Глава 9

RHCSA Система уровня

Административные задачи

ЦЕЛИ СЕРТИФИКАЦИИ

9.01 Элементарные команды системного администрирования
9.02 Автоматизация системного администрирования:
cron и at
9.03 Анализ локальных файлов журнала
✓ Двухминутная тренировка
Q & A Самопроверка

Эта последняя глава RHCSA охватывает задачи по администрированию функциональных систем, которые еще не рассмотрены в других главах. Он начинается с обсуждения управления процессами и продолжается использованием архивных файлов. Кроме того, эта глава поможет вам автоматизировать повторяющиеся задачи системного администрирования. Некоторые из этих задач случаются, когда вы хотите иметь «жизнь», другие, когда вы предпочитаете спать. В этой главе вы узнаете, как планировать как разовое, так и периодическое выполнение заданий. Это стало возможным с помощью **cron** и **at** демонов. В этом случае «**at**» не является предлог, но служба, которая контролирует систему для одноразовых запланированных заданий. Аналогичным образом, **cron** - это сервис, который контролирует систему для регулярных запланированных заданий.

Когда вы устраняете неполадки, регистрация в системе часто дает подсказки, необходимые для решения многих проблем. В этой главе основное внимание уделяется локальной регистрации.

ВНУТРИ ЭКЗАМЕНА

Администрирование системы

Администраторы работают в системах Linux несколькими способами. В этой главе вы изучите различные методы для достижения следующих целей RHCSA. Первая из этих задач включает в себя базовые навыки работы с командами:

- Архивирование, сжатие, распаковка и распаковка файлов с использованием tar, star, gzip и bzip2.

Эти другие задачи более тесно связаны с системным администрированием:

- Идентификация процессов, интенсивно использующих **процессор/память**, отрегулируйте приоритет процесса с помощью **renice** и **kill** процессов
- Планируйте задачи, используя **at** и **cron**

В конце, вы узнаете, где найти информацию, занесенную в журнал **systemd** и **rsyslog**. Соответствующей целью **RHCSA** является

- Поиск и интерпретация системных журналов и журналов.

ЦЕЛЬ СЕРТИФИКАЦИИ 9.01

Элементарные команды системного администрирования

Несколько команд системного администрирования в задачах RHCSA не рассматриваются в предыдущих главах. Они связаны с управлением системными ресурсами и архивами. Команды управления системными ресурсами позволяют увидеть, какие процессы запущены, проверить ресурсы, которые они используют, а также завершить или перезапустить эти процессы. Команды архивирования поддерживают объединение группы файлов в один архив, который затем можно сжать.

Команды управления системными ресурсами

Linux включает в себя множество команд, которые могут помочь вам идентифицировать те процессы, которые монополизируют систему. Наиболее простой из этих команд является команда **ps**, которая предоставляет снимок запущенных в данный момент процессов. Эти процессы могут быть ранжированы с помощью команды **top**, которая может отображать запущенные задачи Linux в порядке использования их ресурсов. С помощью **top** вы можете идентифицировать те процессы, которые используют больше всего ресурсов **ЦП** и **ОЗУ**.

Команды, которые могут регулировать приоритет процесса, включают **nice** и **renice**. Иногда недостаточно отрегулировать приоритет процесса, после чего может оказаться целесообразным отправить сигнал процессу с помощью таких команд, как **kill** и **killall**. Если вам нужно отслеживать использование системы, также могут быть полезны команды **sar** и **iostat**.

!!!!!
Задача, связанная с управлением системными ресурсами, состоит в том, чтобы «идентифицировать процессы, интенсивно использующие ЦП / память, корректировать приоритет процесса с помощью renice и kill процессы».
!!!!!

Управление процессами с помощью команды ps

Важно знать, что работает на компьютере с **Linux**. Чтобы помочь с этой задачей, команда **ps** имеет несколько полезных ключей. Когда вы пытаетесь диагностировать проблему, одним из распространенных методов является начать с полного списка запущенных процессов, а затем искать конкретную программу. Например, если веб-браузер Firefox внезапно завершит работу, вы захотите уничтожить все связанные процессы. **ps aux| grep firefox** может помочь вам определить процесс (ы), который вам нужно убить.

!!!!
Команда **pgrep** также полезна, потому что она сочетает в себе функции **ps и grep**. В этом случае команда **pgrep -a firefox** функционально эквивалентна команде **ps aux | grep firefox**
!!!!

Одной команды **ps** обычно недостаточно. Все, что он делает, это идентифицирует те процессы, которые выполняются в текущем терминале. Эта команда обычно возвращает только процесс, связанный с текущей оболочкой, и сам процесс команды **ps.**

Чтобы идентифицировать эти процессы, связанные с именем пользователя, может помочь команда **ps -u username**. Иногда есть конкретные пользователи, которые могут быть проблемными по разным причинам. Поэтому, если вы подозреваете пользователя **mjang**, следующая команда может помочь вам просмотреть все процессы, связанные с этим пользователем в настоящее время:

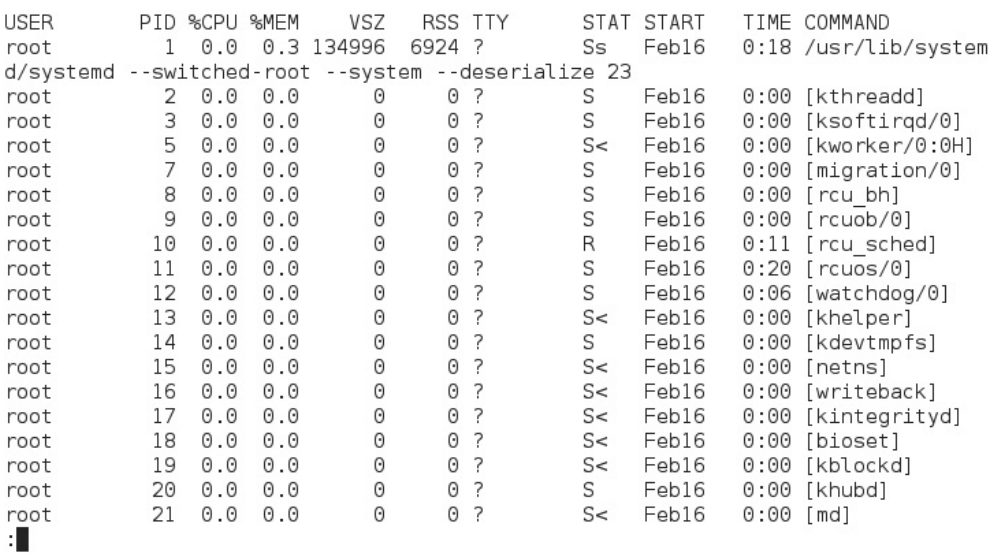
$ ps -u mjang

Как администратор, вы можете сосредоточиться на конкретной учетной записи по разным причинам, например, из-за действий, обнаруженных командой **top**, описанных в

следующем разделе. В качестве альтернативы вы можете проверить все запущенные в данный момент процессы с помощью команды, например:

$ ps aux

РИСУНОК 9-1 Вывод команды ps aux



```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3 134996  6924 ?        Ss   Feb16   0:18 /usr/lib/system
d/systemd --switched-root --system --deserialize 23
root         2  0.0  0.0      0     0 ?        S    Feb16   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Feb16   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   Feb16   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    Feb16   0:00 [migration/0]
root         8  0.0  0.0      0     0 ?        S    Feb16   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    Feb16   0:00 [rcuob/0]
root        10  0.0  0.0      0     0 ?        R    Feb16   0:11 [rcu_sched]
root        11  0.0  0.0      0     0 ?        S    Feb16   0:20 [rcuos/0]
root        12  0.0  0.0      0     0 ?        S    Feb16   0:06 [watchdog/0]
root        13  0.0  0.0      0     0 ?        S<   Feb16   0:00 [khelper]
root        14  0.0  0.0      0     0 ?        S    Feb16   0:00 [kdevtmpfs]
root        15  0.0  0.0      0     0 ?        S<   Feb16   0:00 [netns]
root        16  0.0  0.0      0     0 ?        S<   Feb16   0:00 [writeback]
root        17  0.0  0.0      0     0 ?        S<   Feb16   0:00 [kintegrityd]
root        18  0.0  0.0      0     0 ?        S<   Feb16   0:00 [bioset]
root        19  0.0  0.0      0     0 ?        S<   Feb16   0:00 [kblockd]
root        20  0.0  0.0      0     0 ?        S    Feb16   0:00 [khubd]
root        21  0.0  0.0      0     0 ?        S<   Feb16   0:00 [md]
:
```

Команда **ps aux** дает более полную базу данных запущенных в данный момент процессов, в порядке их **PID**. Опция **a** перечисляет все запущенные процессы, **u** выводит вывод в формате, ориентированном на пользователя, и **x** снимает стандартное ограничение, что перечисленные процессы должны быть связаны с терминалом или консолью. Один пример показан на **рисунке 9-1**. Хотя выходные данные могут включать в себя сотни процессов и более, выходные данные могут быть перенаправлены для дальнейшего анализа с помощью таких команд, как **grep**. Выходные столбцы, показанные на рисунке 9-1, описаны в таблице 9-1.

ТАБЛИЦА 9-1 Столбцы вывода из ps aux

Название колонки	Описание
USER	Имя пользователя, связанное с процессом.
PID	Идентификатор процесса.
%CPU	Загрузка ЦП, как процент времени, потраченного на работу в течение всего времени жизни процесса.
% MEM	Текущее использование оперативной памяти.
VSZ	Размер виртуальной памяти процесса в КиБ.
RSS	Физическая память, используемая процессом, не включая пространство подкачки, в КиБ.
TTY	Связанная терминальная консоль.
STAT	Состояние процесса.
START	Время начала процесса. Если вы просто видите дату, процесс начался более 24 часов назад.
TIME	Общее время процессора.
COMMAND	Команда, связанная с процессом, включая все его аргументы.

