как часть арифметических выражений. В **bash** арифметические выражения заключены в синтаксис **$((expression))**. Вот пример:

 # tomorrow=$(($today + 1))
 # echo "Tomorrow is the ${tomorrow}th of June"
Tomorrow is the 5th of June

Но это еще не все. Переменные также могут хранить выходные данные команды. Это можно сделать двумя способами: с помощью синтаксиса **$(command)** и с помощью **backticks `command`**. Вот пример каждого:

 # day=$(date +%d)
 # month=`date +%b`
 # echo "The current date is $month, $day"
The current date is Jun, 29

Команды Bash

Скрипты заполнены различными командными конструкциями. Некоторые группы команд выполняются только при соблюдении условия. Другие организованы в цикл, который продолжает работать, пока выполняется условие. Эти командные конструкции также известны как условные и управляющие структуры. Общие команды включают **for, if и test**. Конец цикла может быть помечен ключевым словом, таким как **done** или **fi**. Некоторые команды существуют только в контексте других, что будет описано в следующих подразделах.

Тестовые операторы с if

Оператор **if** в основном используется для проверки выполнения условия, например, существует ли файл. Например, следующая команда проверяет, существует ли файл **/etc/sysconfig/network** и является ли он обычным файлом:

if [! -f /etc/sysconfig/network]; then

Восклицательный знак **(!)** Является **оператором «не»** и отрицает результат теста. **-f** проверяет, является ли следующее имя файла текущим обычным файлом. Операторы тестирования очень распространены в сценариях оболочки **bash**. Некоторые из этих операторов перечислены в **таблице 12-1**.

ТАБЛИЦА 12-1 Операторы тестирования для скриптов bash

Оператор	Описание
STRING1 = *STRING2*	**Истинно**, если две строки равны
STRING1! = *STRING2*	**Истинно**, если две строки не равны
INTEGER1 **-eq** *INTEGER2*	**Истинно**, если два целых числа равны
INTEGER1 **-ne** *INTEGER2*	**Истинно**, если два целых числа не равны
INTEGER1 **-ge** *INTEGER2*	**Истинно**, если **INTEGER1** больше или равен **INTEGER2**
INTEGER1 **-gt** *INTEGER2*	**Истинно**, если **INTEGER1** больше, чем **INTEGER2**
INTEGER1 **-le** *INTEGER2*	**Истинно**, если **INTEGER1** меньше или равен **INTEGER2**
INTEGER1 **-lt** *INTEGER2*	**Истинно**, если **INTEGER1** меньше, чем **INTEGER2**
-d *FILE*	**Истинно**, если ***FILE*** является каталогом
-e *FILE*	**Истинно**, если ***ФАЙЛ*** существует
-f *FILE*	**Истинно**, если ***FILE*** существует и является обычным файлом
-r *FILE*	**Истинно**, если ***FILE*** существует и ему предоставлены разрешения на чтение
-w *FILE*	**Истинно**, если ***FILE*** существует и ему предоставлены права на запись
-x *FILE*	**Истинно**, если ***FILE*** существует и ему предоставлены разрешения на выполнение

Оператор **if** обычно связан с оператором **then** и, возможно, оператором **else**. Например, возьмем следующий гипотетический блок:

```
if [ -e /etc/fstab];
then
        cp /etc/fstab /etc/fstab.bak
else
        echo "Don't reboot, /etc/fstab is missing!"
fi
```

В этом коде, если файл **/etc/fstab** существует (любезно предоставлено **-e**), запускается команда, связанная с оператором **then**. Если этот файл отсутствует, отображается отмеченное сообщение.

Пример: сценарий 0anacron

Мы суммировали назначение скрипта **0anacron** в **главе 9**, но вы подробно проанализируете его здесь. Вы можете найти скрипт в каталоге **/etc/cron.hourly**. Копия скрипта показана на **рисунке 12-2**.

Сценарий начинается со строки **shebang**, которая сообщает **Linux**, что это **bash-скрипт**. Затем, есть **следующий блок if**:

```
if test -r /var/spool/anacron/cron.daily; then
        day=`cat /var/spool/anacron/cron.daily`
fi
```

Оператор **test** иногда используется в качестве условия в **if**. Например, строка **if test -r /var/spool/anacron/cron.daily**;

РИСУНОК 12-2 Скрипт 0anacron

```
[root@server1 ~]# cat /etc/cron.hourly/0anacron
#!/bin/sh
# Check whether 0anacron was run today already
if test -r /var/spool/anacron/cron.daily; then
    day=`cat /var/spool/anacron/cron.daily`
fi
if [ `date +%Y%m%d` = "$day" ]; then
    exit 0;
fi

# Do not run jobs when on battery power
if test -x /usr/bin/on_ac_power; then
    /usr/bin/on_ac_power >/dev/null 2>&1
    if test $? -eq 1; then
    exit 0
    fi
fi
/usr/sbin/anacron -s
[root@server1 ~]#
```

Это функционально эквивалентно

if [-r /var/spool/anacron/cron.daily];

Этот **блок if** проверяет, существует ли файл /**var/spool/anacron/cron.daily** и доступен ли он для чтения. Если тест пройден успешно, содержимое файла cron.daily сохраняется в **переменной day**. Фактически, файл **cron.daily** содержит последнюю дату (в формате **ГГГГММДД**), когда был запущен анакрон.

Следующие строки содержат другой **блок if**:

```
if [ `date +%Y%m%d` = "$day" ]; then
        exit 0
fi
```

Этот код сравнивает две строки: текущую дату, возвращаемую командой **date** в формате **ГГГГММДД** (обратите внимание на пометки, чтобы заменить вывод команды даты в качестве первого операнда при сравнении тестов), и содержимое переменной дня. Хорошей практикой является то, что имя переменной дня заключено в двойные кавычки, чтобы любые специальные символы в строке в кавычках, кроме знака доллара, не интерпретировались **bash**.

Если две даты равны, скрипт немедленно завершает работу со **значением 0**, что указывает на отсутствие ошибок. Другими словами, если анакрон уже был запущен сегодня, содержание файла **/var/spool/anacron/cron.daily** будет содержать сегодняшнюю дату. В этом случае скрипт не будет запущен во второй раз и завершится со **значением 0**.

Следующий раздел кода содержит два вложенных **блока if**:

```
if test -x /usr/bin/on_ac_power; then
        /usr/bin/on_ac_power >/dev/null 2>&1
        if test $? -eq 1; then
        exit 0
        fi
fi
```

Первая **инструкция if** проверяет, существует ли файл **/usr/bin/on_ac_power** и является ли он исполняемым. Если это так, он запускает программу и подавляет все свои выходные данные, перенаправляя стандартный вывод и стандартную ошибку в **/dev/null**. Как указано в справочной странице **on_ac_power**, эта команда возвращает **код выхода 0**, если система подключена к сети, **и 1** в противном случае.

Затем скрипт проверяет **код выхода ($?)** Последней команды. Если **это 1** (то есть, если система не подключена к сети переменного тока), сценарии завершаются со **значением 0**.

Наконец, если все предыдущие тесты пройдены, скрипт запускает команду **anacron**:

```
/usr/bin/anacron –s
```

В свою очередь, **anacron** прочитает список заданий из **/etc/anacrontab** и выполнит их в последовательном (-ых) порядке.

Цикл for

Цикл **for** выполняет список команд для всех элементов, указанных в списке. Это довольно просто и имеет разные формы. В следующем примере команда цикла **for** выполняется три раза для каждого значения переменной **n в списке 1, 2, 3**:

```
for n in 1 2 3; do
        echo "I love Linux #$n"
done
```

Вывод предыдущего фрагмента кода:

```
I love Linux #1
I love Linux #2
I love Linux #3
```

Другой пример существует в скрипте **certwatch** в каталоге **/etc/cron.daily**. Если вы не видите его в своей системе, установите пакет **crypto-utils**.

Здесь список в цикле **for** заменяется значением переменной:

```
for c in $certs; do
        # Check whether a warning message is needed, then issue one if so.
```

```
        /usr/bin/certwatch $CERTWATCH_OPTS -q "$c" &&
                /usr/bin/certwatch $CERTWATCH_OPTS "$c" | /usr/bin/sendmail –oem \
                -oi -t 2>/dev/null
done
```

Переменная **$certs** содержит список всех файлов сертификатов, используемых **веб-сервером Apache**. Команда **for** проходит каждый сертификат и проверяет, не истекает ли срок его действия. Если это так, он отправляет предупреждение.

Обратите внимание на оператор **&&** между двумя командами **certwatch**. Он говорит **bash** выполнить вторую команду, только если первая успешна (то есть, если она возвращает состояние 0).

Более сложный пример показан ниже. **Цикл for** выполняется для всех пользователей в системе, что возвращается **командой getent passwd**:

```
for username in $(getent passwd | cut -f 1 -d ":"); do
        usergroups=$(groups $username | cut -f 2 -d ":")
        echo "User $username is a member of the following groups: $usergroups"
done
```

В первой строке команда **getent passwd** возвращает всех пользователей в системе. Это может включать пользователей, определенных локально в **/etc/passwd**, а также пользователей, определенных в центральной службе каталогов, такой как **LDAP**. Выходные данные команды усекаются до первого столбца (**-f 1**), определяемого символом-разделителем (**-d ":"**). Это дает список имен пользователей, которые **цикл for** может циклически проходить и назначать переменной **username** на каждой итерации.

Затем предыдущий фрагмент кода выполняет команду **groups** с каждым именем пользователя в качестве аргумента. Эта команда возвращает группы, в которые входит пользователь, в следующем формате:

```
user : group1 group2 ...
```

Команда **cut -f 1 -d ":"** извлекает все выходные данные после разделителя столбцов, а результат сохраняется в переменной **usergroups**. Наконец, результат отображается командой **echo**.

Аргументы скрипта

Вы можете использовать аргументы для передачи информации в скрипт, так же, как вы делали бы это с обычными командами. В сценарии **bash** первый аргумент команды сохраняется в специальной переменной $1, второй в **$2** и т.д. Общее количество аргументов сохраняется в специальной переменной **$#.** В качестве примера рассмотрим следующий скрипт:

```
#!/bin/bash
echo "The number of arguments is $#"
if [ $# -ge 1 ]; then
        echo "The first argument is $1"
fi
```

Сохраните код в файле с именем **args.sh** и сделайте его исполняемым с помощью команды **chmod + x args.sh**. Затем запустите программу, как показано:

```
# ./args.sh orange
```

Вы должны увидеть следующий вывод:

The number of arguments is 1
The first argument is orange

В упражнении 12-1 у вас будет возможность применить эти уроки на практике.