Кстати, вы можете заметить, что команда **ps aux** не включает знакомую черту перед переключателями **aux**. В этом случае команда работает с чертой и без нее (хотя немного по-другому). Допустимые параметры команды с тире также известны как **стиль UNIX или POSIX**; напротив, опции без тире известны как стиль BSD. Следующая альтернатива включает текущие переменные среды для каждого процесса:

$ ps eux

Процессы могут быть организованы в виде дерева. В частности, первый процесс с **PID** 1 - это **systemd**. Этот процесс является основой дерева, что может быть показано с помощью команды **pstree**. В некоторых случаях невозможно использовать стандартную команду **kill** для уничтожения процесса. В таких случаях ищите «родителя» процесса в дереве. Вы можете определить родителя процесса, известного как **PPID**, с помощью следующей команды:

$ ps axl

Переключатель **l** отображает вывод в длинном формате и не совместим с переключателем **u**. Вы можете просмотреть **PID** и **PPID** всех запущенных процессов на **рисунке 9-2.**

РИСУНОК 9-2. Вывод команды ps axl

```
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
4     0     1     0  20   0 134996  6924 ep_pol Ss   ?          0:19 /usr/lib/system
d/systemd --switched-root --system --deserialize 23
1     0     2     0  20   0      0     0 kthrea S    ?          0:00 [kthreadd]
1     0     3     2  20   0      0     0 smpboo S    ?          0:00 [ksoftirqd/0]
1     0     5     2   0 -20      0     0 worker S<   ?          0:00 [kworker/0:0H]
1     0     7     2 -100  -      0     0 smpboo S    ?          0:00 [migration/0]
1     0     8     2  20   0      0     0 rcu_gp S    ?          0:00 [rcu_bh]
1     0     9     2  20   0      0     0 rcu_no S    ?          0:00 [rcuob/0]
1     0    10     2  20   0      0     0 -      R    ?          0:11 [rcu_sched]
1     0    11     2  20   0      0     0 rcu_no S    ?          0:20 [rcuos/0]
5     0    12     2 -100  -      0     0 smpboo S    ?          0:06 [watchdog/0]
1     0    13     2   0 -20      0     0 rescue S<   ?          0:00 [khelper]
5     0    14     2  20   0      0     0 devtmp S    ?          0:00 [kdevtmpfs]
1     0    15     2   0 -20      0     0 rescue S<   ?          0:00 [netns]
1     0    16     2   0 -20      0     0 rescue S<   ?          0:00 [writeback]
1     0    17     2   0 -20      0     0 rescue S<   ?          0:00 [kintegrityd]
1     0    18     2   0 -20      0     0 rescue S<   ?          0:00 [bioset]
1     0    19     2   0 -20      0     0 rescue S<   ?          0:00 [kblockd]
1     0    20     2  20   0      0     0 hub_th S    ?          0:00 [khubd]
1     0    21     2   0 -20      0     0 rescue S<   ?          0:00 [md]
:
```

С ключом **-Z** (это заглавная буква Z) команда **ps** также может идентифицировать контексты **SELinux**, связанные с процессом. Например, следующая команда включает контексты SELinux каждого процесса в начале вывода. Если вы читали главу 4, контексты уже должны казаться вам знакомыми. Например, сопоставьте контекст процесса сервера vsFTP со следующей выдержкой:

system_u:system_r:ftpd_t:s0-s0:c0.c1023 2059 ? Ss 0:00 /usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf

Сравните это с контекстом реального демона. Роль объекта работает с реальным демоном; вы можете просмотреть его с другими демонами в каталоге / usr / sbin. Демон vsftpd работает со связанным файлом конфигурации с типом etc_t. Напротив, только демон vsftpd является исполняемым с типом ftpd_exec_t.

-rwxr-xr-x. root root system_u:object_r:ftpd_exec_t:s0 /usr/sbin/vsftpd

Роль различных демонов и их соответствующих процессов должна совпадать и сопоставлятся аналогичным образом. Если они этого не делают, демон может не работать, и проблема должна быть задокументирована в журнале аудита, описанном в главе 4, в каталоге **/var/log/audit**.

Просмотр загрузок с помощью команды top

Команда **top** сортирует активные процессы в первую очередь по их загрузке процессора и использованию памяти RAM. Посмотрите на **рисунок 9-3**. Он предоставляет обзор текущего состояния системы, начиная с текущее время безотказной работы, количество подключенных пользователей, активные и спящие задачи, загрузка процессора и многое другое. Результатом является, по сути, браузер задач.

РИСУНОК 9-3. Выход из верхней команды

```
top - 21:19:27 up 26 days, 26 min,  5 users,  load average: 0.75, 0.24, 0.17
Tasks: 169 total,   2 running, 167 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.7 us,  0.3 sy,  0.0 ni, 95.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.3 st
KiB Mem:   2279972 total,  2100720 used,   179252 free,     2612 buffers
KiB Swap:  1679356 total,        0 used,  1679356 free.   873464 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 2831 alex      20   0 1809068 467208  39644 S  4.0 20.5  70:33.07 gnome-shell
 3240 root      20   0  123648   1572   1092 R  0.3  0.1   0:00.29 top
    1 root      20   0  134996   6924   3752 S  0.0  0.3   0:19.12 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.21 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.29 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcuob/0
   10 root      20   0       0      0      0 S  0.0  0.0   0:11.42 rcu_sched
   11 root      20   0       0      0      0 R  0.0  0.0   0:20.25 rcuos/0
   12 root      rt   0       0      0      0 S  0.0  0.0   0:06.64 watchdog/0
   13 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 khelper
   14 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kdevtmpfs
   15 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 netns
   16 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 writeback
   17 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kintegrityd
```

Поле сортировки по умолчанию - загрузка ЦП. Другими словами, процесс, который потребляет больше ресурсов процессора, указан первым. Вы можете изменить поле сортировки с помощью клавиш со стрелками влево и вправо (<,>). Большинство столбцов такие же, как показано на **рисунке 9-2,** как показано в таблице 9-1. Дополнительные столбцы описаны в **таблице 9-2.**

ТАБЛИЦА 9-2 Дополнительные столбцы вывода сверху

Название колонки	Описание
PR	Приоритет задачи. Для получения дополнительной информации см. Команды **nice** и **renice**.
NI	Хорошая ценность задачи, корректировка приоритета.
VIRT	Виртуальная память в КиБ, используемая задачей.
RES	Физическая память, используемая процессом, не включая пространство подкачки, в КиБ (аналогично **RSS** в выводе команды **ps aux**).
SHR	Общая память в КиБ доступна для задачи.
S	Состояние процесса (аналогично **STAT** в выходных данных команды **ps aux**).
%CPU	Загрузка **ЦП**, как процент времени, потраченного на работу с момента последнего обновления главного экрана.

Одна проблема с командами **top** и **ps** заключается в том, что они отображают состояние процессов в системе как моментальный снимок. Этого может быть недостаточно. Процессы могут загружать систему всего за мгновение или даже за периодическое время. Один из

способов получения дополнительной информации об общей загрузке системы - две команды из пакета **sysstat: sar и iostat**. Эта информация о системной активности регистрируется благодаря командам **sa1 и sa2**, связанным со скриптом **/etc/cron.d/sysstat**, который будет вскоре описан.

Отчеты об активности системы с помощью команды sar

По сути, команда **sar** может использоваться для предоставления отчета о работе системы. Например, на **рисунке 9-4** показан вывод команды **sar -A**.

РИСУНОК 9-4 Вывод команды sar -A

```
Linux 3.10.0-123.el7.x86_64 (Maui)      14/03/15        _x86_64_        (8 CPU)

21:05:31          LINUX RESTART

21:10:01        CPU      %usr     %nice      %sys   %iowait    %steal      %irq     %soft    %guest    %gnice     %idle
21:20:01        all      0.10      0.31      0.26      0.06      0.00      0.00      0.00      0.24      0.00     99.02
21:20:01          0      0.12      0.04      0.16      0.04      0.00      0.00      0.00      0.34      0.00     99.30
21:20:01          1      0.12      0.03      0.12      0.10      0.00      0.00      0.00      0.35      0.00     99.27
21:20:01          2      0.14      0.08      0.13      0.03      0.00      0.00      0.00      0.21      0.00     99.41
21:20:01          3      0.16      0.05      0.09      0.01      0.00      0.00      0.00      0.22      0.00     99.47
21:20:01          4      0.01      0.68      0.64      0.00      0.00      0.00      0.00      0.15      0.00     98.52
21:20:01          5      0.04      1.44      0.79      0.01      0.00      0.00      0.00      0.01      0.00     97.72
21:20:01          6      0.06      0.08      0.05      0.01      0.00      0.00      0.00      0.39      0.00     99.41
21:20:01          7      0.14      0.13      0.09      0.30      0.00      0.00      0.00      0.26      0.00     99.08
21:30:01        all      0.01      0.00      0.02      0.03      0.00      0.00      0.00      0.03      0.00     99.90
21:30:01          0      0.01      0.00      0.03      0.03      0.00      0.00      0.00      0.10      0.00     99.83
21:30:01          1      0.01      0.00      0.03      0.09      0.00      0.00      0.00      0.03      0.00     99.85
21:30:01          2      0.01      0.00      0.05      0.01      0.00      0.00      0.00      0.11      0.00     99.82
21:30:01          3      0.02      0.00      0.02      0.00      0.00      0.00      0.00      0.02      0.00     99.95
21:30:01          4      0.01      0.00      0.02      0.01      0.00      0.00      0.00      0.00      0.00     99.96
21:30:01          5      0.01      0.00      0.01      0.01      0.00      0.00      0.00      0.00      0.00     99.97
21:30:01          6      0.00      0.00      0.01      0.01      0.00      0.00      0.00      0.00      0.00     99.98
21:30:01          7      0.04      0.00      0.02      0.08      0.00      0.00      0.00      0.00      0.00     99.87
:
```

Как вы можете видеть, выходные данные показывают различные измерения **ЦП** в разные моменты времени. Настройки по умолчанию измеряют нагрузку на процессор с 10-минутными интервалами. Эта система имеет восемь логических процессоров (четыре ядра с включенной гиперпоточностью), которые измеряются по отдельности и в целом. Большие числа простоя, показанные на рисунке, являются хорошим признаком того, что процессор не перегружен; однако на рисунке показана нагрузка менее часа.

10-минутные интервалы, связанные с выводом команды **sar**, определяются обычным заданием в каталоге **/etc/cron.d**. Выходные данные этих отчетов собираются в файлах журналов в каталоге **/var/log/sa**. Имена файлов связаны с числовым днем месяца; например, отчет о состоянии системы за **15-е** число месяца можно найти в Файл sa15 в указанном каталоге. Однако такие отчеты обычно хранятся как минимум в течение последних 28 дней, основываясь на следующем значении по умолчанию в файле /**etc/sysconfig/sysstat**:

HISTORY=28

Статистика процессора и запоминающего устройства с iostat

В отличие от **sar**, команда **iostat** сообщает более общую статистику **ввода/вывода** для системы, не только для процессора, но и для подключенных устройств хранения, таких как локальные диски и подключенные общие каталоги **NFS**. Пример, показанный на рис. 9-5, отображает информацию о процессоре и устройствах хранения с момента запуска системы на **server1.example.com**.