УПРАЖНЕНИЕ 12-1

Создать скрипт

В этой лабораторной работе вы создадите скрипт с именем **get-shell.sh**. Сценарий принимает имя пользователя в качестве первого аргумента и отображает оболочку по умолчанию для указанного пользователя в следующем формате:

 # ./get-shell.sh mike
mike's default shell is /bin/bash

Если аргумент не указан, сценарий должен отображать оболочку по умолчанию для текущего пользователя. Если задано более одного аргумента, сценарий должен распечатать следующее сообщение об ошибке и выйти со значением 1:

Error: too many arguments

Если пользователь, указанный в качестве аргумента, не существует, сценарий должен отобразить следующее сообщение об ошибке и выйти со значением 2:

Error: cannot retrieve information for user <user>

1. Создайте файл с именем **get-shell.sh** и назначьте разрешения на выполнение для этого файла:

   $ touch get-shell.sh
   $ chmod + x get-shell.sh

2. Откройте файл в вашем любимом редакторе. Запустите скрипт со следующей строкой:

   #!/bin/sh

3. Добавьте следующие строки, которые проверяют, больше ли число аргументов (**$#),** чем один. Если это так, выведите сообщение об ошибке и выйдите со значением 1:

   ```
   if [ $# -gt 1 ]; then
           echo "Error: too many arguments"
           exit 1
   fi
   ```

4. Добавьте следующие строки. Если аргументы не переданы, сценарий сохраняет имя текущего пользователя (**$USER**) в переменной **username**. В противном случае переменная **username** принимает значение первого аргумента (**$1**). Чтобы выразить эту логику, мы используем конструкцию **if-then-else**:

   ```
   if [ $# -eq 0 ]; then
           username=$USER
   else
           username=$1
   ```

```
fi
```

5. Получить информацию о пользователе. Вы можете запросить базу данных пользователей с помощью команды **getent passwd**. Эта команда возвращает информацию о пользователе из локального файла **/etc/passwd** и из любых настроенных систем каталогов:

```
userinfo=$(getent passwd $username)
```

6. Проверьте значение выхода предыдущей команды. Любое ненулевое значение выхода означает, что произошла ошибка. Если это так, немедленно выйдите из программы со статусом выхода 2:

```
if [ $? -ne 0 ]; then
        echo "Error: cannot retrieve information for user $username"
        exit 2
fi
```

7. Извлеките оболочку пользователя из переменной **userinfo**. Это седьмое поле (**-f 7**) **/etc/passwd**, где каждое поле разделено символом столбца **(-d ":")**:

```
usershell=$(echo $userinfo | cut -f 7 -d ":")
```

8. Распечатайте результат. В качестве хорошей практики выйдите со значением 0, чтобы указать, что ошибок не было:

```
echo "$username's shell is $usershell"
exit 0
```

9. Сохраните ваши изменения. Выполните скрипт с разными аргументами, чтобы проверить все возможные условия:

```
$ ./get-shell.sh alex
alex's shell is /bin/bash
$ ./get-shell.sh mike
mike's shell is /bin/bash
$ ./get-shell.sh daemon
daemon's shell is /sbin/nologin
$ ./get-shell.sh mikes
Error: cannot retrieve information for user mikes
$ ./get-shell.sh alex mike
Error: too many arguments
```

ЦЕЛЬ СЕРТИФИКАЦИИ 12.02

Настройка отчетов об использовании системы

Администратору полезно знать, когда система перегружена. Чтобы помочь вам, RHEL 7 включает пакет **sysstat**. Кроме того, есть другие команды, связанные с использованием измерительной системы, в частности, **top**. Конечно, вы можете определить текущее использование диска с помощью таких команд, как **df и fdisk**. Как только отчеты об использовании системы собраны, вы можете просмотреть результаты, чтобы определить моменты, когда система интенсивно используется.

Перефразируя соответствующую цель **RHCE**, существуют другие важные команды, которые могут помочь вам «**создавать и доставлять отчеты**» о нагрузке на **ЦП, ОЗУ, жесткие диски и сети**. Хотя они собирают данные, аналогичные командам, таким как **top, df и fdisk**, команды, связанные с пакетом **sysstat**, собирают такие данные по каждому из отмеченных компонентов. Данные о производительности собираются в лог-файлы. Затем команда **sadf** предназначена для фактического использования этих данных журнала для подготовки отчета. Когда такие отчеты записываются в соответствующий текстовый файл или файл базы данных, они могут быть переданы для оценки и обработки.

РИСУНОК 12-4 Команда dstat отображает использование системы.

```
[root@server1 ~]# dstat
You did not select any stats, using -cdngy by default.
----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send|  in   out | int   csw
  1   0  99   0   0   0|3019B  989B|   0     0 |   0     0 |   5    11
 42   1  57   0   0   0|   0     0 |   0     0 |   0     0 | 519   667
 43   1  56   0   0   0| 120k    0 | 104B    0 |   0     0 | 519   919
 88   3   9   0   0   0|   0     0 |  66B  163B|   0     0 |1003  1841
 77   2  21   0   0   0|   0  1956k| 104B    0 |   0     0 | 918  1641
 73   2  24   0   0   0|  24k    0 |2346B 1862B|   0     0 | 885  1145
 17   2  81   0   0   0|   0     0 | 104B    0 |   0     0 | 345  1050
 27   2  71   0   0   0|   0     0 |   0     0 |   0     0 | 442   827
 47   1  52   0   0   0|   0     0 | 104B    0 |   0     0 | 587   874
 22   1  77   0   0   0|   0     0 |   0     0 |   0     0 | 285   335
 26   0  74   0   0   0|   0     0 | 104B    0 |   0     0 | 322   376
 92   3   5   0   0   0|  84k   68k|2092B 1224B|   0     0 |1037  1414
 95   4   0   0   0   1|1288k  452k| 237k   10k|   0     0 |1203  1508
 95   4   1   0   0   0|8192B    0 | 163k   15k|   0     0 |1214  1595
 40   3  56   1   0   0| 128k    0 | 190B  173B|   0     0 | 666  1279
 38   2  60   0   0   0|   0     0 |3917B 6707B|   0     0 | 649  1112
 39   1  60   0   0   0|   0    27k| 146B  156B|   0     0 | 637  1099
 57   3  40   0   0   0|   0     0 |   0     0 |   0     0 | 772  1274
 41   1  58   0   0   0|   0     0 | 104B    0 |   0     0 | 604   966
 39   2  59   0   0   0|   0     0 |   0     0 |   0     0 | 565   902
```

Файл **/etc/cron.d/sysstat**. Пакет также содержит ряд связанных команд, которые описаны здесь.

Команды, входящие в состав **sysstat**, используют параметры, показанные в файлах **sysstat** и **sysstat.ioconf** в каталоге **/etc/sysconfig**. Файл **sysstat** относительно прост; следующая директива указывает, что файлы журналов должны храниться в **течение 28 дней**:

HISTORY=28

И эта директива указывает, что файлы журналов старше 31 дня должны быть сжаты:

COMPRESSAFTER=31

Конечно, это означает, что файлы журналов стираются до того, как они могут быть сжаты. Естественно, вы можете изменить любую переменную по мере необходимости. Содержательный файл **/etc/sysconfig** - это **sysstat.ioconf**, поскольку он помогает собирать данные о деятельности с различных устройств хранения. Это помогает некоторым командам пакета **sysstat** собирать данные с дисковых устройств. Несмотря на то, что файл **sysstat.ioconf** имеет большой размер, изменения в этом файле не требуются, если только не появилось новое оборудование для хранения на диске, а экзамены **Red Hat** не являются экзаменами на оборудование.

Собрать информацию о состоянии системы в журналы

Пакет **sysstat** включает в себя обычную работу **cron**. Доступное в каталоге **/etc/cron.d**, это задание собирает информацию об использовании системы и отправляет ее в файлы журналов в **/var/log/sa** каталог. Проверьте файл **sysstat** в каталоге **/etc/cron.d**. Первая строка определяет задание, которое запускается каждые 10 минут пользователем **root**:

***/10 * * * * root /usr/lib64/sa/sa1 1 1**

Команда **sa1** с 1 и 1 в конце указывает, что команда должна выполняться один раз, через одну секунду после запуска задания. Информация из этой команды собирается в файле с именем ***sadd*** в каталоге **/var/log/sa**, где **dd** представляет день месяца.

Следующая строка более мощная, чем кажется. Ежедневно, за семь минут до полуночи, с привилегиями администратора **root**, команда **sa2** записывает ежедневный отчет о большинстве операций системы.

53 23 * * * root /usr/lib64/sa/sa2 -A

Ключ **-A** связан с командой **sar**. Как следует из следующего отрывка из справочной страницы **sar**, он собирает все разумные сведения об использовании системы:

-A This is equivalent to specifying -bBdqrRSuvwWy
-I SUM -I XALL -n ALL -u ALL -P ALL.

Подготовить отчет о состоянии системы

Этот раздел не будет готовить отчет для презентации. Это просто анализ команды **sadf** и того, как ее можно использовать для указания информации для фильтрации из файлов журнала в каталог **/var/log/sa**. Двоичные файлы журнала с именами, такими как **sa10** (для 10-го числа месяца), могут быть обработаны несколькими способами с помощью команды **sadf**. Некоторые из наиболее важных переключателей **sadf** перечислены в **таблице 12-2**.

Например, следующая команда устанавливает отчет с данными между началом и концом **10-го** числа месяца:

 # sadf -s 00:00:01 -e 23:59:59 /var/log/sa/sa10 > activity10

Данные перенаправляются в файл **activity10** для последующей обработки. Но мощь пакета **sysstat** зависит от того, как он взаимодействует с командой **sar**. Однако только некоторые параметры команды **sar** работают с **sadf**. Как предлагается на справочной странице **sadf**, следующая команда подготавливает отчет на основе «**памяти, пространства подкачки и сетевой статистики**» из файла **/var/log/sa/sa21** в формате, который может обрабатываться базой данных:

 # sadf -d /var/log/sa/sa21 -- -r -n DEV

ТАБЛИЦА 12-2. Опции для команды sadf

Переключатель	Описание
-d	Отображает содержимое в формате, используемом системой реляционных баз данных.
-e hh:mm:ss	Перечисляет время окончания отчета в 24-часовом формате.
-p	Отображает содержимое в формате, используемом командой awk; не используйте с **-d или -x**.
-s hh:mm:ss	Перечисляет время начала отчета в 24-часовом формате.
-x	Отображает содержимое в формате XML; не используйте с -d или -p.

Хотя ключ **-d** связан с командой **sadf**, двойная черта (-) указывает на опции, связанные с командой **sar**. Таким образом, **ключ -r** сообщает об использовании памяти и **-n DEV** сообщает статистику с сетевых устройств.

Страница **man sadf** - отличная справочная информация о параметрах команд, необходимых для создания отчета во время работы или даже во время экзамена **Red Hat**. Как и во многих других командах, вы можете найти примеры в разделе примеров на странице руководства.

Конечно, есть и другие важные переключатели команды **sar**. Те из них, которые могут быть полезны при подготовке отчета об использовании «**процессора, памяти, диска и сети**», описаны в **таблице 12-3**.

С помощью опций, перечисленных в **таблице 12-3**, вы можете изменить предыдущую команду **sadf**, чтобы она соответствовала всем четырем элементам, перечисленным в соответствующей задаче **RHCE**:

 # sadf -d /var/log/sa/sa21 -- -u -r -dp -n DEV

ТАБЛИЦА 12-3 Параметры использования системы для команды sar

Переключатель	Описание
-d	Списки блокируют активность устройства. Обычно используется с **-p** для указания общих имен файлов устройств, таких как **sda** и **sdb**.
-n DEV	Сообщает статистику с сетевых устройств.
-P cpu	Выводит статистику для каждого процессора (или ядра); например, **-P 0** указывает первый **cpu**.
-r	Сообщает статистику использования памяти.
-S	Показывает статистику использования пространства подкачки.
-u	Отчеты об использовании процессора, включая связанные категории, пользователя, системы и времени простоя, и многое другое.
-W	Отчеты статистик

Другими словами, команда **sadf** указывает выходные данные, используемые базой данных **(-d**) из файла базы данных в каталоге **/var/log/sa**, связанном **с 21-м числом месяца**. Двойная черта (--) указывает на переключатели команды **sar**, где загрузка **CPU (-u),** использование **ОЗУ (-r)** и активность по блочному устройству **(-d)** представлены в более знакомых именах блочных устройств, таких как **sda (-p)**, и со статистикой с сетевых устройств **(-n DEV)**.