И команда **sar**, и команда **iostat** могут собирать статистику через регулярные промежутки времени. В качестве примера, следующая команда показывает статистику процессора и устройства хранения данных каждый
пять секунд и останавливается через минуту (12 отчетов):

 # iostat 5 12

РИСУНОК 9-5 Статистика процессора и устройства хранения

```
[root@server1 ~]# iostat
Linux 3.10.0-123.13.2.el7.x86_64 (server1.example.com)  14/03/15        _x86_64_
(1 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.88    0.01    0.18    0.00    0.02   98.92

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
vda               0.85         1.99         4.56    1051023    2405632
dm-0              0.85         1.84         4.56     970234    2403513
dm-1              0.00         0.00         0.00       1464          0

[root@server1 ~]#
```

Вариации на sar с sa1 и sa2

Команды **sa1 и sa2** часто используются для сбора данных отчета об активности системы. В **/etc/cron.d/sysstat** скрипт, команда **sa1** используется для сбора данных о системной активности каждый 10 минут. В этом же файле **cron** команда **sa2** записывает ежедневный отчет в каталог **/var/log/sa**. Как отмечается в сценарии, этот отчет обрабатывается каждый день, за семь минут до полуночи.

nice и renice

Команды **nice** и **renice** могут использоваться для управления приоритетом различных процессов. В то время как команда **nice** используется для запуска процесса с другим приоритетом, команда **renice** используется для изменения приоритета запущенного в данный момент процесса.

Приоритеты процессов в **Linux** указывают числа, которые кажутся нелогичными. Диапазон доступных порядковых номеров может варьироваться **от –20 до 19**. Стандартное порядковое число процесса унаследовано от родительского и обычно равно 0. Процессу с приоритетом 19 придется ждать, пока система почти полностью не освободится, прежде чем беру любые ресурсы. Напротив, процесс с приоритетом –20 имеет приоритет над всеми другими процессами. На практике это справедливо почти для всех процессов, потому что задачи «в реальном времени» имеют приоритет над самым низким значением –20. Но это выходит за рамки экзамена RHCSA, поэтому пока игнорируйте существование процессов реального времени, и ради этого обсуждения предположим, что всем нормальным процессам может быть присвоено подходящее значение от –20 до 19.

Команда **nice** предваряет другие команды. Например, если у вас есть интенсивный скрипт для запуска ночью, вы можете запустить его с помощью команды, подобной следующей:

$ nice -n 19 ./intensivescript

Эта команда запускает указанный скрипт с наименьшим возможным приоритетом. Если сценарий запускается ночью (или в другое время, когда система не загружается другими программами), сценарий запускается до тех пор, пока не будет запланировано практически любое другое задание, например сценарий в одном из каталогов **/etc/cron.*.** для выполнения. Поскольку такие сценарии выполняются по расписанию, они обычно должны иметь приоритет над некоторыми настроенными пользователем программами.

Иногда программа просто занимает слишком много ресурсов. Если вы не хотите завершать процесс, вы можете понизить его приоритет с помощью команды **renice**. Как правило, самый простой способ определить процесс, который потребляет слишком много ресурсов, - использовать команду **top**. Определите **PID**, который занимает слишком много ресурсов. Этот номер **PID** находится в левом столбце вывода.

Если **PID** целевого процесса равен **1234**, следующая команда изменит правильное число этого процесса на **10**, что придает этому процессу более низкий приоритет, чем значение по умолчанию **0**:

 # renice -n 10 1234

Если вы хотите уменьшить хороший уровень процесса, вы должны запустить **renice** от имени пользователя root. Несмотря на то, что выходные данные команды ссылаются на «**приоритет**», в действительности они просто перечисляют старые и новые «**nice**» числа для процесса:

1234: old priority 0, new priority, 10

Новое значение **nice** отображается в выводе команды **top** под столбцом **NI**.

Команды kill процесса

Иногда недостаточно переназначить процесс. Некоторые процессы могут просто перегружать систему. В большинстве случаев вы можете остановить такие сложные процессы с помощью команд **kill** и **killall**. Во многих случаях вы можете убить(kill) процесс прямо из браузера задач.

Если есть ситуация, когда процесс занимает много памяти или **ЦП**, он, вероятно, замедляет все остальное, работающее в этой системе. Как показано на **рисунке 9-6, Firefox** довольно сильно загрузил процессор указанной системы. Если бы он не отвечал, мы бы нажали кнопку **k** в браузере задач.

РИСУНОК 9-6 Главный браузер задач top с большой загрузкой Firefox

```
top - 13:57:44 up 10 min,  3 users,  load average: 0.29, 0.32, 0.25
Tasks: 257 total,   1 running, 256 sleeping,   0 stopped,   0 zombie
%Cpu(s):  9.6 us,  1.6 sy,  0.0 ni, 88.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:  16153912 total,  3167324 used, 12986588 free,     1212 buffers
KiB Swap: 16383316 total,        0 used, 16383316 free.   915604 cached Mem
PID to signal/kill [default pid = 4537]
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 4537 alex      20   0 2357456 613972  57400 S  85.5  3.8   0:27.09 firefox
 3375 alex       9 -11 1147288  24136  17980 S   4.3  0.1   0:06.18 pulseaudio
 2867 root      20   0  213992  24264  12312 S   1.3  0.2   0:18.29 Xorg
 3443 alex      20   0 1800404 110844  34464 S   1.3  0.7   0:27.82 gnome-shell
 3762 alex      20   0  622768  21156  12856 S   0.7  0.1   0:00.82 gnome-term+
 2846 qemu      20   0 5688400 691360   7468 S   0.3  4.3   0:41.09 qemu-kvm
 3471 alex      20   0  461276   5672   3468 S   0.3  0.0   0:00.39 ibus-daemon
    1 root      20   0  134836   6900   3776 S   0.0  0.0   0:01.31 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.00 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    7 root      rt   0       0      0      0 S   0.0  0.0   0:00.04 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/1
   11 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/2
   12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/3
```

Как показано на рисунке, команда **k** отображает приглашение **PID To Signal/Kill:,** где мы вводим **PID** процесса **Firefox** или принимаем значение по умолчанию **4537**, которое выглядит как **Firefox**. Он применяет сигнал по умолчанию (**SIGTERM**) к процессу с этим номером **PID**.

Конечно, вы можете применить команду **kill** непосредственно к номеру **PID**. Например, следующая команда эквивалентна шагам, только что описанным в главном браузере задач:

 # kill 4537

ТАБЛИЦА 9-3 Список общих сигналов POSIX

Название сигнала	Номер сигнала	Описание
SIGHUP	**1**	Перезагрузка конфигурации.
SIGINT	**2**	Прерывание клавиатуры (**Ctrl-C**). Вызывает завершение программы.
SIGKILL	**9**	Завершает программу немедленно.
SIGQUIT	**15**	Аналогичен **SIGKILL**, но программа может игнорировать или обрабатывать сигнал для освобождения существующих ресурсов и выполнения чистого завершения.
SIGCONT	**18**	Возобновляет приостановленный процесс.
SIGSTOP	**19**	Временно приостанавливает выполнение процесса.

Команда **kill** может быть запущена владельцем процесса из его учетной записи. Таким образом, пользователь **alex** может запустить команду **kill 4537** из своей обычной учетной записи, поскольку он обладает административными привилегиями над процессами, связанными с его именем пользователя.

Несмотря на свое название, команда **kill** может отправлять самые разные сигналы различным процессам. Для получения полного списка выполните команду **kill -l** или введите **man 7 signal**. В **таблице 9-3** перечислены некоторые из наиболее распространенных сигналов.

До появления **systemd** и сценариев в каталоге **/etc/init.d** команда **kill -1** использовалась для отправки сигнала перезагрузки конфигурации демонам службы. Например, если номер **PID** основного процесса, связанного с **веб-сервером Apache, равен 2059**, следующая команда функционально эквивалентна команде **systemctl reload httpd**:

 # kill -1 2059

Без переключателя **-1** (и это тире номер 1) команда **kill** в нормальных условиях завершит данный процесс. В этом случае это приведет к прекращению работы **веб-сервера Apache**. Но иногда процессы застревают. В некоторых таких случаях команда **kill** не работает сама по себе. Процесс продолжается. В этом случае вы можете попробовать две вещи.

Во-первых, вы можете попробовать команду **kill -9**, которая пытается «нечисто» остановить процесс, отправив сигнал **SIGTERM**. Если это успешно, другие связанные процессы могут все еще оставаться в работе.

Иногда несколько процессов выполняются под одним и тем же именем. Например, как вы увидите в главе 14, **веб-сервер Apache** запускает несколько процессов, которые выполняются одновременно. В лучшем случае неэффективно убивать только один процесс; следующая команда уничтожит все запущенные в настоящий момент процессы сервера, не допуская других проблем:

 # killall httpd

Архивы и Сжатие

Linux включает в себя различные команды для архивирования групп файлов. Некоторые архивы могут быть переработаны в пакеты, такие как **RPM**. Другие архивы просто используются в качестве резервных копий. В любом случае архивы могут быть очень удобными, особенно когда они сжаты. В этом разделе рассматриваются команды архивирования и сжатия, специально указанные в целях **RHCSA**. Эти «основные инструменты» включают команды **gzip, bzip2, tar и star**.

gzip и bzip2

Команды **gzip** и **bzip2** функционально похожи, поскольку они сжимают и распаковывают файлы, просто используя разные алгоритмы. Команда **gzip** использует алгоритм **DEFLATE**, тогда как команда **bzip2** использует алгоритм сортировки блоков **Burrows-Wheeler**. Хотя они оба работают хорошо, команда **bzip2** имеет лучшую степень сжатия. Например, любая из двух следующих команд может быть использована для сжатия большого файла документа с именем **big.doc**:

```
# gzip big.doc
# bzip2 big.doc
```

Это добавляет к файлу суффикс **.gz или .bz2**, сжатый в соответствующие алгоритмы. С ключом -d вы можете использовать те же команды, чтобы полностью изменить процесс:

```
# gzip -d big.doc.gz
# bzip2 -d big.doc.bz2
```

В качестве альтернативы команды **gunzip и bunzip2** могут использоваться для одной и той же цели.

tar

Команда **tar** изначально была разработана для архивирования данных на ленточные накопители. Однако сегодня он обычно используется для сбора ряда файлов, особенно из каталога, в один архивный файл. Например, следующая команда создает резервную копию информации из **/home** каталог в файле **home.tar.gz:**

```
# tar czvf home.tar.gz /home
```

Как и команда **ps**, это одна из немногих команд, которая не требует тире перед опциями. Эта конкретная команда создает **(c)** архив, сжимает **(z)** его, выводит на экран процесс архивирования **(v)** с именем файла **(f)**, которое следует. Кроме того, вы можете извлечь **(x)** из этого файла с помощью следующей команды:

```
# tar xzvf home.tar.gz /home
```

Указанное сжатие **(z)** связано с командой **gzip**; если вы хотите использовать сжатие **bzip2**, замените опцию, на **(j)**. Команда **tar** может сохранять и извлекать настройки списка управления доступом или атрибуты **SELinux** с параметром **--selinux**.