ЦЕЛЬ СЕРТИФИКАЦИИ 12.03

Параметры времени выполнения ядра

Параметры времени выполнения ядра, определенные в целях **RHCE**, относятся к файлам в каталоге **/proc/sys** и команде **sysctl**. С ним тесно связан файл конфигурации **/etc/sysctl.conf**, который используется командой **sysctl** во время процесса загрузки для настройки параметров на различные файлы в каталоге **/proc/sys**. Поэтому целесообразно начать этот раздел с просмотра этого файла **sysctl.conf**.

Как sysctl работает с /etc/sysctl.conf

Вы можете включить **пересылку IPv4 в два этапа**. Сначала добавьте следующую логическую директиву для активации пересылки IPv4 в конфигурации:

net.ipv4.ip_forward = 1

Затем заставьте систему перечитать файл конфигурации с помощью следующей команды:

 # sysctl –p

Давайте рассмотрим этот процесс немного подробнее. Во-первых, параметры времени выполнения ядра задокументированы в различных файлах в каталоге **/proc/sys**. Содержащая **net.ipv4** переменная **.ip_forward** хранится в файле **ip_forward**, в подкаталоге **net/ipv4/.** Другими словами, пересылка IPv4 описана в файле **ip_forward**, в каталоге **/proc/sys/net/ipv4**.

Поскольку этот файл содержит **0 или 1**, это логическая переменная. Таким образом, значение **1** для переменной **net.ipv4.ip_forward** активирует пересылку **IPv4**.

Что если вы хотите добавить пересылку **IPv6**? Хотя это не настроено в файле **/etc/sysctl.conf**, это функция, которую вы можете добавить. Пересылка **IPv6** может быть установлена в файле с именем **forwarding**, в каталоге **/proc/sys/net/ipv6/conf/all**. Другими словами, чтобы установить пересылку **IPv6** при перезагрузке, вы должны включить следующую директиву в **/etc/sysctl.conf**:

net.ipv6.conf.all.forwarding=1

Подобные директивы будут работать для других настроек, связанных с файлами в каталоге **/proc/sys**. Посмотрите на директивы **icmp_*** в каталоге **/proc/sys/net/ipv4**. Вы могли бы признать, что протокол управляющих сообщений Интернета (**ICMP**) иногда ассоциируется с помощью команды **ping**. Фактически, команда **ping** является запросом эха. Таким образом, **icmp_echo_ignore_all** и **icmp_echo_ignore_broadcasts** относятся напрямую к команде **ping**, а также к команде **ping**, связанной с широковещательным адресом.

Другими словами, если вы добавите директивы

net.ipv4.icmp_echo_ignore_all = 1
net.ipv4.icmp_echo_ignore_broadcasts = 1

в файл **/etc/sysctl.conf** локальная система не будет отвечать на команду прямого **ping** и не будет отвечать на запрос, сделанный **ping** к широковещательному адресу для сети.

Настройки в файле /etc/sysctl.conf

Настройки в файле **/etc/sysctl.conf** - это небольшая часть того, что можно настроить. В RHEL 7 файл **/etc/sysctl.conf** содержит только комментарии, а конфигурация по умолчанию была перемещена в файлы в каталоге **/usr/lib/sysctl.d**. Посмотрите на эти файлы. Справедливо предположить, что **RHEL 7** включает опции в этих файлах по какой-то причине, и эти настройки, скорее всего, будут рассмотрены на экзамене **RHCE**. Вы уже изучили первую директиву для пересылки **IPv4**. Следующая директива включена в файл **50-default.conf** в каталоге **/usr/lib/sysctl.d.** Если активен, он гарантирует, что пакеты, которые приходят с внешней сети фактически является внешней, выполняя проверку переадресации обратного пути:

net.ipv4.conf.default.rp_filter = 1

Следующая директива обычно отключается в качестве меры безопасности, чтобы избежать потенциальной атаки с использованием исходной маршрутизации:

net.ipv4.conf.default.accept_source_route = 0

Также известный как ключ ядра **sysrq**, разработчики могут изменить значение этой директивы для целей разработки. Как правило, вы должны сохранить настройки по умолчанию:

kernel.sysrq = 16

Если происходит сбой **ядра Linux**, этот параметр включает в себя номер **PID** с файлом дампа ядра ядра, чтобы помочь определить виновника:

kernel.core_uses_pid = 1

Другим стандартным методом, используемым хакерами white-hat для перегрузки системы, является поток пакетов **SYN**. Это похоже на так называемый «пинг смерти». Следующая настройка позволяет избежать перегрузки:

net.ipv4.tcp_syncookies = 1

Мост - это более старый термин для коммутатора, который может передавать трафик между различными сегментами сети. Следующие директивы, включенные в файл **00-system.conf** в каталоге **/usr/lib/sysctl.d**, запрещают использование отмеченных фильтров **iptables, ip6tables и arptables** на таких мостах:

net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0

Такие мосты обычно связаны с виртуальными сетями на хосте **KVM**.

УПРАЖНЕНИЕ 12-2

Отключить ответы на команду ping

В этом упражнении вы будете использовать параметры ядра, чтобы отключить ответы на команду **ping**. Хотя это упражнение можно запустить на любых двух подключенных системах, предполагается, что вы настроите систему **server1.example.com** и протестируете результат на **tester1.example.com** системах.

1. В системе **server1.example.com** просмотрите текущую настройку, связанную с ответами на сообщения **ping**, с помощью следующей команды:

   # cat /proc/sys/net/ipv4/icmp_echo_ignore_all

2. Предполагая, что вывод равен 0, попробуйте команду **ping localhost**. Что просходит? Не забудьте нажать **Ctrl-C** для выхода из потока вывода. Если вывод равен 1, перейдите к шагу 5.
3. Подтвердите результат из удаленной системы, такой как **tester1.example.com**. В некоторых ситуациях у вас может не быть физического доступа к этой системе, поэтому подключитесь с помощью соответствующей команды **ssh**. В удаленной системе попробуйте команду **ping server1.example.com** или **ping 192.168.122.50**.
4. Вернитесь в систему **server1.example.com.** Измените настройки ядра, описанные в шаге 1, с помощью следующей команды:

   # echo "1"> /proc/sys/net/ipv4/icmp_echo_ignore_all

   Подтвердите, повторив команду из шага 1. Повторите команду ping localhost. Что просходит?
5. Восстановите исходную настройку **0** в параметре **icmp_echo_ignore_all**.

ЦЕЛЬ СЕРТИФИКАЦИИ 12.04

IP-маршруты

Как описано в целях **RHCE**, вам необходимо знать, как «**маршрутизировать IP-трафик и создавать статические маршруты**». Это действительно две задачи. Во-первых, это стандартная часть конфигурации сети для настройки маршрута по умолчанию к внешней сети. Но есть также связанная с этим задача, когда система имеет два или более сетевых устройства, установить статический маршрут к определенной сети.

Настройте маршрут по умолчанию

Маршрут по умолчанию - это путь, используемый сетевым пакетом, когда нет других более конкретных маршрутов для этого адреса назначения. Когда сервер протокола динамической конфигурации хоста (**DHCP**) работает и настроен на предоставление шлюзу по умолчанию **IP-адресов**, маршрут по умолчанию назначается **IP-адресу**, полученному сервером **DHCP**. Это обычно проявляется в выводе команды **ip route**, рассмотренной в главе 3. Здесь показан один из примеров такого вывода для системы, которая использует **DHCP**-сервер:

default via 192.168.122.1 dev eth0 proto static metric 1024
192.168.122.0/24 dev eth0 proto kernel scope link src 192.168.122.50

Чтобы просмотреть, маршрут по умолчанию проходит через адрес шлюза 192.168.122.1. Аналогичным образом маршрут по умолчанию для статически сконфигурированной сетевой системы настраивается с помощью директивы **GATEWAY** в ее файле конфигурации. Такие файлы конфигурации хранятся в каталоге **/etc/sysconfig/network-scripts** с такими именами, как **ifcfg-eth0**.

Но существуют ситуации, такие как временная проблема сети, когда маршрут по умолчанию не предоставляется сервером **DHCP**. Возможно, **DHCP-сервер** необходимо заменить, и вам придется настроить информацию о статическом **IP-адресе**. В таких случаях маршрут по умолчанию можно временно добавить с помощью команды **ip route**. Например, следующая команда восстановит маршрут по умолчанию, показанный ранее:

 # ip route add default via 192.168.122.1 dev eth0

Чтобы убедиться, что маршрут по умолчанию сохраняется после перезагрузки, необходимо убедиться, что либо система настраивает **IP-адрес шлюза по умолчанию**, как часть статической конфигурации сети, либо **DHCP-сервер**, используемый для сети, может назначить этот **IP-адрес шлюза**. Чтобы рассмотреть, **рисунок 12-5** отражает способ настройки **IPv4-адреса шлюза по умолчанию** с помощью инструмента **Network Manager**. В качестве альтернативы, вы можете убедиться, что добавленный маршрут по умолчанию выживет после перезагрузки путем прямого изменения файла конфигурации **ifcfg-ethx**.

Некоторые системы могут иметь несколько сетевых устройств. В этом случае вам может потребоваться настроить статический маршрут.

!!!!!EXAM WATCH
Фиктивный интерфейс - это особый тип виртуального интерфейса, который не связан ни с одним сетевым адаптером в системе. Вы можете использовать фиктивный интерфейс, чтобы практиковаться в определенных сетевых сценариях, когда у вас нет доступа к физической сети или ваша система отключена.
!!!!!!

РИСУНОК 12-6 Статический маршрут для определенного сетевого назначения

УПРАЖНЕНИЕ 12-3

Практика со статическими маршрутами

В этом упражнении вы создадите фиктивный интерфейс для практики настройки статических маршрутов. Фиктивный интерфейс - это виртуальный интерфейс, который не связан ни с одним адаптером на хосте. В этом упражнении предполагается, что вы будете настраивать фиктивный интерфейс на системе **server1.example.com**, в то время как статический маршрут будет добавлен к физической системе хоста.

1. На **server1.example.com** выполните следующие команды, чтобы добавить фиктивный интерфейс. Убедитесь, что диапазон **IP 192.168.123.0/24** еще не используется в вашей сети. Если это так, выберите другой диапазон сети:

   # modprobe dummy
   # ip link set name eth2 dev dummy0
   # ip address add 192.168.123.123/24 dev eth2
   # ip link set eth2 up

2. Запустите команду **ping 192.168.123.123** на сервере **server1.example.com**. Если вы правильно настроили фиктивный интерфейс, вы должны получить ответ на ваши запросы **ping**. Не забудьте нажать **Ctrl-C** для выхода из потока вывода.
3. Запустите команду **ip route на сервере server1.example.com**. Вы увидите действительный маршрут к **192.168.123.0/24**, потому что этот сегмент сети напрямую подключен к фиктивному интерфейсу eth2 **192.168.123.0/24 dev eth2 proto kernel scope link src 192.168.123.123**

4. Повторите команду **ping 192.168.123.123** с физического хоста. Поскольку у вашего физического хоста, вероятно, нет маршрута к **192.168.123.0/24** через сервер1, ваша команда **ping** не получит ответ.
5. Добавьте статический маршрут к **192.168.123.0/24** на вашем физическом хосте. Для этого откройте инструмент «**Редактор соединений диспетчера сети** (**Network Manager Connection Editor**)». Выберите устройство моста **virbr0** и нажмите «**Изменить**». На вкладке «**Настройки IPv4**» нажмите кнопку «**Маршруты**», чтобы добавить статический

маршрут. Установите **192.168.123.0** в качестве сетевого адреса, 24 в качестве маски сети и **192.168.122.50 (IP-адрес server1)** в качестве шлюза.