Если у вас есть архив **tar**, созданный без опции **--selinux**, вы можете компенсировать это. Вы можете использовать такие команды, как **restorecon**, как описано в Главе 4, для восстановления контекста **SELinux** архива.

star

Команда **star** приобрела некоторую популярность, потому что она была первой, кто ввел поддержку архивирования файлов в системе **SELinux**. Поскольку команда **star** обычно не установлена, вам необходимо установить ее; один метод с помощью следующей команды:

 # yum install star

К сожалению, команда **star** работает не так, как **tar**. Если вам когда-либо понадобится использовать **star**, попрактикуйтесь в ней. Например, следующая команда создаст архив со всеми контекстами **SELinux** из текущего каталога **/home:**

 # star -xattr -H =exustar -c -f=home.star /home/

Ключ **-xattr** сохраняет расширенные атрибуты, связанные с **SELinux**. Ключ **-H=exustar** записывает архив в формате **exustar**, который позволяет вам хранить списки **ACL**, если указана опция **-acl. -c** создает новый файл архива. -f указывает имя файла архива.

После создания архива его можно распаковать с помощью следующей команды, которая извлекает архив:

 # star -x -f=home.star

При желании архив можно сжать с помощью вышеупомянутой команды **gzip или bzip2** или из **star** с параметром командной строки **-z или -bz**. Команда **star -x** может обнаруживать и восстанавливать файлы из архивов, настроенных с использованием различных схем сжатия. Например, на основе сжатого **gzip** архива команда **star** распаковывает этот архив, как отмечено в следующем информационном сообщении журнала:

star: WARNING: Archive is 'gzip' compressed, trying to use the -z option

ЦЕЛЬ СЕРТИФИКАЦИИ 9.02

Автоматизация системного администрирования: cron и at

Система **cron** - это умный будильник. Когда звучит сигнал тревоги, **Linux** автоматически запускает выбранные вами команды. Вы можете установить будильник на все виды регулярных временных интервалов. Многие задания **cron** запланированы на выполнение в середине ночи, когда активность пользователя ниже. Конечно, это время может быть скорректировано. В качестве альтернативы система **at** позволяет пользователям запускать команды по своему выбору, один раз, в указанное время в будущем.

RHEL 7 по умолчанию устанавливает демон **cron** и включает систему **anacron** в **cron**. Демон **cron** запускает задания по регулярному расписанию. Система **anacron** помогает демону **cron** работать в системах, которые отключены ночью. Это гарантирует, что важные задания всегда выполняются, даже если система была отключена на некоторое время.

Система **cron** настроена на проверку каталога **/var/spool/cron** для заданий пользователя. Кроме того, он включает задания, определенные в файле **/etc/anacrontab** на основе сценария **0anacron** в каталоге **/etc/cron.hourly**. Он также проверяет запланированные задания для компьютера, описанного в файле **/etc/crontab** и в каталоге **/etc/cron.d**.

!!!!!
Потому что cron всегда проверяет изменения, вам не нужно перезапускать cron каждый раз, когда изменение было внесено.
!!!!!

Системный crontab и компоненты

Файл **/etc/crontab** настроен в определенном формате. Каждая строка может быть пустой, комментарий (который начинается с #), переменная или строка конфигурации. Естественно, пустые строки и комментарии игнорируются. В некоторых дистрибутивах Linux этот файл содержит расписание заданий. В RHEL 7 файл **crontab** по умолчанию просто включает формат для других связанных файлов конфигурации.

Пользователи запускают обычные команды. Любой, кто запускает новый процесс, будь то вы или демон, наследует «среду», состоящую из различных переменных среды. Чтобы увидеть переменные окружения для текущего пользователя, выполните команду **env**. Если этот пользователь является вашей учетной записью, некоторые из стандартных переменных в **RHEL** включают **HOME**, который должен соответствовать вашему домашнему каталогу, **SHELL**, который должен соответствовать оболочке по умолчанию, и **LOGNAME** в качестве имени пользователя.

Другие переменные могут быть установлены в **/etc/crontab** и других файлах **cron** (в **/etc/cron.d, /etc/cron.daily и т. д.):**

Variable=Value

Некоторые переменные уже установлены для вас. Например, **MAIL** - это **/var/spool/mail/michael**, если ваше имя пользователя - **michael**, **LANG - en_US.UTF-8, а PATH -** это место, где оболочка ищет команды. Вы можете установить эти переменные на разные значения в разных файлах конфигурации **cron**. Например, файл по умолчанию **/etc/crontab** содержит следующие переменные:

SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

Обратите внимание, что значения **PATH** и **MAILTO** отличаются от стандартных переменных среды. Переменная **PATH** в файле конфигурации **cron** может отличаться от переменной **PATH**, связанной с оболочкой. На самом деле две переменные независимы. Таким образом, вы захотите указать точный путь каждой команды в каждом файле конфигурации **cron**, если его нет в **PATH crontab**.

!!!!!
Переменная MAILTO может помочь вам администрировать несколько систем Linux. Демон cron отправляет по электронной почте любые выходные данные, которые задание отправляет в stdout или stderr. Просто добавь
строка, такая как MAILTO=me@example.net, чтобы направить весь вывод заданий cron на этот адрес электронной почты.
!!!!

Формат строки в **/etc/crontab** теперь подробно описан в комментариях, как показано на **рисунке 9-7.** Каждый из этих столбцов более подробно поясняется в

Если вы видите звездочку в каком-либо столбце, демон **cron** запускает эту команду для всех возможных значений этого столбца. Например, поле ***** в минуте означает, что команда запускается каждую минуту в течение указанного часа. Рассмотрим пример, показанный здесь:

1 5 3 4 * ls

Эта строка запускает команду **ls 3 апреля в 5:01 утра**. Звездочка в столбце дня недели просто означает, что не имеет значения, какой это день недели; **crontab** все еще выполняет команду **ls** в указанное время.

РИСУНОК 9-7 Формат crontab

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,f
ri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

ТАБЛИЦА 9-4 Столбцы в файле конфигурации cron

Поле	Значение
minute(минут)	0-59.
Hour(час)	На основе 24-часовых часов; например, 23 = 11 p.m.
day of month(день месяца)	1-31.
month(месяц)	1–12, или январь, февраль, март и т. Д.
day of week(день недели)	0–7 (где 0 и 7 - воскресенье) или Sun, Mon, Tue и т. Д.
command(команда)	Команда, которая будет выполнена; в системном файле заданий **cron** этому предшествует имя пользователя для запуска команды от имени.

Записи, связанные с демоном **cron**, довольно удобные. Например, запись 7–10 в поле часа запускает указанную команду в 7:00, 8:00, 9:00 и 10:00. Список записей в поле минут, например 0,5,10,15,20,25,30,35,40,45,50,55, будет запускать указанную команду каждые пять минут. Но это много цифр. Ввод */5 в поле минут приведет к тому же результату. Демон **cron** также распознает сокращения месяцев и дня недели.

Фактическая команда - шестое поле. Вы можете установить новые строки с символом процента **(%).** Весь текст после первого знака процента отправляется команде в качестве стандартного ввода. Это полезно для форматирования стандартного ввода. Ниже приведен пример файла **cron**:

```
# crontab -l
# Sample crontab file
#
# Force /bin/bash to be my shell for all of my scripts.
SHELL=/bin/bash
# Run 15 minutes past Midnight every Saturday
15 0 * * sat $HOME/scripts/scary.script
# Do routine cleanup on the first of every Month at 4:30 AM
30 4 1 * * /usr/scripts/removecores >> /tmp/core.tmp 2>>&1
# Mail a message at 10:45 AM every Friday
45 10 * * Fri mail -s "Project Update" employees@example.com
%Can I have a status
update on your project?%%Your Boss.%
# Every other hour check for alert messages
0 */2 * * * /usr/scripts/check.alerts
```

Почасовые работы в cron

Теперь пришло время для некоторых примеров файлов **cron**. Обсуждаемые файлы и сценарии ограничены теми, которые можно увидеть в системе server1.example.com. Ряд

различных пакетов добавить свои **cron** вакансии. Определенные задания, связанные с демоном **cron**, выполняются каждый час на основе сценария **0hourly** в каталоге **/etc/cron.d**. Этот файл содержит те же переменные, что и только что описаный файл **/etc/crontab**. Для почасовых заданий она включает одну строку:

01 * * * * root run-parts /etc/cron.hourly

Учитывая информацию, представленную в предыдущем разделе, вы сможете прочитать эту строку. Команда **run-parts** загружает каждый скрипт в следующем каталоге; Сценарии в этом каталоге выполняются от имени пользователя **root**. Конечно, первые пять столбцов указывают время; Сценарии выполняются в одну минуту после часа, каждый час, каждый день, каждый месяц, в каждый день недели.

Интересующий скрипт в каталоге **/etc/cron.hourly** - **0anacron**, который просматривает содержимое файла **/var/spool/anacron/cron.daily**, чтобы увидеть, была ли команда **anacron** запущена в текущий день. Если нет, и если система не работает от батареи (например, на ноутбуке, отключенном от основного источника питания), выполняется команда **/usr/sbin/anacron -s**, которая запускает сценарии, определенные в файле конфигурации **/etc/anacrontab**, описанный ранее скрипт состояния системы хранится в файле **/etc/cron.d/sysstat**. В этом файле есть две активные команды. Первая команда, **sa1**, запускается каждые **10** минут, как показано ***/10**. Эта команда выполняется каждый час, каждый день и так далее.

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

Вторая команда **sa2** выполняется через **53** минуты после часа, **23**-го часа каждого дня. Другими словами, отчет о работе системы не собирается до **11:53** вечера. ночью, вечером.

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

Регулярные работы Anacron

Сценарий **0anacron** в каталоге **/etc/cron.hourly**, описанный ранее, выполняет команду **anacron** после включения системы. Эта команда выполняет три сценария, определенных в файле **/etc/anacrontab**.

Содержание файла
 # /etc/anacrontab: configuration file for anacron
 # See anacron(8) and anacrontab(5) for details.
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
 # the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
 # the jobs will be started during the following hours only
START_HOURS_RANGE=3-22
#period in days delay in minutes job-identifier command
1 5 cron.daily nice run-parts /etc/cron.daily
7 25 cron.weekly nice run-parts /etc/cron.weekly
@monthly 45 cron.monthly nice run-parts /etc/cron.monthly

Файл включает в себя три переменные среды, которые должны казаться знакомыми:

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=root

Директива **SHELL** может выглядеть немного иначе, но команда **ls -l /bin/sh** должна подтвердить символическу ссылку на команду **/bin/bash**, которая запускает оболочку **bash** по умолчанию. Следующая директива означает, что сценарии запускаются в произвольное время до **45** минут после запланированного времени:

RANDOM_DELAY = 45

В соответствии со следующей директивой задания **anacron** выполняются только между 3:00 и 10:59:00.

START_HOURS_RANGE = 3-22

Хотя формат **/etc/anacrontab** аналогичен формату, указанному в сценарии для обычной работы **cron**, существуют различия. Порядок данных в каждой строке определяется следующим комментарием:

#period in days delay in minutes job-identifier command
(#period в днях задержка в минутах команда идентификатора задания)

Период в днях равен **1, 7 или @monthly**, потому что число дней в месяце меняется. Задержка в минутах связана с директивой **RANDOM_DELAY**. Поскольку файл **/etc/anacrontab** выполняется с помощью скрипта **/etc/cron.d/0hourly**, часы запускаются через одну минуту после часа, после запуска системы. Задержка в минутах наступает перед директивой **RANDOM_DELAY.**

Другими словами, основываясь на следующей строке, сценарии в каталоге **/etc/cron.daily** могут выполняться где-то через **5–50** минут после запуска команды **anacron** или от **6 до 51** минуты после часа:

1 5 cron.daily nice run-parts /etc/cron.daily

Дополнительные примеры можно найти в некоторых сценариях в каталоге **/etc/cron.daily**. Вот три ключевых сценария, которые вы должны исследовать:

- **logrotate** Перемещает файлы журнала
- **mlocate** Обновляет файловую базу данных «**locate**».
- **man-db.cron** Создает или обновляет базу данных **mandb**.

Настройка cron для пользователей

Каждый пользователь может использовать команду **crontab** для создания и управления заданиями **cron** для своих учетных записей. Четыре переключателя связаны с командой **crontab**:

- **-u user** Позволяет пользователю **root** редактировать **crontab** другого конкретного пользователя.
- **-l** Перечисляет текущие записи в файле **crontab**.
- **-r** Удаляет записи **cron**.
- **-e** Редактирует существующую запись в **crontab**. По умолчанию **crontab** использует **vi**, если в переменной среды **EDITOR** не указан другой редактор.

Чтобы настроить записи **cron** для своей учетной записи, начните с команды **crontab -e**. Обычно он открывает файл в редакторе **vi**, где вы можете добавить соответствующие переменные и команды, аналогично тому, что вы видели в других файлах заданий **cron**.

После сохранения задания **cron** вы можете подтвердить изменение с помощью команды **crontab -l** или, как пользователь **root**, прочитав содержимое файла в каталоге **/var/spool/cron**, связанного с именем пользователя. Все текущие задания **cron** для пользователя можно удалить с помощью команды **crontab -r.**

УПРАЖНЕНИЕ 9-1

Создать cron

В этом упражнении вы измените свой **crontab**, чтобы прочитать текстовый файл в 13:05. каждый понедельник в январе месяце. Для этого выполните следующие действия:

1. Войдите как обычный пользователь.
2. Создайте каталог ~ /bin. Добавьте файл с именем **taxrem.sh**, который читает текстовый файл из вашего домашнего каталога. В файле **taxrem.sh** должно быть достаточно команды, такой как:
   #!/bin/bash
   cat /home/michael/reminder.txt
   Обязательно добавьте соответствующие строки в файл **remder.txt** в вашем домашнем каталоге, например «**Не забудьте сделать налоги**!». Убедитесь, что файл **taxrem** является исполняемым с помощью команды **chmod + x ~ /bin/taxrem.sh**
3. Откройте **crontab** для вашей учетной записи с помощью команды **crontab -e**.
4. Добавьте соответствующую команду в **crontab**. Исходя из описанных условий, он будет выглядеть следующим образом:
   5 13 * 1 1 /home/michael/bin/taxrem.sh
5. Не забывайте директивы, такие как **MAILTO=user@example.com** в начале **crontab**.
6. Сохранить и выйти. Запустите **crontab -l** и подтвердите наличие файла **cron** пользователя в каталоге **/var/spool/cron**. Этот файл должен иметь то же имя, что и пользователь.

Выполнение работы с системой at

Как и **cron**, **at** демон поддерживает обработку заданий. Тем не менее, вы можете установить на работу для запуска один раз. Задания в системе **cron** должны быть настроены на регулярную работу. Демон **at** работает аналогично процессу печати; задания помещаются в каталог **/var/spool/at** и запускаются в указанное время.

Вы можете использовать **at daemon** для запуска команды или сценария по вашему выбору. Для целей этого раздела предположим, что пользователь **michael** создал сценарий с именем **797.sh** в своем домашнем каталоге для обработки некоторой базы данных о продажах самолетов.

Из командной строки вы можете запустить команду **at time**, чтобы запустить задание, которое будет запущено в указанное время. Здесь время может быть сейчас; в указанное количество минут, часов или дней; или во время по вашему выбору. Несколько примеров приведены в **таблице 9-5**.

Вы можете использовать одну из примеров команд, показанных в **Таблице 9-5**, чтобы открыть задание **at**. Он открывает другой интерфейс командной строки, где вы можете указать команду по вашему выбору. В этом примере предположим, что вы собираетесь уйти с работы и хотите начать работу через час. Из указанных условий выполните следующие команды:

$ at now + 1 hour
at> /home/michael/797.sh
at> Ctrl-D

Команда **ctrl-d** выходит из оболочки at и возвращает к исходному интерфейсу командной строки.

ТАБЛИЦА 9-5 Примеры команд

Временной период	Пример	Время начала работы
Minutes	**at now + 10 minutes**	In 10 minutes
Hours	**at now + 2 hours**	In 2 hours
Days	**at now + 1 day**	In 24 hours
Weeks	**at now + 1 week**	In 7 days
n/a	**at teatime**	At 4:00 p.m.
n/a	**at 3:00 12/21/16**	On December 21, 2016, at 3:00 a.m.

В качестве альтернативы вы можете использовать перенаправление ввода следующим образом:

$ at now + 1 hour < /home/michael/797.sh

Команда **atq**, как показано здесь, проверяет состояние тока в заданиях. Все ожидающие задания перечислены в выводе команды **atq**:

$ atq
1 2016-12-21 03:00 a michael

Если есть проблема с заданием, вы можете удалить его с помощью команды **atrm**. Например, вы можете удалить отмеченное задание, помеченное как задание 1, с помощью следующей команды:

$ atrm 1

Безопасный cron и at

Возможно, вы не хотите, чтобы все могли работать среди ночи. Вы также можете ограничить эту привилегию по соображениям безопасности.

Пользователи могут быть настроены в файлах **/etc/cron.allow и /etc/cron.deny**. Если ни один из этих файлов не существует, использование **cron** разрешено только пользователю с правами администратора. Если файл **/etc/cron.allow** существует, только пользователи, указанные в этом файле, могут использовать cron. Если файл **/etc/cron.allow** отсутствует, только пользователи, указанные в **/etc/cron.deny**, не могут использовать **cron**.

Эти файлы отформатированы как одна строка на пользователя; если вы включите следующие записи в **/etc/cron.deny**, а файл **/etc/cron.allow** не существует, пользователям **elizabeth** и **nancy** не разрешено создавать свои собственные сценарии **cron**:

elizabeth
nancy

Однако, если файл **/etc/cron.allow** существует с тем же списком пользователей, он имеет приоритет. В этом случае и пользователям, и Элизабет, и Нэнси разрешено создавать свои собственные скрипты **cron**. Диапазон возможностей представлен в **Таблице 9-6**.

Безопасность пользователя для системы **at** практически идентична. Соответствующими файлами конфигурации безопасности являются **/etc/at.allow и /etc/at.deny**. Диапазон возможностей суммирован в **Таблице 9-7**.

Если вы не уверены в безопасности, возможно, будет целесообразно включить только желаемых пользователей в файлы **/etc/cron.allow и /etc/at.allow**. В противном случае нарушение

безопасности в служебной учетной записи может позволить хакеру «черной шляпы» запустить **cron** или по сценарию из связанной учетной записи.

ТАБЛИЦА 9-6 Эффекты безопасности cron.allow и cron.deny

	/etc/cron.deny существует	**/etc/cron.deny не существует**
/etc/cron.allow существует	Только пользователи, перечисленные в **/etc/cron.allow** могут выполнить **crontab -e**; содержание **/etc/cron.deny** игнорируются.	Только пользователи, перечисленные в **/etc/cron.allow**, могут выполнить **crontab -e**.
/etc/cron.allow не существует	Все пользователи, перечисленные в **/etc/cron.deny**, не могут использовать **crontab -e**.	Только пользователь **root** может запускать **crontab -e**.

ТАБЛИЦА 9-7 Эффекты безопасности at.allow и at.deny

	/etc/at.deny существует	**/etc/at.deny не существует**
/etc/at.allow существует	Только пользователи, перечисленные в **/etc/at.allow** может запустить команду; **at /etc/at.deny** игнорируются.	Только пользователи, перечисленные в **/etc/at.allow**, могут запускать команду **at**.
/etc/at.allow не существует	Все пользователи, перечисленные в **/etc/at.deny**, не могут Запустите команду.	Только пользователь **root** может запустить команду **at**.

ЦЕЛЬ СЕРТИФИКАЦИИ 9.03

Анализ локальных файлов журнала

Важной частью поддержания защищенной системы является мониторинг тех действий, которые происходят в системе. Если вы знаете, что обычно происходит, например, понимаете, когда пользователи входят в систему, вы можете использовать файлы журналов, чтобы обнаружить необычные действия. **Red Hat Enterprise Linux** поставляется с новыми утилитами для мониторинга системы, которые могут помочь определить виновника в случае возникновения проблемы.