6. Перезапустите **Network Manager**, вот так:

   # systemctl restart NetworkManager

7. Убедитесь, что маршрут к **192.168.123.0/24** установлен в таблице маршрутизации с помощью команды **ip route**.
8. Повторите команду **ping 192.168.123.123** с вашего физического хоста. Что просходит?
9. Удалите статический маршрут на физическом хосте.
10. Удалите фиктивный интерфейс на сервере server1:

    # ip link delete eth2

ЦЕЛЬ СЕРТИФИКАЦИИ 12.05

Введение в IPv6

Одной из особых задач экзамена **RHCE** является создание сетей **IPv6**. Хотя большинство современных сетей настроены на использование адресов **IPv4**, в некоторых регионах исчерпаны общедоступные адреса **IPv4**.

Интернет-протокол версии 6 (**IPv6**) был введен в конце 1990-х годов в качестве замены **IPv4**. Оказывается, 4 миллиарда (232) адресов IPv4 недостаточно. IPv6 поддерживает гораздо больше адресов, потенциально до 2128 или 3,4 × 1038 (340 ундециллионов) адресов.

Базовая адресация IPv6

В главе 3 мы ввели нотацию «точка-десятичность» для адресов IPv4, где каждый десятичный октет представляет 8 бит 32-битного адреса (например, 192.168.122.50). Адреса IPv6 состоят из 128 битов и устанавливаются в шестнадцатеричном формате, также известном как основание 16. Другими словами, адрес IPv6 может содержать следующие «цифры»:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f

Адрес **IPv6** обычно состоит из восьми групп по четыре шестнадцатеричных числа в каждой, называемых «полубайтами», в следующем формате:

2001:0db8:3dab:0001:0000:0000:0000:0072

Вы можете упростить адресацию **IPv6** далее:

- Удалите все ведущие нули в клеве. Например, вы можете записать 0db8 как db8, 0072 как 72, 0000 как 0 и так далее.
- Замените любую последовательность грызунов 0000 парой двоеточий (: :). Например, вы можете сократить 0000: 0000: 0000 с помощью пары двоеточий. Однако, чтобы избежать двусмысленности, вы можете применять это правило только один раз для адреса IPv6.

Следовательно, мы можем переписать предыдущий адрес в гораздо более компактной форме:

2001:db8:3dab:1::72

Подобно **IPv4**, адреса **IPv6** состоят из двух частей: хоста и сетевого адреса. Часть хоста адреса **IPv6** называется «идентификатором интерфейса». В **IPv6** маски подсетей обычно указываются в префиксной нотации (например, /48).
В качестве примера предположим, что **IPv6-адрес 2001:db8:3dab:1::72** имеет сетевой префикс **/64**. Другими словами, сетевая часть этого адреса **IPv6** включает в себя первые **64 бита** этот адрес. В этом случае префикс сети - **2001:db8:3dab:1**. Идентификатор интерфейса включает в себя последние **64 бита**, показанные в виде шестнадцатеричного числа **72**.

Адреса **IPv6** подразделяются на несколько категорий. Во-первых, есть три формата адреса:

- **Unicast** Одноадресный адрес связан с одним сетевым адаптером.
- **Anycast** Anycast-адрес может быть назначен нескольким хостам одновременно. Может использоваться для балансировки нагрузки и резервирования. Адреса Anycast организованы так же, как и адреса одноадресной рассылки.
- **Multicast** Многоадресная рассылка Адрес многоадресной рассылки используется для одновременной отправки сообщения нескольким получателям.

При таком разнообразии форматов адресов широковещательные адреса в стиле **IPv4** не нужны. Если вы хотите отправить сообщение нескольким системам, используйте многоадресные адреса **IPv6**.

Адреса **IPv6** также организованы в нескольких различных диапазонах, как описано в **таблице 12-4**. Маршрут по умолчанию в **IPv4 (0.0.0.0/0)** отображается как **::/0 в IPv6**.

Диапазон адресов локальной ссылки требует объяснения. Каждый интерфейс в сети **IPv6** автоматически настраивается с локальным адресом связи. Эти адреса не маршрутизируются; как таковая связь ограничена сегментом локальной сети. Локальные адреса связи необходимы для различных операций **IPv6**.

Даже если вы не настроили **IPv6** на **серверах RHEL 7**, каждому сетевому интерфейсу автоматически назначается локальный адрес канала, как показано в следующем выводе:

```
# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast □
state UP qlen 1000
    link/ether 52:54:00:85:61:c0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.50/24 brd 192.168.122.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe85:61c0/64 scope link
        valid_lft forever preferred_lft forever
```

ТАБЛИЦА 12-4 Типы адресов IPv6

Тип адреса IPv6	Диапазон адресов	Описание
Global unicast	**2000::/3**	Используется для связи хост-хост.
Anycast	**Такой же как одноадресный**	Назначено на любое количество хостов.
Multicast	**ff00::/8**	Используется для связи «один ко многим» и «многие ко многим».
Link-local	**fe80::/10**	Зарезервировано для локальной связи.
Unique local	**fc00::/7**	Это эквивалент частных адресов RFC 1918 в IPv4.

Чтобы определить локальный адрес ссылки, найдите адрес, начинающийся с fe80. Обратите внимание на запись «Область действия». Как видите, интерфейс eth0 имеет следующий локальный IPv6-адрес: **fe80::5054:ff:fe85:61c0/64.**

Инструменты для устранения неполадок

Большинство сетевых инструментов, которые мы представили в **главе 3,** беспрепятственно работают с адресами **IPv4 и IPv6**. Есть два заметных исключения: команды **ping** и **traceroute**. Для работы в сети **IPv6** вы должны использовать команды **ping6** и **traceroute6**.

Команда **ping6** работает аналогично **ping**. Даже до того, как вы настроите **IPv6**-адрес, вы можете запустить команду **ping6** для локального адреса системы **server1.example.com**:

```
# ping6 -I virbr0 fe80::5054:ff:fe85:61c0
```

Так как локальные адреса канала не маршрутизируемы, вы должны указать исходящий интерфейс **(-I**) в команде **ping6** при проверке удаленного локального адреса канала.

Настройте адреса IPv6

Как и в случае сетей **IPv4**, вы можете настроить адрес **IPv6** с помощью инструмента командной строки **Network Manager nmcli**, текстового графического инструмента **nmtui** или редактора соединений **Network Manager**.

Запустите редактор соединений **Network Manager** из графического интерфейса пользователя командой **nm-connection-editor**.

Выделите профиль подключения первого устройства **Ethernet (eth0 в нашей системе)** и нажмите «**Изменить**»; затем перейдите на вкладку **Настройки IPv6**. Откроется окно, показанное на **рисунке 12-7**.

Щелкните раскрывающееся текстовое поле «**Метод**» и выберите «**Вручную**». Теперь вы можете добавить информацию об **IP-адресе** для системы. Например, на **server1.example.com** мы добавили следующие настройки:

```
IP Address 2001:db8:3dab:2
Prefix 64
Gateway Address 2001:db8:3dab:1
```

Аналогично, мы связали IPv6-адрес 2001:db8:3dab:1 с интерфейсом **virbr0** в нашей физической системе. Вы можете проверить конфигурацию с помощью следующей команды:

```
# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast □
state UP qlen 1000
    link/ether 52:54:00:85:61:c0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.50/24 brd 192.168.122.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 2001:db8:3dab::2/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe85:61c0/64 scope link
        valid_lft forever preferred_lft forever
```

РИСУНОК 12-7 Редактирование адреса IPv6 в редакторе соединений Network Manager

Конфигурация сохраняется в файле **ifcfg-eth0** в каталоге **/etc/sysconfig/network-scripts**. Откройте этот файл. Вы заметите, что редактор соединений **Network Manager** добавил следующие строки конфигурации:

IPV6_AUTOCONF=no
IPV6ADDR=2001:db8:3dab::2/64
IPV6_DEFAULTGW=2001:db8:3dab::1
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no

Директива **IPV6_AUTOCONF** отключает автоматически настраиваемые адреса **IPv6**. Следующие переменные, **IPV6ADDR** и **IPV6_DEFAULTGW**, устанавливают **IP-адреса** интерфейса и шлюза по умолчанию, соответственно, тогда как **IPV6_DEFROUTE** устанавливает маршрут по умолчанию в таблицу маршрутизации. Наконец, если включена директива **IPV6_FAILURE_FATAL**, то сбой в конфигурации **IPv6** приведет к отключению интерфейса, даже если конфигурация **IPv4** завершится успешно.

ЦЕЛЬ СЕРТИФИКАЦИИ 12.06
Связывание и объединение сетевых интерфейсов

В критически важных центрах обработки данных вы обычно подключаете сервер **Linux** к сети, подключая два его интерфейса **Ethernet** к различным коммутаторам доступа. Вы также обычно объединяете два физических порта в «**логический**» сетевой интерфейс (интерфейс «**bond**» или «**team**»). Эта конфигурация обеспечивает полное резервирование, поскольку один сбой не повлияет на способность сервера взаимодействовать с остальной частью сети. Кроме того, в некоторых конфигурациях сервер может активно отправлять и получать пакеты через оба сетевых интерфейса, удваивая доступную пропускную способность сети.
RHEL 7 предлагает два способа настройки таких конфигураций:

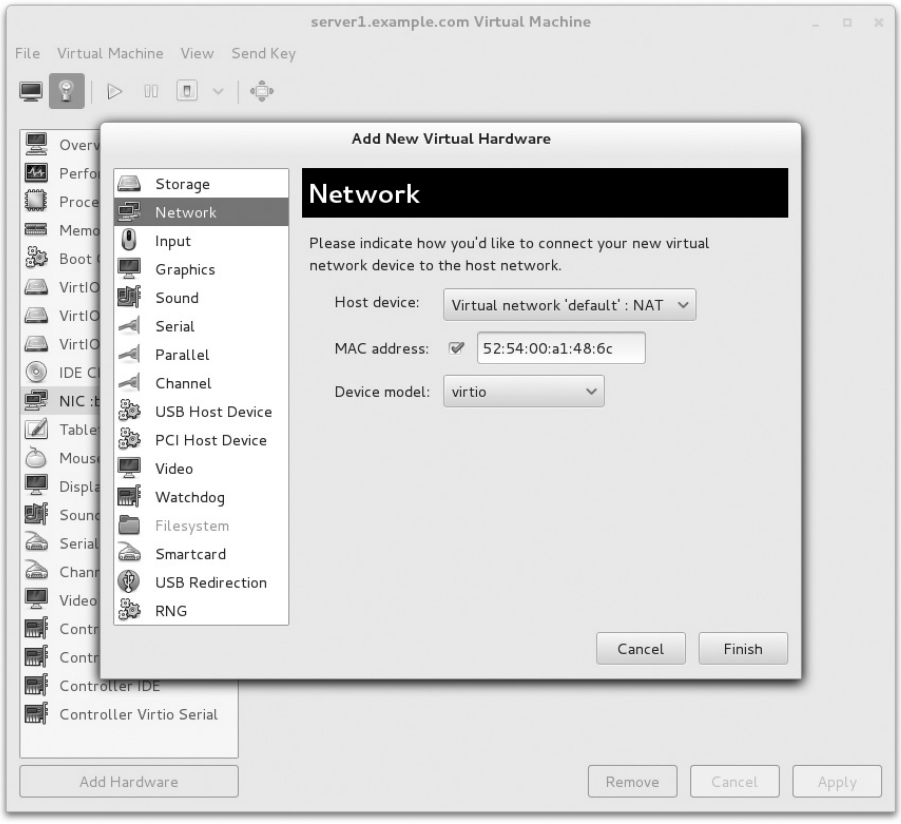
Interface bonding (Связывание интерфейса) Стандартный метод объединения в **RHEL 6** и все еще доступный в **RHEL 7**
Network teaming (Сетевое объединение) Введено в **RHEL 7**

На момент написания этих двух методов предлагались сходные функции, но объединение в сеть реализует более модульную и расширяемую конструкцию, чем традиционный драйвер связывания. Для экзамена RHCE (и для ваших повседневных рабочих обязанностей) вы должны быть знакомы с обоими методами конфигурации.

Чтобы попрактиковаться в связывании и объединении интерфейсов, начните с двух интерфейсов **Ethernet**. Для этого выключите виртуальную машину **server1.example.com** и добавьте второй адаптер **Ethernet**. Для этого запустите диспетчер виртуальных машин, откройте консоль виртуальной машины и окно сведений и нажмите кнопку сведений о виртуальном оборудовании. Нажмите **Add Hardware** и выберите сетевое устройство, как показано на **рисунке 12-8**. Установите «**virtio**» в качестве модели устройства и нажмите «**Готово**». Включите виртуальную машину и выполните команду **ip link show**, чтобы подтвердить, что новый виртуальный адаптер распознается системой. Вы должны увидеть один петлевой и два **Ethernet-адаптера**, установленных в вашей системе, как показано в следующем выводе:

```
# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast □
state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:b6:0d:ce brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast □
state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:a1:48:6c brd ff:ff:ff:ff:ff:ff
```

РИСУНОК 12-8 Добавьте новое сетевое устройство к виртуальной машине.



!!!! on the job

Как отмечено в Главе 3, RHEL 7 пытается назвать сетевые интерфейсы на основе их физического местоположения (например, «enoX» или «emX» для встроенных сетевых

интерфейсов). Если вы сконфигурировали виртуальные адаптеры в системах, использующих тип «virtio», как описано в этой главе, RHEL 7 должен вернуться к традиционному методу перечисления интерфейсов eth0, eth1 .... Если вы хотите заставить свою систему используя традиционный стиль именования ethX, вы можете применить процедуру, описанную в статье 283233 базы знаний по адресу https://access.redhat.com/solutions/283233.
!!!!!

Настроить связывание(bonding) интерфейсов

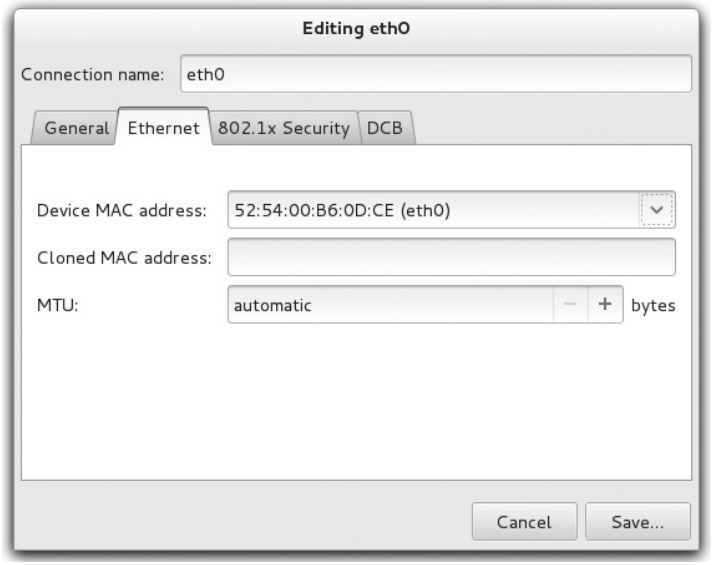
У вас есть несколько способов настройки связывания интерфейса: программа **nmcli** из командной строки, текстовый инструмент **nmtui** и графический редактор **Network Manager Connections**. Кроме того, если вы знаете синтаксис файлов конфигурации сети в **/etc/sysconfig/network-scripts/**, вы также можете создать новую конфигурацию, непосредственно отредактировав несколько файлов.

В этом разделе мы покажем, как настроить интерфейс связи на **server1.example.com** с помощью редактора **Network Manager Connections**. Цель состоит в том, чтобы объединить два интерфейса **eth0 и eth1** («**slave**» интерфейсы) в один логический интерфейс с именем «**bond0**» («**master**» интерфейс).

1. Запустите приложение Редактор **Network Manager Connections** из графического интерфейса с помощью команды **nm-connection-editor**.
2. Удалите любую существующую конфигурацию из интерфейса **eth0**. Выберите интерфейс в редакторе **Network Manager Connections** и нажмите «**Удалить**».
3. Нажмите кнопку «**Добавить**», выберите **Bond** в качестве типа подключения и подтвердите, нажав кнопку «**Создать**». Это открывает новое окно, как показано здесь:

4. Следующий шаг состоит в добавлении «**подчиненного(slave)**» интерфейса **eth0** к конфигурации связи. Нажмите кнопку «**Добавить**», выберите «**Ethernet**» в качестве типа подключения и нажмите «**Создать**».

5. Откроется окно Редактирование **Bond0 Slave 1**. Установите в качестве имени соединения значение **eth0** и в раскрывающемся меню установите MAC-адрес устройства на адрес интерфейса **eth0**, как показано здесь. Нажмите **Сохранить**.



6. Перейдите на вкладку «**Общие**» и выберите параметр «**Автоматически подключаться к этой сети, когда она доступна**». Нажмите «**Сохранить**». Это обеспечит активацию устройства при загрузке.
7. Повторите шаги 4, 5 и 6 для другого **подчиненного интерфейса eth1**.
8. Вернувшись к главному окну на первом рисунке, выберите в поле **режим *Active-backup*** в качестве режима отработки отказа. Доступные режимы для драйвера соединения обсуждаются в **таблице 12-5**.
9. При желании вы можете установить имя основного интерфейса в поле **Primary**.
10. Оставьте другие настройки в этом окне по умолчанию.
11. Перейдите на вкладку «**Настройки IPv4**». Настройте IP-адрес, маску сети и шлюз для системы с настройками из **таблицы 1-2 в главе 1.**
12. Нажмите **Сохранить**.

ТАБЛИЦА 12-5 Режимы склеивания (bonding)

Режим склеивания	Описание
Round-robin	Пакеты передаются в циклическом режиме через подчиненные интерфейсы. Обеспечивает балансировку нагрузки и отказоустойчивость. Требуется поддержка сетевых коммутаторов (например, настройка «канала порта» на устройствах Cisco).
Active-backup	Активен только один подчиненный интерфейс. Если этот активный интерфейс дает сбой, другой ведомый становится активным. Обеспечивает отказоустойчивость и не требует специальной поддержки переключателя.
XOR	Пакеты передаются через подчиненные интерфейсы с использованием политики хеширования XOR. Обеспечивает балансировку нагрузки для каждого потока и отказоустойчивость.
Broadcast	Пакеты передаются по всем подчиненным интерфейсам. Редко используемый.
802.3ad	Использует агрегацию каналов IEEE 802.3ad, которая должна поддерживаться сетевыми коммутаторами. Обеспечивает балансировку нагрузки и отказоустойчивость.

(Adaptive transmit load balancing) Адаптивная балансировка нагрузки передачи	Пакеты передаются через интерфейсы в зависимости от их текущей загрузки. Обеспечивает балансировку нагрузки и отказоустойчивость.
(Adaptive load balancing) Адаптивная балансировка нагрузки	Подобно адаптивной балансировке нагрузки передачи, но также обеспечивает балансировку входящей нагрузки посредством согласования **ARP**.

После завершения настройки у вас должен быть настроен интерфейс **bond0** в режиме активного резервного копирования с двумя подчиненными интерфейсами: **eth0 и eth1**. Следующая команда подтверждает текущие настройки конфигурации **IP**:

```
# ip addr show bond0
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc \
noqueue state UNKNOWN
        link/ether 52:54:00:b6:0d:ce brd ff:ff:ff:ff:ff:ff
        inet 192.168.122.50/24 brd 192.168.122.255 scope global dynamic bond0
              valid_lft 3367sec preferred_lft 3367sec
        inet6 fe80::5054:ff:feb6:dce/64 scope link
              valid_lft forever preferred_lft forever
```

Чтобы показать состояние интерфейса **bond0** и его подчиненных с точки зрения канального уровня, введите команду **cat /proc/net/bonding/bond0**. Выходные данные показаны на **рисунке 12-9** и указывают, что оба подчиненных интерфейса работают, причем интерфейс **eth0** является активным **slave (ведомым)**.

РИСУНОК 12-9 Отображение статуса интерфейса bond0

УПРАЖНЕНИЕ 12-4

Тест сбоя соединения

В этом упражнении вы протестируете восстановление соединения. Мы предполагаем, что вы настроили интерфейс связывания активной резервной копии с двумя подчиненными устройствами, как описано в предыдущем разделе.

1. Запустите непрерывную команду **ping** с вашего физического хоста на **server1.example.com**
   чтобы убедиться, что **IP-соединение** работает:

   # ping 192.168.122.50

2. Завершите работу активного интерфейса на сервере **server1** с помощью команды **ifdown eth0**. Server 1 все еще отвечает на запросы **ping**?
3. Подтвердите статус активного подчиненного интерфейса с помощью следующей команды:

   # cat /proc/net/bonding/bond0

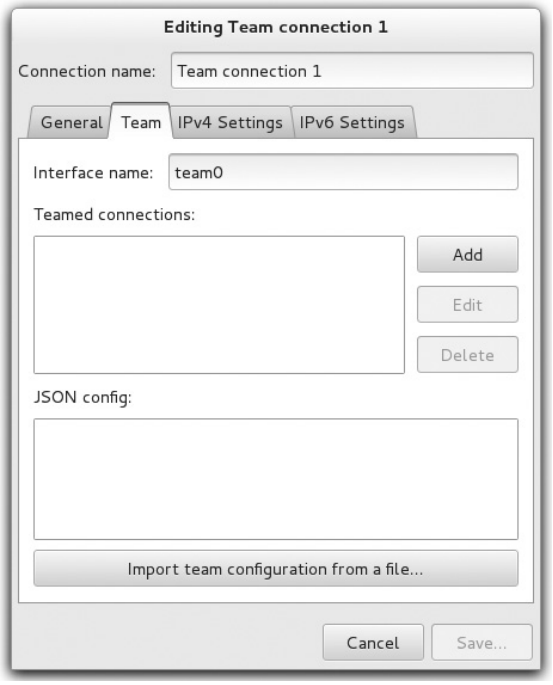
4. Включите интерфейс **eth0** с помощью команды **ifup eth0**. **Server 1** все еще отвечает на запросы **ping**? Какой активный интерфейс главного интерфейса связи?

5. Повторите **шаги 2–4 для интерфейса eth1**. Пока у вас активен один подчиненный интерфейс, IP-соединение всегда должно быть работоспособным.
6. Отключите оба интерфейса **eth0 и eth1**. Что происходит?

Настройка Объединения (teaming) Интерфейсов

Объединение в сеть - это новый метод агрегирования каналов, доступный в RHEL 7. Функционально он похож на соединение (**bond**) интерфейсов. Однако его архитектура существенно отличается. Принимая во внимание, что склеивание (**bonding**) реализован в ядре Linux, объединение (**teaming**) интерфейсов опирается на очень маленький **драйвер ядра**. Все остальная часть кода выполняется в пользовательском пространстве как часть демона пользовательского сервиса (**teamd**). Такой подход гарантирует более модульный и расширяемое проектирование, которое облегчает внедрение новых функций.