RHEL 7 поставляется с двумя системами ведения журнала: традиционной службой ведения журнала, **rsyslog**, и усовершенствованным демоном ведения журнала, известным как **systemd-journald**. Мы кратко обсудили ведение журнала **systemd** в главе 5. Благодаря своей архитектуре **systemd** может перехватывать и сохранять все загрузочные сообщения и сообщения системного журнала, а также вывод, который сервисы отправляют со стандартной ошибкой и со стандартным выводом. Это гораздо больше, чем может сделать традиционный сервер системного журнала. По умолчанию журналы журнала **systemd** временно хранятся (в файловой системе **RAM tmpfs**) в каталоге **/run/log/journal.**

Демон **rsyslog** включает в себя функциональные возможности ядра и системных журналов, используемых через **RHEL 7**. Вы можете использовать файлы журнала, сгенерированные таким образом, для отслеживания действий в системе. Способ вывода журнала **rsyslog** в файлы основан на конфигурации, определенной в файл **/etc/rsyslog.conf** и файлы в каталоге **/etc/rsyslog.d**.

Во многих случаях такие службы, как **SELinux, Apache и Samba**, имеют свои собственные файлы журналов, определенные в их собственных файлах конфигурации. Подробности рассматриваются в главах, связанных с этими услугами.

Файл конфигурации системного журнала

Вы можете настроить то, что регистрируется через файл конфигурации **/etc/rsyslog.conf**. Как показано на **рисунке 9-8**, он включает в себя набор правил для различных условий: **authpriv, cron, kern, mail, news, user и uucp.**

Каждое **условие** также связано с несколькими различными уровнями регистрации, известными как приоритет. В порядке возрастания приоритетами журнала являются **debug(**отладка), **info**(информация), **notice**(уведомление), **warn**(предупреждение), **err**(ошибка), **crit**(критическое замечание), **alert**(предупреждение), **emerg**(экстренное сообщение). Существует также общий приоритет отсутствия, который не регистрирует сообщения конкретного объекта; например, директива authpriv.none пропустит все сообщения аутентификации.

Для каждого условия и приоритета информация журнала отправляется в определенный файл журнала. Например, рассмотрим следующую строку из **/etc/syslog.conf**:

***.info;mail.none;authpriv.none;cron.none /var/log/messages**

РИСУНОК 9-8 Конфигурационный файл rsyslog.conf

```
#### RULES ####

# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog


# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 :omusrmsg:*

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
:
```

Эта строка отправляет информацию журнала из всех указанных средств в файл **/var/log/messages**. Сюда входят все служебные сообщения информационного уровня и выше, за исключением сообщений журнала, относящихся к службам **mail, authpriv** (аутентификация) и **cron**.

Вы можете использовать звездочку в качестве символа подстановки в /**etc/syslog.conf**. Например, строка, начинающаяся с ***. ***, Указывает демону **rsyslogd** на запись всего. Строка,

начинающаяся с **authpriv. *,** Означает, что вы хотите регистрировать все сообщения из условия **authpriv**.

По умолчанию **rsyslogd** регистрирует все сообщения с заданным приоритетом или выше. Другими словами, строка **cron.err** будет содержать все сообщения журнала от демона **cron** на уровнях **err, crit, alert и emerg**.

Большинство сообщений от демона **rsyslogd** записываются в файлы в каталоге **/var/log.** Вы должны регулярно сканировать эти журналы и искать шаблоны, которые могут указывать на нарушение безопасности. Также можно настроить задания **cron** для поиска таких шаблонов.

Управление файлами журнала

Журналы могут легко стать очень большими и трудными для чтения. По умолчанию утилита **logrotate** создает новый файл журнала еженедельно, используя директивы в файле **/etc/logrotate.conf**, который также извлекает директивы из файлов в каталоге **/etc/logrotate.d**. Как показано на **рисунке 9-9**, директивы в файле просты и хорошо объясняются в комментариях.

РИСУНОК 9-9 Ротация журнала настроена в /etc/logrotate.conf

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# use date as a suffix of the rotated file
dateext

# uncomment this if you want your log files compressed
#compress

# RPM packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp and btmp -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
        minsize 1M
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0600 root utmp
    rotate 1
}

# system-specific logs may be also be configured here.
```

В частности, настройки по умолчанию меняют файлы журналов еженедельно, сохраняя журналы за последние четыре недели. Новые файлы журналов создаются во время ротации, а старые файлы имеют дату ротации в качестве суффикса. Различные положения даны для журналов **wtmp** и **btmp**, связанных с записями пользователей.

Разнообразие лог файлов

Различные файлы журналов и их функциональные возможности описаны в **таблице 9-8**. Эти файлы создаются на основе ранее описанной конфигурации файла **/etc/rsyslog.conf** и файлов конфигурации службы в каталоге **/etc/rsyslog.d**. Некоторые из файлов журналов (например, в **/var/log/httpd**) создаются непосредственно приложениями. Все показанные файлы находятся в каталоге **/var/log**. Если вы не установили, не активировали или не использовали указанную службу, связанный файл журнала может не отображаться. Напротив, вы можете увидеть файлы журнала, которые здесь не показаны, на основе дополнительных установленных сервисов.

ТАБЛИЦА 9-8 Стандартные файлы журналов Red Hat

Лог-файлы	Описание
anaconda/*	Включает как минимум пять файлов журнала: **anaconda.log** для общих сообщений об установке; **anaconda.packaging.log** для установки пакета; **anaconda.program.log** для вызовов на внешние программы; **anaconda.storage.log** для настройки и разбиения устройства хранения; **anaconda.ifcfg.log** для инициализации сетевого адаптера; и иногда системный журнал для сообщений ядра; и **anaconda.xlog** для первого запуска сервера графического интерфейса.
audit/	Включает файл **audit.log**, который собирает сообщения из подсистемы аудита ядра.
boot.log	Связан со службами, которые запускают и закрывают процессы.
btmp	Списки неудачных попыток входа в систему; читается с помощью команды **utmpdump btmp**.
cron	Собирает информацию из скриптов, запущенных демоном **cron**.
cups/	Каталог доступа к принтеру, страницы и журналы ошибок.
dmesg	Включает в себя основные загрузочные сообщения.
gdm/	Каталог сообщений, связанных с запуском через Диспетчер отображения **GNOME**; включает ошибки входа в систему.
httpd/	Каталог файлов журналов, связанных с веб-сервером **Apache**.
lastlog	Списки входов в систему; читается с помощью команды **lastlog.**
maillog	Собирает сообщения журнала, относящиеся к почтовым серверам.
messages	Включает журналы ядра и сообщения от других служб, как определено в **/etc/rsyslog.conf**.
pm-powersave.log	Журнал сообщений, связанных с управлением питания.
ppp/	Каталог с протоколами протокола «точка-точка»; обычно связаны с телефонными модемами.
rhsm/	Каталог с журналами из плагина **Red Hat Subscription Manager.**
sa/	Справочник с отчетами о работе системы.
samba/	Каталог доступа и журналы обслуживания для сервера **Samba**.
secure	Аутентификация и доступ к сообщениям.
spooler	Показывает файл журнала, который может содержать критические сообщения.
sssd/	Каталог сообщений, связанных с демоном **System Security Services.**
tallylog	Поддерживает **pam_tally**, который блокирует пользователя после чрезмерных неудачных попыток входа в систему.
up2date	Включает в себя сообщения журнала от агента обновлений **Red Hat**.
wtmp	Список логинов в двоичном формате; можно прочитать с помощью команды **utmpdump**.
xferlog	Добавляет сообщения, связанные с передачей файлов с локального **FTP**-сервера.
Xorg.0.log	Примечания по установке сообщений для системы **X Window**; может включать проблемы с конфигурацией.

yum.log	Журналы пакетов установлены, обновлены и удалены с помощью **yum**.

Журналы для конкретных сервисов

Как предлагалось ранее, ряд сервисов контролируют свои собственные файлы журналов. Например, файлы журнала для сервера **vsFTP** настраиваются в файле **vsftpd.conf** в каталоге **/etc/vsftpd**. Как отмечено в этом файле, следующая директива позволяет регистрировать как загрузки, так и отдачи в файле **/var/log/xferlog:**

xferlog_enable=YES

Регистрация других сервисов может быть более сложной. Например, отдельные файлы журнала настроены для доступа и ошибок на веб-сервере **Apache** в каталоге **/var/log/httpd.**

УПРАЖНЕНИЕ 9-2
Изучите файлы журналов

В этом упражнении вы будете проверять файлы журналов в локальной системе, чтобы попытаться выявить различные проблемы.

1. Перезагрузите компьютер **Linux**. Войдите в систему как пользователь **root**. Используйте неправильный пароль один раз.
2. Войдите в систему с правильным паролем как пользователь **root**.
3. В консоли перейдите в каталог **/var/log** и откройте файл с именем **«secure».** Перейдите к сообщению «**Failed password**», ближайшему к концу файла. Просмотрите, что случилось. Закройте файл.
4. Просмотрите другие журналы в каталоге **/var/log**. Используйте **Таблицу 9-8** для руководства. Ищите сообщения, связанные с оборудованием. В каких логах они находятся? Имеет ли это смысл?
5. Большинство, но не все, файлы журналов являются текстовыми файлами. Попробуйте прочитать файл **lastlog** в каталоге **/var/log** как текстовый файл. Что просходит? Попробуйте команду **lastlog**. Вы сейчас читаете содержимое файла **/var/log/lastlog**? Можете ли вы подтвердить это на соответствующей странице руководства?

Просмотр записей журнала журнала systemd

Помимо инициализации системы и управления сервисами, **systemd** также реализует мощную систему регистрации. По умолчанию журналы хранятся в кольцевом буфере в двоичном формате в каталоге **/run/log/journal**, и они не сохраняют перезагрузку системы. В главе 5 мы кратко представили **journalctl** и объяснили, как включить постоянное ведение журнала. В этом разделе мы рассмотрим некоторые основные функции команды **journalctl** и покажем, как выполнять расширенный поиск.

Одним из основных преимуществ системного журнала перед **rsyslog** является то, что он может хранить не только сообщения ядра и системного журнала, но также любые другие выходные данные, которые сервисы отправляют на стандартный вывод или стандартную ошибку. Вам не нужно знать, куда демон отправляет свои журналы, потому что все захвачено **systemd** и зарегистрировано в журнале. Журнал индексируется так, чтобы его можно было легко найти с помощью различных параметров.

По умолчанию команда **journalctl** показывает все сообщения в журнале в постраничном формате в хронологическом порядке. Он отображает сообщения об ошибках **err** и серьезных **crit** жирным шрифтом, а также показывает строки предупреждений **alert** и аварийных **emerg** сообщений красным цветом. Полезный командный ключ - это **-f**, который работает аналогично команде **tail -f**, отображая последние 10 записей журнала и непрерывно печатая все новые записи журнала по мере их добавления в журнал.

Вы можете отфильтровать вывод **journalctl** несколькими способами. Вы можете использовать ключ **-p** для отображения сообщений, приоритет которых такой же или выше указанного. Например, следующая команда показывает только записи с приоритетом **err** или выше:

 # journalctl -p err

Командные ключи **--since** и **--until** могут ограничить вывод заданным диапазоном времени. Следующие примеры должны быть понятны:

 # journalctl --since yesterday
 # journalctl --until "2015-03-28 11:59:59"
 # journalctl --since 04:00 --until 10:59

Вы также можете отфильтровать вывод, просмотрев самые последние записи журнала с помощью параметра **-n**. Например, вы можете запустить следующую команду, чтобы показать последние 20 строк в журнале:

 # journalctl -n 20

Но это еще не все. Каждая запись в журнале **systemd** имеет набор метаданных, которые вы можете отобразить с помощью параметра **-o verbose**. На **рисунке 9-10** показано, как выглядит запись журнала при включении подробного вывода.

Команда **journalctl** может фильтровать вывод, используя любое из полей, перечисленных на **рис. 9-10**.

РИСУНОК 9-10. Журнальная запись с метаданными