Чтобы создать новый **team** интерфейс, запустите редактор **Network Manager Connection**, нажмите кнопку «**Добавить**» и выберите «**Team**» в качестве типа подключения. После того, как вы нажмете кнопку «**Создать**», появится окно, подобное показанному ниже.



С этой точки зрения конфигурация для основных аспектов аналогична конфигурации интерфейса соединения. Таким образом, вы можете обратиться к предыдущему разделу для деталей.

После настройки нового командного интерфейса вы можете подтвердить его статус с помощью следующей команды:

```
# teamdctl team0 state
setup:
        runner: roundrobin
ports:
        eth0
                link watches:
                        link summary: up
                        instance[link_watch_0]:
                                name: ethtool
                                link: up
        eth1
                link watches:
```

link summary: up
instance[link_watch_0]:
name: ethtool
link: up

ЦЕЛЬ СЕРТИФИКАЦИИ 12.07

Аутентификация с помощью Kerberos

Две системы, настроенные с помощью **Kerberos** и прошедшие проверку подлинности, могут взаимодействовать в зашифрованном формате с симметричным ключом. Этот ключ предоставляется **Центром распространения ключей** (**KDC**). Хотя нет цели RHCE, связанной с конфигурацией **Kerberos KDC**, вам нужен **KDC**, чтобы попрактиковаться с конфигурациями, описанными в **этом разделе и в главе 16**. В следующих разделах мы начнем с основ **Kerberos**, а затем попрактикуемся с Конфигурация KDC и простого клиента.

Учебник для начинающих - Kerberos

Kerberos - это протокол сетевой аутентификации, первоначально разработанный в Массачусетском технологическом институте (**MIT**), который поддерживает безопасную идентификацию сетевых систем. RHEL 7 включает в себя клиент **Kerberos 5** и пакеты программного обеспечения, разработанные MIT.

Kerberos - это не служба каталогов, как **LDAP**. Другими словами, для правильной аутентификации клиента на сервере **Kerberos** ему также потребуется подключение к базе данных сетевой аутентификации, такой как **LDAP, NIS**, или базе данных пользователей в файле **/etc/passwd**. Службы каталогов содержат идентификаторы пользователей и групп, домашние каталоги пользователей и информацию оболочки по умолчанию. **Kerberos предназначен не для хранения этой информации, а для предоставления услуг аутентификации.**

Каждый участник **сети Kerberos** (также известный как область) идентифицируется принципалом. Участник для пользователя имеет форму **username/instance @ REALM**. Часть экземпляра является необязательной и обычно определяет тип пользователя. Область указывает область действия домена **Kerberos** и обычно указывается заглавной версией имени домена **DNS**. Например, областью **Kerberos** для домена **DNS example.com** обычно является **EXAMPLE.COM**.

Предварительные условия для серверов и клиентов Kerberos

Kerberos полагается на точные метки времени. Если время на серверах и клиентах больше пяти минут, это приведет к сбоям аутентификации. Чтобы избежать этой проблемы, в производственной сети обычно все хосты синхронизируют свое время через **NTP** (сетевой протокол времени).

Kerberos также использует сервис разрешения имен. Вы можете заставить его работать либо с **локальным DNS-сервером**, либо с полным файлом **/etc/hosts** на каждом хосте вашей сети.

Для этой книги мы создали физическую рабочую станцию с именем **maui.example.com**. Этот хост запускает виртуальные машины **server1.example.com, tester1.example.com** и **outsider1.example.com**. Файл **/etc/hosts** для этой лабораторной среды показан на **рисунке 12-10**.

РИСУНОК 12-10
Содержимое файла **/etc/hosts**

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.122.1   maui.example.com maui
192.168.122.50  server1.example.com server1
192.168.122.150 tester1.example.com tester1
192.168.100.100 outsider1.example.com outsider1
```

УПРАЖНЕНИЕ 12-5

Установите Kerberos KDC

В этом руководстве мы покажем, как настроить центр распространения ключей. Хотя это не требование **RHCE**, вам нужно **KDC** для практики с **Упражнением 12-6** и лабораторными работами в конце главы. Вы можете установить **KDC** либо на рабочей станции, на которой запущены виртуальные машины, развернутые в главе 1, либо на выделенной виртуальной машине.

1. Установите **RPM-пакеты** для **krb5-server и krb5-workstation**:

   # yum install -y krb5-server krb5-workstation

ЯЯ

2. Отредактируйте файл **/etc/krb5.conf**. Раскомментируйте строку **default_realm=EXAMPLE.COM** и четыре строки в разделе [**realms**]. Замените значения по умолчанию для **kdc** и **admin_server** на полное доменное имя вашего сервера (в нашем случае **maui.example.com**). Результат показан здесь

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 dns_lookup_realm = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true
 rdns = false
 default_realm = EXAMPLE.COM
 default_ccache_name = KEYRING:persistent:%{uid}

[realms]
 EXAMPLE.COM = {
  kdc = maui.example.com
  admin_server = maui.example.com
 }

[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

3. Просмотрите содержимое файла **/var/kerberos/krb5kdc/kdc.conf**. По умолчанию этот файл настроен для **области Kerberos EXAMPLE.COM**, как показано ниже. Вам не нужно изменять этот файл, если только вы не хотите настроить имя области **Kerberos**, отличное от имени по умолчанию.