```
Sun 2015-03-08 22:01:03.074289 GMT [s=6b28fd9c29aa4618ba499fc63109198e;i=31c97;b
=7afe9ed7d1c04a00ad954c9cb7cbff99;m=220188bce86;t=5115ade9c68dd;x=825a08f554ea90
65]
    _TRANSPORT=syslog
    PRIORITY=3
    SYSLOG_FACILITY=3
    SYSLOG_IDENTIFIER=nslcd
    SYSLOG_PID=11103
    _PID=11103
    _UID=65
    _GID=55
    _COMM=nslcd
    _EXE=/usr/sbin/nslcd
    _CMDLINE=/usr/sbin/nslcd
    _CAP_EFFECTIVE=0
    _SYSTEMD_CGROUP=/system.slice/nslcd.service
    _SYSTEMD_UNIT=nslcd.service
    _SYSTEMD_SLICE=system.slice
    _SELINUX_CONTEXT=system_u:system_r:nslcd_t:s0
    _BOOT_ID=7afe9ed7d1c04a00ad954c9cb7cbff99
    _MACHINE_ID=b37be8dd26f97ac4ba4a6152f5e92b44
    _HOSTNAME=server1.example.com
    MESSAGE=[7721c9] <group/member="alex"> no available LDAP server found: Serve
r is unavailable: Transport endpoint is not connected
    _SOURCE_REALTIME_TIMESTAMP=1426456863074289
```

Например, следующая команда показывает все записи журнала, связанные с идентификатором пользователя **1000**:

 # journalctl _UID=1000

Аналогично, в следующем примере отображаются все записи журнала, связанные с демоном **nslcd**:

 # journalctl _COMM=nslcd

Вы также можете указать несколько условий в одной строке. По мере того, как вы будете больше практиковаться с командой **journalctl**, вы обнаружите, что журнал **systemd** очень надежен и гибок, и его можно запрашивать, используя множество различных опций.

SCENARIO & SOLUTIONA	
В файле **crontab** не выполняется.	Проверьте **/var/log/cron**. Убедитесь, что у сценария есть разрешения на выполнение.
Обычные пользователи не могут получить доступ к команде **crontab** или поддержку **at**.	Просмотрите файлы **cron.allow и cron.deny** в каталоге **/etc**, чтобы убедиться, что пользователи могут запускать команду **crontab**. Аналогично, чтобы предоставить пользователям разрешение на планирование на рабочих местах, просмотрите файлы **at.allow и at.deny**.
Файлы журнала не содержат достаточной информации.	Верните **/etc/rsyslog.conf**. Сосредоточьтесь на желаемом объекте, таком как **authpriv, mail** или **cron**, и измените приоритет, чтобы включить более подробную информацию. Найдите записи в журнале **systemd**.

РЕЗЮМЕ СЕРТИФИКАЦИИ

RHEL 7 включает в себя множество команд системного администрирования, которые могут помочь вам контролировать и управлять ресурсами, используемыми в системе. Эти команды включают **ps, top, kill, nice, и renice**. Кроме того, с помощью правильных команд вы можете создавать архивы. Тем не менее, специальные параметры команды необходимы для резервного копирования файлов со специальными атрибутами, такими как основанный на **ACL** и **SELinux**.

Демоны **cron и at** могут помочь вам управлять тем, какие задания выполняются в системе по график. При наличии связанных файлов конфигурации доступ к этим демонам может быть ограничен пользователям. Хотя файлы конфигурации **cron** следуют определенному формату, задокументированному в **/etc/crontab,** эти директивы конфигурации были интегрированы с системой **anacron**, которая поддерживает управление заданиями в системах, которые регулярно отключаются.

RHEL 7 включает в себя две системы ведения журнала - журнал **systemd** и демон **rsyslog** которые настроены в основном для локальных систем в файле **/etc/rsyslog.conf**. Записи журнала обычно собирается **systemd** в каталоге **/run/log/journal**, тогда как **rsyslog** хранит журнал файлы постоянно находятся в каталоге **/var/log**. Демон **rsyslog** также поддерживает создание сервера журналирования, который может собирать информацию файла журнала из множества систем.

ПАРУ МИНУТ ПОВТОРЕНИЯ

Вот некоторые из ключевых моментов целей сертификации в главе 9.

Элементарные команды системного администрирования

- Команда **ps** может идентифицировать запущенные в данный момент процессы.
- Команда **top** запускает браузер задач, который может идентифицировать процессы, использующие чрезмерные ресурсы в системе.
- **sar** и связанные команды предоставляют отчеты о работе системы.
- Команда **iostat** может предоставить статистику процессора и устройства хранения.
- Команды **nice и renice** могут использоваться для перераспределения процессов.
- Команды **kill и killall** можно использовать для остановки запущенных в данный момент процессов и даже демоны с различными сигналами.
- Архивы могут быть созданы, извлечены и сжаты с помощью команд **gzip, bzip2, tar** и **star**.

Автоматизация системного администрирования: cron и at

- Система **cron** позволяет пользователям планировать задания так, чтобы они выполнялись с заданными интервалами.
- Система **at** позволяет пользователям настраивать задания для запуска один раз в запланированное время.
- Команда **crontab** используется для работы с файлами **cron**. Используйте **crontab -e** для редактирования, **crontab -l** посмотреть список и **crontab -r** для удаления файлов **cron**.
- Файлы **/etc/cron.allow** и **/etc/cron.deny** используются для управления доступом к **cron** планировщик работы; файлы **/etc/at.allow** и **/etc/at.deny** используются для управления доступом к работе планировщика аналогичным образом.

Анализ локальных файлов журнала

- **Red Hat Enterprise Linux** включает демон **rsyslog**, который контролирует систему на предмет сообщения ядра, а также другие действия процесса, как настроено в **/etc/rsyslog.conf**.
- Вы можете использовать файлы журнала, сгенерированные в каталоге **/var/log**, для отслеживания действий в системе.
- Другие файлы журнала можно создавать и настраивать с помощью файлов конфигурации службы.
- Файлы журналов можно регулярно перемещать, как это указано в файле **/etc/logrotate.conf**.
- Журнал **systemd** регистрирует все загрузочные сообщения, сообщения ядра и службы в кольцевом буфере внутри каталога **/run/log/journal**.
- Команда **journalctl** используется для отображения и фильтрации записей журнала.

САМОПРОВЕРКА

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

Элементарные команды системного администрирования

1. Какая команда идентифицирует все запущенные процессы в текущей терминальной консоли?

_____

2. Какой наивысший приоритетный номер вы можете установить для процесса командой nice?
_____

3. Какую опцию команды tar можно использовать для архивирования файлов существующего каталога при сохранении его Контексты SELinux?
_____

4. Вы хотите создать архив каталога /etc. Какую команду нужно запустить, чтобы создать сжатый bzip2 архив этой директории? Предположим, что архив называется /tmp/etc.tar.bz2
_____

Автоматизация системного администрирования: cron и at

5. Вы хотите запланировать работу по обслуживанию, maintenance.pl, для запуска из вашего домашнего каталога на первое число каждого месяца в 4:00 утра. Вы запускаете команду crontab -e, чтобы открыть свой личный файл crontab. Предположим, вы добавили соответствующие директивы PATH и SHELL. Какая директива вы бы добавили, чтобы запустить указанное задание в указанное время?
_____

6. Предположим, вы видите следующую запись в выводе команды crontab -l:

42 4 1 * * root run-parts /etc/cron.monthly

Когда в следующий раз Linux запустит задания в каталоге /etc/cron.monthly?
_____

7. Если пользователи tim и stephanie указаны в файлах /etc/cron.allow и /etc/cron.deny,
а пользователи donna и elizabeth перечислены только в файле /etc/cron.allow, который из этих пользователей qразрешено запускать команду crontab -e?
_____

8. Какой файл используется для настройки ротации файла журнала?
_____

Анализ локальных файлов журнала

9. Какая запись в файле /etc/rsyslog.conf будет уведомлять вошедших в систему пользователей при возникновении критической проблем с ядром?
_____

10. В каталоге /var/log есть несколько файлов, связанных с тем, что произошло во время установки процесса. Какое первое слово разделяют имена этих файлов журнала?
_____

11. Какая команда отображает все записи журнала systemd с приоритетом, равным alert или выше?
_____

12. Как вы можете показать записи журнала systemd, относящиеся к демону httpd, зарегистрированному с 16го числа март 2015?
_____

LAB ВОПРОСЫ

Некоторые из этих лабораторных работ включают упражнения, которые могут серьезно повлиять на систему. Вы должны сделать эти упражнения только на тестовых машинах. Вторая лаборатория главы 1 устанавливает KVM для этой цели.

Red Hat представляет свои экзамены в электронном виде. По этой причине лаборатории для этой главы доступны на DVD, который сопровождает книгу, в подкаталоге Chapter9 /. Они доступны в .doc, .html, и форматы .txt. Если вы еще не настроили RHEL 7 в системе, обратитесь к первому лабораторному заданию главы 2 для Инструкция по установке. Тем не менее, ответы для каждой лаборатории следуют за ответами самопроверки для заполнения пустые вопросы.

Лабораторная работа к главе 9

Labs

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 9. Большинство лабораторных работ для этой главы просты и требуют очень небольшого количества команд или изменений в одном или двух файлах конфигурации.

Лаборатория 1

Как пользователь **root**, создайте задания **cron**, которые изменяют сообщение для входа пользователей в текстовой консоли. Для этого вы захотите изменить содержимое **/etc/motd**. Убедитесь, что люди, которые входят в систему в разное время, получают соответствующие сообщения:

- Если пользователи входят в систему между 7 утра. и 1 час дня, создайте сообщение для входа «Coffee time!»
- Если пользователи входят между 1 вечера. и в 6 часов вечера создайте сообщение для входа «Хотите мороженого?»
- Если пользователи входят между 6 вечера. и 7 часов утра создайте сообщение для входа «Разве вы не делаете что-то еще?»