```
[kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88

[realms]
 EXAMPLE.COM = {
  #master_key_type = aes256-cts
  acl_file = /var/kerberos/krb5kdc/kadm5.acl
  dict_file = /usr/share/dict/words
  admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
  supported_enctypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal
 arcfour-hmac:normal camellia256-cts:normal camellia128-cts:normal des-hmac-sha1
:normal des-cbc-md5:normal des-cbc-crc:normal
 }
```

4. Создайте новую базу данных **Kerberos**, выполнив следующую команду. Вам будет предложено ввести главный **ключ (пароль),** который **KDC** использует для шифрования базы данных:

   # kdb5_util create -s
   Loading random data
   Initializing database '/var/kerberos/krb5kdc/principal' for realm □
   'EXAMPLE.COM',
   master key name 'K/M@EXAMPLE.COM'
   You will be prompted for the database Master Password.
   It is important that you NOT FORGET this password.
   Enter KDC database master key:
   Re-enter KDC database master key to verify:

   Опция **-s** сохраняет главный ключ в файле-хранилище, так что вам не нужно вводить главный ключ вручную при каждом запуске службы **Kerberos**.
5. Запустите и включите службы **Kerberos** для запуска при загрузке:

   # systemctl start krb5kdc kadmin
   # systemctl enable krb5kdc kadmin

6. Разрешите подключения к серверу **Kerberos** через зону по умолчанию на брандмауэре:

   # firewall-cmd --permanent --add-service=kerberos
   # firewall-cmd –reload

7. Запустите команду **kadmin.local** для администрирования **KDC** и создания, просмотра или удаления принципалов, как показано в следующем примере:

   # kadmin.local
   Authenticating as principal root/admin@EXAMPLE.COM with password
   kadmin.local: listprincs
   K/M@EXAMPLE.COM
   kadmin/admin@EXAMPLE.COM
   kadmin/changepw@EXAMPLE.COM
   kadmin/maui.example.com@EXAMPLE.COM
   krbtgt/EXAMPLE.COM@EXAMPLE.COM
   kadmin.local: addprinc mike
   Enter password for principal "mike@EXAMPLE.COM":
   Re-enter password for principal "mike@EXAMPLE.COM":
   Principal "mike@EXAMPLE.COM" created.
   kadmin.local: addprinc alex
   Enter password for principal "alex@EXAMPLE.COM":
   Re-enter password for principal "alex@EXAMPLE.COM":

Principal "alex@EXAMPLE.COM" created.
kadmin.local: delprinc alex
Are you sure you want to delete the principal "alex@EXAMPLE.COM"? □
(yes/no): yes
Principal "alex@EXAMPLE.COM" deleted.
Make sure that you have removed this principal from all ACLs before □
reusing.
kadmin.local

Установите клиент Kerberos

Для целей экзамена, а также на работе почти всегда лучше, чтобы решения были максимально простыми. Вот где может помочь инструмент настройки аутентификации. Чтобы увидеть, что делает этот инструмент для настройки клиента **Kerberos**, вы можете создать резервную копию файлов в каталоге **/etc/sssd** вместе с файлом конфигурации **/etc/nsswitch.conf**. Этот файл относится к демону **System Security Services**.

Инструмент настройки графической аутентификации

Один из способов открыть версию **GUI** инструмента настройки аутентификации - с помощью команды **authconfig-gtk**. Это должно открыть инструмент настройки аутентификации с двумя вкладками, показанными на **рисунке 12-11**. Хотя другие базы данных аутентификации поддерживаются, основное внимание уделяется **LDAP**. Параметры в разделе **LDAP** на вкладке «Идентификация и аутентификация» обсуждались в **главе 8**.

РИСУНОК 12-11
Настройте клиент на основе Kerberos с помощью графического инструмента настройки аутентификации.

Основное внимание в этом разделе уделяется второй половине вкладки. Для клиента на основе **Kerberos** вы сохраните пароль **Kerberos** в качестве параметра «**Метод аутентификации**». Вот другие варианты:

- **Realm (Область)** По соглашению область **Kerberos** совпадает с именем домена для сети прописными буквами.
- **KDC KDC** является центром распространения ключей **Kerberos**. Запись здесь должна соответствовать либо полному доменному имени (**FQDN**), либо **IP-адресу** фактического сервера **Kerberos**.
- **Admin Servers(Серверы администрирования)** Сервер администрирования, связанный с **KDC**, часто находится в одной и той же системе. На административном сервере **Kerberos** работает демон **kadmind**.
- Использование **DNS** для преобразования хостов в области. Если для локальной сети существует доверенный **DNS-сервер**, вы можете разрешить локальной системе использовать **DNS-сервер** для поиска области. Если эта опция активирована, текстовое поле **Realm** будет отключено.
- Использование **DNS** для поиска **KDC** для областей. Если для локальной сети существует доверенный **DNS-сервер**, вы можете разрешить локальной системе использовать **DNS-сервер** для поиска **KDC** и административного сервера. Если эта опция активирована, текстовые поля **KDC** и **Admin Server** будут отключены.

Для целей этого раздела примите параметры по умолчанию, как показано на **рисунке 12-11**. Нажмите Применить. Через несколько секунд окно конфигурации аутентификации закроется, и в файлы конфигурации будут внесены изменения.

Консольный инструмент настройки аутентификации

Чтобы запустить текстовую версию средства настройки аутентификации, выполните команду **authconfig-tui**. Как показано на **рисунке 12-12**, вам не нужно активировать **LDAP**, по крайней мере, для аутентификации.

После того, как вы выбрали **Next**, инструмент отобразит экран **Kerberos Settings**, показанный на **Рисунке 12-13**. Параметры по умолчанию, показанные здесь, те же, что показаны в графической версии инструмента на **рисунке 12-11**.

Вам также может потребоваться настроить изменения в файлах конфигурации, как описано далее.

РИСУНОК 12-12 Настройте клиент на основе Kerberos с помощью инструмента настройки аутентификации консоли.

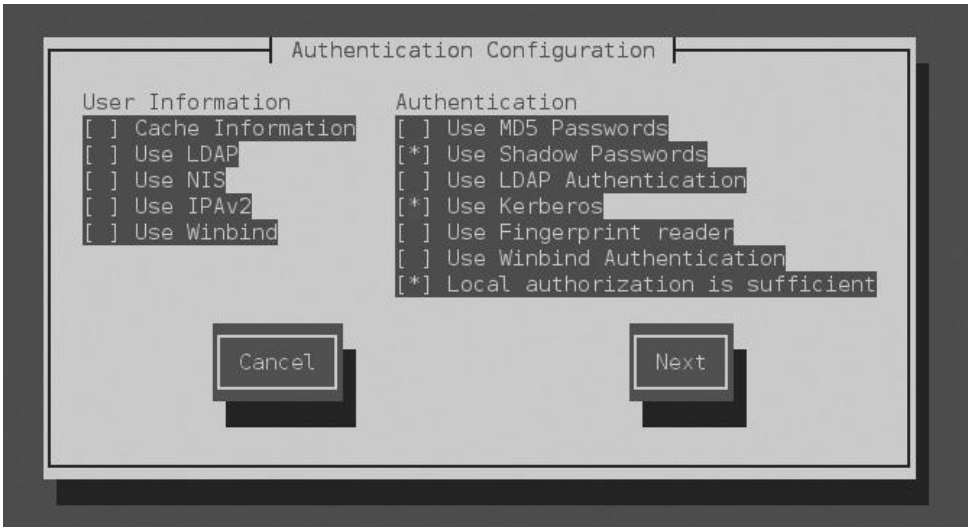
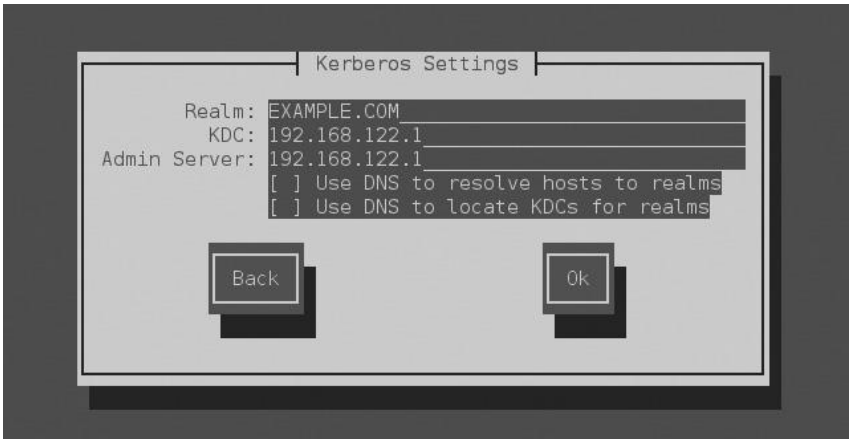


РИСУНОК 12-13 Укажите настройки клиента Kerberos.



УПРАЖНЕНИЕ 12-6

Настройте аутентификацию Kerberos

В этом упражнении вы настроите пользователя с соответствующим принципалом **Kerberos** для аутентификации. Мы предполагаем, что в вашей физической системе установлен KDC, прослушивающий IP-адрес 192.168.122.1, и вы хотите настроить пользователя на виртуальной машине server1.example.com для аутентификации на KDC. Следуй этим шагам:

1. Установите **RPM-пакеты krb5-workstation** и **pam_krb5** на клиенте **Kerberos server1.example.com**:

   # yum -y install krb5-workstation pam_krb5

2. Добавьте нового пользователя на **server1.example.com** для проверки аутентификации **Kerberos**. Например:

   # useradd mike

3. Из терминала **GNOME** запустите команду **authconfig-tui** и настройте **server1.example.com** для использования **Kerberos** для аутентификации, как показано ранее на **рисунках 12-11 и 12-12**. В качестве альтернативы вы можете выполнить следующую команду:

   **# authconfig --update --enablekrb5 --krb5kdc=192.168.122.1 **
   > --krb5adminserver=192.168.122.1 --krb5realm=EXAMPLE.COM

4. На **KDC** запустите **kadmin.local** и добавьте принципала для пользователя **mike**:

 # kadmin.local

Authenticating as principal root/admin@EXAMPLE.COM with password
kadmin.local: addprinc mike
Enter password for principal "mike@EXAMPLE.COM":
Re-enter password for principal "mike@EXAMPLE.COM":
Principal "mike@EXAMPLE.COM" created.

5. Проверьте аутентификацию, выполнив вход на **server1** как **mike** через SSH.

6. В случае успеха команда **klist** покажет **TGT** для пользователя **mike**:

[mike@server1 ~]$ klist
Ticket cache: KEYRING:persistent:1001:krb_ccache_0YxfosR
Default principal: mike@EXAMPLE.COM

Valid starting	**Expires**	**Service principal**
12/08/15 17:42:53	**13/08/15 17:42:53**	**krbtgt/EXAMPLE.COM@EXAMPLE.COM**

СЦЕНАРИЙ И РЕШЕНИЕ	
Вам необходимо настроить отчет об использовании системы для различных системных ресурсов.	Начните со страницы **man** для команды **sadf**; используйте параметры, связанные с командой **sar** для нужных ресурсов.
Вам сказали настроить пересылку **IPv6** в системе.	Включите параметр **net.ipv6.conf.all.forwarding=1** в **/etc/sysctl.conf** и Активируйте его с помощью команды **sysctl -p**.
Вам необходимо установить специальный статический маршрут через устройство **eth1**.	Используйте инструмент редактора сетевых подключений, чтобы настроить этот специальный маршрут, учитывая **сетевой адрес, маску подсети и желаемый IP-адрес шлюза**.
Вам нужна избыточность сети в вашей системе.	Добавьте второй сетевой интерфейс. Объедините два интерфейса, используя **bond** или **team**.
Вам необходимо настроить систему в качестве клиента **Kerberos**.	Используйте инструмент настройки аутентификации **GUI**; область должна быть прописной буквой для домена. Вам также понадобится полное доменное имя для серверов администрирования **KDC** и **Kerberos** (которые могут совпадать).

РЕЗЮМЕ СЕРТИФИКАЦИИ

Администраторы **Linux** должны регулярно настраивать скрипты. Примеры сценариев уже доступны в разных каталогах **/etc/cron.***. Обычно скрипты **bash** начинаются со строки **#!/bin/sh**, которая устанавливает интерпретатор. Административные сценарии могут использовать команды **Linux** вместе с внутренними командами **bash**, такими как **for**, **if**, **do** и **test**.

В **RHCE**, вы должны иметь возможность контролировать производительность администрируемых систем. Это провинция службы **sysstat**. Хотя такие команды, как **df, top** и **dstat**, могут отображать данные об использовании **ЦП, ОЗУ, диска и сети**, реальные отчеты можно подготовить с помощь команды **sadf**. Пример того, как утилита собирает статистику **ОЗУ и сетевые данные**, доступен на справочной странице **sadf**; Затем вы можете добавить данные об использовании процессора и диска из соответствующих командных ключей **sar**.

Параметры времени выполнения ядра можно найти в каталоге **/proc/sys**, но изменения в таких файлах являются временными. Для более постоянных изменений вы должны настроить параметры в файле **/etc/sysctl.conf**. Изменения в этом файле могут быть реализованы с помощью команды **sysctl -p**. Многие стандартные параметры ядра относятся к сети.

Задачи **RHCE** включают требования для нескольких специальных конфигураций сети. С помощью редактора сетевых подключений статические IP-маршруты можно настроить в файле

в каталоге **/etc/sysconfig/network-scripts**. Используя тот же инструмент, вы также можете настроить **адреса IPv6**, а также интерфейсы **bond** и **team**.

Клиенты **Kerberos** можно настроить с помощью команды **authconfig-gtk**. Чтобы попрактиковаться с **Kerberos**, вам необходимо настроить Центр распространения ключей (**Key Distribution Center KDC**), как описано в этой главе.

Пару минут проверки

Вот некоторые из ключевых моментов целей сертификации в главе 12.

Автоматизировать обслуживание системы

- Стандартные административные сценарии могут предоставить модель для пользовательских сценариев для автоматизации задач обслуживания системы.
- Различные команды в скриптах включают **do, for, if** и **test**.
- Скрипты **Bash** начинаются со строки **#!/bin/sh** или **#!/bin/bash**.

Настройка отчетов об использовании системы

- Несколько команд использования системы доступны в RHEL 7 с помощью пакета **sysstat**.
- Команда **sa1** регулярно собирает данные в каталоге **/var/log/sa**.
- Отчеты о состоянии системы можно создавать с помощью команды **sadf** с помощью переключателей(опций) команды **sar**.
- Один пример команды отчета о состоянии системы показан на справочной странице **sadf**.

Параметры времени выполнения ядра

- Параметры времени выполнения ядра находятся в каталоге **/proc/sys**.
- Многие параметры времени выполнения ядра относятся к параметрам сети, таким как **переадресация IP и безопасность.**
- Параметры времени выполнения ядра можно настраивать на постоянной основе с помощью файла **/etc/sysctl.conf**.

IP-маршруты

- Конфигурация маршрута по умолчанию требует IP-адреса шлюза.
- **Статические маршруты** в разные сети можно настроить с помощью инструмента «**Редактор сетевых подключений(Network Connections Editor**)» и его текстового аналога nmtui.

Введение в IPv6

- Адреса **IPv6 имеют 128 бит**, организованных в полубайтах по 16 бит.
- Три разных типа а**дресов IPv6 - одноадресная, произвольная и многоадресная.**
- Адреса **IPv6** могут быть ограничены сегментами локальной сети (link-local) или маршрутизируемыми.

Связывание и объединение сетевых интерфейсов

- Объединение(**bonding**) в сеть и **teaming** в сеть обеспечивают избыточность канала и, возможно, более высокую пропускную способность сети через различные режимы конфигурации, такие как циклический перебор и активное резервное копирование.

Аутентификация с помощью Kerberos

- Для аутентификации в **Kerberos** необходим Центр распространения ключей (**KDC**).
- Чтобы настроить клиент Kerberos, вы можете использовать команду authconfig-gtk.

САМОПРОВЕРКА

Следующие вопросы помогут оценить ваше понимание материалов, представленных в этой главе. Поскольку на экзаменах Red Hat не появятся вопросы с несколькими вариантами ответов, в этой книге нет вопросов с несколькими вариантами ответов. Эти вопросы исключительно проверяют ваше понимание главы. Это нормально, если у вас есть другой способ выполнить задачу. Получение результатов, а не запоминание пустяков, это то, что рассчитывает на экзамены Red Hat.

Автоматизировать обслуживание системы

1. Какой код выхода связан с успехом в скрипте?

_____

2. Напишите команду тестирования **bash**, чтобы проверить, существует ли файл и является ли он исполняемым.

_____

3. Напишите **bash** для оператора, который будет переключаться между всеми пользователями в системе.

_____

Настройка отчетов об использовании системы

4. Какой каталог включает в себя задание **cron**, которое регистрирует активность системы? Предположим, что установлен соответствующий пакет.

_____

5. Где в системе RHEL 7 найти пример команды для создания отчета об использовании системы?
Где можно найти дополнительные параметры для этого отчета?

_____

Временные параметры выполнения ядра

6. Какой полный путь к файлу **/proc**, связанному с параметром **net.ipv4.ip_forward?**

_____

IP-маршруты

7. Какие параметры конфигурации связаны со статическим маршрутом?

_____

Введение в IPv6

8. Какое самое короткое представление адреса **2001:0db8:00aa:0000:04ba:0000:0000:00cd IPv6**?

_____

9. Какую команду вы можете использовать для проверки адреса **IPv6**?

_____

Связывание и объединение сетевых интерфейсов

10. Какую команду вы можете запустить, чтобы проверить состояние интерфейса **bond0** и его подчиненных интерфейсов?

_____

Аутентификация с помощью Kerberos

11. Какова стандартная область **Kerberos** для системы **server1.example.com**?

_____

12. Какую команду вы запускаете, чтобы вывести список билетов **Kerberos** для текущего пользователя?

_____

ВОПРОСЫ ЛАБОРАТОРНОЙ РАБОТЫ

Некоторые из этих лабораторий включают в себя упражнения по настройке. Вы должны делать эти упражнения только на тестовых машинах. Предполагается, что вы выполняете эти упражнения на виртуальных машинах, таких как **KVM**.

Red Hat представляет свои экзамены в электронном виде. По этой причине лаборатории для этой главы доступны от носитель, сопровождающий книгу в подкаталоге **Chapter12/.** Если вы еще не настроили **RHEL 7** в системе, обратитесь к **Главе 1** за инструкциями по установке.

Ответы для лабораторных работ следуют за ответами самопроверки для вопросов, которые нужно заполнить.

Лабораторная работа

Во время экзаменов **Red Hat** задания будут представлены в электронном виде. Таким образом, эта книга также представляет большинство лабораторий в электронном виде. Для получения дополнительной информации см. Раздел «Лабораторные вопросы» в конце **главы 12**. Большинство лабораторных работ для этой главы просты и требуют очень небольшого количества команд или изменений в одном или двух файлах конфигурации.

Лаборатория 1

В этой лабораторной работе вы создадите скрипт с именем **backup.sh**, который принимает два аргумента. Сценарий должен выполнить резервное копирование всех файлов из каталога, указанного в качестве первого аргумента в несжатом архивном файле **.tar**, расположенном в каталоге, переданном в качестве второго аргумента. Файл назначения должен называться **backup-MMDDHHSS.tar**, где **MMDDHHSS** - текущий день, выраженный в виде двухзначных блоков для текущего месяца, дня, часа и секунды.

Если скрипт запускается с другим количеством аргументов, он должен отобразить следующее сообщение об ошибке и выйти с **кодом 1**:

Usage: backup.sh <source> <destination>

Если первый аргумент не является обычным каталогом, сценарий должен отобразить ошибку, похожую на следующую, и завершиться **с кодом 2**:

Error: directory [substitute with first argument] does not exist

Если каталог, указанный в качестве второго аргумента, не существует, сценарий должен его создать.

Лаборатория 2

В этой лабораторной работе вы настроите отчет об использовании системы за один день (например, за последний 21-й месяц месяца), записанный в файл **sysstat_report.txt**. Параметры отчета могут быть ограничены памятью и статистикой сети.

Лаборатория 3

В этой лабораторной работе вы настроите отчет об использовании системы за один день, записанный в файл **morestat_report.txt**. Параметры отчета должны включать информацию об **использовании ЦП, статистике ОЗУ, использовании диска и сетевых данных**. Отчет должен быть представлен в более наглядном формате, который может быть легко использован утилитой команды **awk**.

Лаборатория 4

В этой лабораторной работе вы отключите ответы на команду **ping**, используя настройки ядра. (На экзамене также было бы приемлемо отключить ответы на команду **ping** с помощью инструмента настройки брандмауэра, но, когда это возможно, полезно знать более одного метода.)

Лаборатория 5

В системе **server1.example.com** настройте пользовательский маршрут к сети с помощью системы **outsider1.example.org**. Используйте тот же адрес шлюза, что и шлюз по умолчанию.

Лаборатория 6

В этой лабораторной работе вы настроите **IPv6-адреса** на виртуальных машинах **server1.example.com и tester1.example.com**, а также в своей физической системе, используя параметры в следующей таблице:

System	Interface	IPv6 Address
Physical Host	virbr0	2001:db8:7a::1/64
server1.example.com	eth0	2001:db8:7a::50/64
tester1.example.com	eth0	2001:db8:7a::150/64

Лаборатория 7

В этой лабораторной работе вы добавите второй сетевой адаптер на **tester1.example.com**, используя тип **virtio**. Затем вы объедините **eth0** и **eth1** в объединенном интерфейсе с именем **team0** с IP-адресом **192.168.122.150/24** и шлюзом по умолчанию **192.168.122.1**.

ОТВЕТЫ НА САМОПРОВЕРКУ

Автоматизировать обслуживание системы

1. Код завершения, связанный с успехом в скрипте, равен 0.
2. Команда **bash test** для проверки того, существует ли файл и является ли он исполняемым, может быть записана следующим образом:

   test -x /path/to/file

3. Оператор **for** для циклического перебора всех имен пользователей в системе может быть записан следующим образом:

   for username in $(getent passwd | cut -f 1 -d ":")

Настройка отчетов об использовании системы

4. Каталог со стандартным заданием **sysstat - /etc/cron.d**.
5. В системе **RHEL 7** одно место, где вы можете найти пример команды использования системы отчет - страница руководства **sadf man**. Дополнительные ключи можно найти на странице руководства **sar**.

Временные параметры выполнения ядра

6. Полный путь к файлу, связанному с параметром **net.ipv4.ip_forward**: **/proc/sys/net/ipv4/ip_forward**.

IP-маршруты

7. Конфигурационными параметрами, связанными со статическим маршрутом, являются сетевой адрес, маска подсети и адрес шлюза.

Введение в IPv6

8. Самое короткое представление адреса **2001:0db8:00aa:0000:04ba:0000:0000:00cd IPv6 2001:db8:aa:0:4ba::cd**.

9. Вы можете использовать команду **ping6** для проверки связи с **IPv6-адресом**. Если это локальный адрес ссылки, вам нужно указать исходящий интерфейс с **ключом -I.**

Связывание и объединение сетевых интерфейсов

10. Чтобы проверить состояние интерфейса **bond0** и его подчиненных интерфейсов, выполните следующую команду:

    # cat /proc/net/bonding/bond0

Аутентификация с помощью Kerberos

11. Стандартной областью **Kerberos** для системы **server1.example.com** является **EXAMPLE.COM**.
12. Команда, которая перечисляет билеты **Kerberos** для текущего пользователя, является **klist**.

ОТВЕТЫ ЛАБОРАТОРНОЙ РАБОТЫ

Успех в этой лаборатории должен быть простым. Самый простой способ настроить скрипт - это начать с основных требований, а затем добавить другие функции. Например, следующий скрипт сохраняет текущую дату в формате **MMDDHHSS** в переменной **$TODAY.** Затем он запускает команду **tar** создать резервную копию каталога, переданного в качестве первого аргумента в файл **backup-MMDDHHSS.tar** внутри каталога, заданного в качестве второго аргумента:

```
#!/bin/bash
TODAY=$(date +%m%d%H%S)
tar cf "$2/backup-$TODAY.tar" "$1"
```

Следующим шагом является добавление других неосновных функций. Вам понадобится тест, чтобы проверить, количество аргументов не равно двум:

```
if [ $# -ne 2 ]; then
        echo "Usage: backup.sh <source> <destination>"
        exit 1
fi
```

Вам также необходимо добавить еще один тест, чтобы подтвердить, что аргументы, передаваемые в сценарий, являются регулярными каталогом:

```
if [ ! -d "$1" ]; then
        echo "Error: directory $1 does not exist"
        exit 2
fi
```

Кроме того, требуется другой тест, чтобы проверить, является ли второй аргумент каталогом. Если тест не пройден, скрипт должен создать каталог:

```
if [ ! -d "$2" ]; then
        mkdir -p "$2"
fi
```

Обратите внимание, что если вторым аргументом является файл, а не каталог, скрипт вернет ошибку. Однако это не условие ошибки, которое упражнение просит вас принять во внимание.

Если вы соберете все блоки кода, у вас будет рабочий скрипт. Протестируйте скрипт с разными аргументы для проверки того, что все условия исключения распознаны и успешно обработаны.

Лаборатория 2

Если вы поняли требования этой лаборатории, ответ должен быть простым. Пока есть другие методы, одна соответствующая команда, которая отвечает данным требованиям, доступна на странице руководства для команда **sadf**:

```
# sadf -d /var/log/sa/sa21 -- -r -n DEV
```

Конечно, чтобы получить эту информацию в указанном файле, вывод должен быть перенаправлен:

 # sadf -d /var/log/sa/sa21 -- -r -n DEV > sysstat_report.txt

Лаборатория 3

Эта лабораторная работа основана на том, что вы делали в лабораторной **работе 2**. Если вы не запомнили дополнительные параметры команды которые определяют информацию об использовании процессора и диска, вы можете найти эти опции на странице **man** для **sar** команда. Как предлагается на странице **man, ключ -u** может использоваться для отчета об использовании процессора, тогда как **-d ключ** сообщает о работе блочного устройства. Это может помочь пользователям прочитать вывод, если **ключ -p** объединен с **-d**.

Но есть еще одно требование: параметр **-p рядом с командой sadf** приводит к выводу в формате может использоваться утилитой команды **awk**. Ниже приведен один из методов, удовлетворяющих требованиям лаборатории:

 # sadf -p /var/log/sa/sa21 -- -u -r -dp -n DEV > morestat_report.txt

Лаборатория 4

Если вы успешно завершили эту лабораторную работу, файл **/etc/sysctl.conf** (или файл в каталоге **/etc/sysctl.d**) теперь должна иметь следующую запись:

net.ipv4.icmp_echo_ignore_all = 1

Это просто гарантирует, что новый параметр переживет перезагрузку. Вы также можете установить связанный файл, **/proc/sys/net/ipv4/icmp_echo_ignore_all, равным 1**, или выполните команду **sysctl -p** для реализации изменения до перезагрузки системы.

Конечно, успех можно подтвердить с помощью команды **ping** как из локальной, так и из удаленной систем. Если вы хотите восстановить исходную конфигурацию, вернитесь в систему **server1.example.com**, а затем удалите параметр **net.ipv4.icmp_echo_ignore_all** из файла **/etc/sysctl.conf**.

Лаборатория 5

Если вы использовали инструмент «**Сетевые подключения**» для настройки специального маршрута, он должен настроить новый файл в каталог **/etc/sysconfig/network-scripts**. Если указан сетевой адаптер **eth0**, этот специальный файл будет **маршрут-eth0**. Учитывая параметры, используемые для сети **outsider1.example.org**, как обсуждалось в Глава 1, этот файл будет содержать следующие три строки:

ADDRESS0 = 192.168.100.0
NETMASK0 = 255.255.255.0
GATEWAY0 = 192.168.122.1

Конечно, если система **outsider1.example.org** находится в другой сети, содержимое **маршрут-eth0** файл изменится соответственно.

Лаборатория 6

Для выполнения этой лабораторной работы используйте редактор подключений Network Manager, добавьте адреса IPv6 указаного на интерфейсах. **Префикс сети - /64**. Вам не нужно устанавливать шлюз по умолчанию IPv6 потому что все указанные адреса **IPv6** находятся в одной подсети.

Затем проверьте соединение между хостами с помощью команды **ping6**. Например, запустите следующее команды от **server1 и tester1** для проверки связи с физическим хостом:

 # ping6 2001:db8:7a::1

Лаборатория 7

Запустите эту лабораторию, выключив **tester1**. Добавьте новый сетевой адаптер, используя модель устройства **virtio**, и затем включите машину. Вы можете подтвердить, что новый адаптер доступен в системе с помощью Команда **ip link show**.

Используйте редактор редактора **Connection Network Manager** для создания адаптера **team0**. Перед созданием нового интерфейса, убедитесь, что существующая конфигурация на **eth0** удалена.

Успех в этой лаборатории означает следующее:

У вас есть полное сетевое подключение, что демонстрируется выполнением команды ping для проверки что другие хосты достижимы.

Интерфейс **team0** запущен и объединяет **eth0** и **eth1**. Вы можете проверить это, запустив команда состояния **teamdctl team0.**

Если вы отключите интерфейс **eth0** или **eth1** с помощью команды **ifdown**, система все еще будет иметь сеть подключение.