Лаборатория 2

В этой лабораторной работе вы настроите на рабочем месте пользователя с правами администратора для сохранения списка установленных RPM-файлов в файле /root/rpms.txt. Это задание будет выполнено один раз, через 24 часа. Если вы хотите проверить свою работу, настройте секунду на работе с теми же командами, чтобы начать через пять минут.

Лаборатория 3

В этой лабораторной работе вы создадите задание **cron** для резервного копирования каталога **/etc** каждую субботу в 2:05 утра. Вы должны убедиться, что контексты **SELinux** сохранены. Вы не можете редактировать корневой **crontab**. Резервная копия должна быть сохранена в формате архива **gzipped-tar** в каталоге **/tmp**, используя имя файла **etc-backup-MMDD.tar.gz**. Извлеките один из созданных файлов резервных копий, чтобы убедиться, что контексты **SELinux** сохранены.

Лаборатория 4

В этой лабораторной работе вы узнаете больше о значении нескольких разных файлов журналов. При подготовке используйте неправильный пароль для входа в обычную учетную запись. Затем выполните следующие действия:

1. Перейдите в **/var/log** от имени пользователя **root**. Все файлы, перечисленные в этой лабораторной работе, находятся в этом каталоге **/var/log**.
2. Изучите содержимое файлов журнала **anaconda. ***.
3. Запустите команду **utmpdump btmp**. Вы видите попытку входа? Можете ли вы сказать, если это удалось?
4. Просмотрите содержимое файла журнала **cron**. Прокрутите это. Если ваш компьютер был включен некоторое время, большая часть того, что вы увидите, будет основана на команде **run-parts /etc/cron.hourly**. В качестве альтернативы, если вы перезагрузите компьютер, вы увидите сообщения, связанные со службой **Anacron**.
5. Просмотрите содержимое файла журнала **dmesg**. Сравните его начало с началом файла **anaconda/syslog**. Какой из них включает загруженное в настоящее время ядро?
6. Перейдите к нижней части файла **dmesg**. Можете ли вы определить объем пространства подкачки? Можете ли вы идентифицировать один или несколько разделов с файловой системой **XFS** по умолчанию?
7. Просмотрите файл журнала почтового журнала. Если этот файл короткий, возможно, существует более старый файл **maillog- *;** если так, рассмотрите это также. Вы видите какие-либо журналы, связанные с почтовыми сообщениями?
8. Просмотрите защищенный файл журнала. Перейдите в конец файла. Вы видите сообщение, связанное с неудачным входом в систему?
9. Просмотрите файл **Xorg.0.log** в каталоге **/var/log**. Вы видите какие-либо сообщения, связанные с графическим экраном в файле? Как это работает, если вы не настроили графический интерфейс во время процесса установки?

ОТВЕТЫ НА САМОПРОВЕРКУ

Элементарные команды системного администрирования

1. Это небольшой вопрос, потому что команда **ps** сама по себе идентифицирует любой запущенный в данный момент процессы в текущей консоли.
2. Наивысший приоритетный номер, который можно использовать с командой **nice**, равен **-20**. Помните, приоритет цифры для процессов являются нелогичными.
3. Командой **tar**, которая сохраняет контексты **SELinux** в архиве, является **--selinux.**
4. Команда, которая создает сжатый архив **bzip2** из каталога **/etc**:

 # tar cvfj /tmp/etc.tar.bz2 /etc

Автоматизация системного администрирования: cron и at

5. Директива, которая запускает скрипт maintenance.pl из домашнего каталога в указанное время:

0 4 1 * * ~ / maintenance.pl

6. Исходя из отмеченной записи в **/etc/crontab**, в следующий раз, когда **Linux** запустит задания в каталог **/etc/cron.monthly** находится в первом из следующего месяца, в 4:42 утра.
7. Когда имена пользователей существуют в файлах **/etc/cron.allow и /etc/cron.deny**, пользователи перечисляются в **/etc/cron.deny** игнорируются. Таким образом, всем четырем перечисленным пользователям разрешено запускать различные команды **crontab**.

8. Файл конфигурации, связанный с ротацией файлов журнала с течением времени, - **/etc/logrotate.conf**. Дополнительные сервисные файлы конфигурации могут быть созданы в каталоге **/etc/logrotate.d**

Анализ локальных файлов журнала

9. В файле **/etc/rsyslog.conf** есть комментарий, который соответствует требованиям вопроса. Просто активируйте его и измените приоритет на **crit**, чтобы уведомлять вас (и всех) всякий раз, когда возникает серьезная проблема с журналами ядра:

   kern.crit /dev/console

   Конечно, это означает, что существуют другие приемлемые способы удовлетворения требований данного вопроса.
10. Файлы журнала в **/var/log**, наиболее соответствующие процессу установки, начинаются с **anaconda**.
11. Команда, которая отображает все записи журнала **systemd** с приоритетом, равным **alert** или выше, имеет вид **journalctl -p alert**.
12. Показать все записи журнала **systemd**, относящиеся к демону **httpd** и зарегистрированные с 16го числа В марте 2015 года выполните команду **journalctl _COMM = httpd --since 2015-03-16**.

ОТВЕТЫ ЛАБАРАТОРНОЙ РАБОТЫ

Лаборатория 1

Один из способов изменить входящие сообщения, как отмечено, состоит в следующих шагах (есть по крайней мере один другой метод, связанный с каталогом **/etc/cron.d**):

1. Войдите в систему как пользователь **root**.
2. Запустите команду **crontab -e**.
3. Добавьте соответствующие переменные среды, по крайней мере, следующие:

   SHELL=/bin/bash

4. Добавьте следующие команды в файл, чтобы перезаписать **/etc/motd** в соответствующее время:

   0 7 * * * /bin/echo 'Coffee time' > /etc/motd
   0 13 * * * /bin/echo 'Want some ice cream?' > /etc/motd
   0 18 * * * /bin/echo 'Shouldn\'t you be doing something else?' > /etc/motd

5. Сохраните файл. Пока активен демон **cron** (по умолчанию), следующий пользователь, который входит в систему в консоли после одного из указанных раз должно появиться сообщение при успешном входе в систему. Если вы хотите проверить результат немедленно, команда date может помочь. Например, команда

   # date 06120659

   устанавливает дату 12 июня в 6:59, незадолго до того, как демон **cron** должен выполнить первую команда в списке. (Конечно, вы хотите заменить сегодняшнюю дату и подождать одну минуту перед входом в эту систему с другой консоли.)

Лаборатория 2

Чтобы настроить работу **at** для запуска через 5 минут, начните с команды **at**. Это приведет вас к приглашению **at>**.

Установленные в настоящее время **RPM** отображаются в выходных данных команды **rpm -qa**. Так как нет **PATH** определенный в приглашении **at>,** вы должны указать полный путь. Так что один из способов создать список в настоящее время установленные **RPM** к файлу **/root/rpms.txt** в одноразовом задании, которое начинается через пять минут после следующие команды:

 # at now + 5 min
at> /bin/rpm -qa > /root/rpms.txt
at> Ctrl+d
#

В течение пяти минут вы должны увидеть файл **rpms.txt** в домашнем каталоге пользователя **root, /root**. Если пять минут слишком долго ждать (как это может быть во время экзамена RHCSA), перейдите к лабораторной работе 3 и вернемся к этой проблеме позже. Не забудьте настроить другую работу, которая будет запущена в течение 24 часов.

Лаборатория 3

Один из способов настройки задания **cron**, указанного в лабораторных требованиях, подробно описан здесь:

1. Войдите в систему как пользователь **root**.
2. Требования лаборатории не позволяют использовать команду **crontab -e** для редактирования корневого **crontab**.
   файл. Следовательно, создайте системный **crontab** в каталоге **/etc/cron.d**, используя следующую команду:

   # cat> /etc/cron.d/etc-backup << EOF

3. Введите следующую строку для настройки задания **cron**:

   5 2 * * 6 root /usr/bin/tar --selinux -czf /tmp/etc-backup-\$(/bin/date +\%m\%d).tar.gz /etc > /dev/null
4. Не забудьте экранировать символы **%** в записи **crontab**; в противном случае они будут интерпретироваться как новые строки.
5. Введите последовательность **EOF**:

   EOF

6. Чтобы протестировать задание, измените запись **crontab**, чтобы она запускалась через несколько минут. Затем измените каталог в **/tmp** и распакуйте сгенерированный архив с помощью следующей команды:

   # tar --selinux -xzf etc-backup - $ (дата +% m% d) .tar.gz

6. Подтвердите, что контексты **SELinux** были сохранены, выполнив следующую команду:

   # ls -lRZ / tmp/etc

Лаборатория 4

В этой лаборатории нет секретных решений; цель состоит в том, чтобы заставить вас просмотреть содержимое файлов ключей понять что там должно быть.

Когда вы просматриваете файлы **anaconda.*** В **/var/log** и сравниваете их с другими файлами, вы можете получить некоторое представление о том, как диагностировать проблемы установки. В следующих главах вы изучите некоторые из файлы журналов, связанные с конкретными услугами; многие находятся в подкаталогах, таких как **/var/log/samba** и **/var/log/httpd.**

Неудачный вход в систему должен быть легко виден в файле **/var/log/secure**. Вы можете получить подсказки в вывод команды **utmpdump btmp.**

Когда вы просмотрите файл **/var/log/cron**, вы увидите, когда выполнялись стандартные задания **cron**. Большинство файла должено быть заполнен (по умолчанию) стандартной почасовой работой, **run-parts /etc/cron.hourly**, из **/etc/cron.d/0hourly** файла конфигурации. Если вы перезагрузились, вы можете увидеть службу **anacron**, и вы должен быть в состоянии искать задачу с тем же именем.

Хотя **/var/log/dmesg** включает загруженное в данный момент ядро, оно может быть тем же ядром, что и связанное с **/var/log/anaconda/syslog**, если вы не обновили ядра. В конце **/var/log/dmesg** вы может найти файловые системы, смонтированные в формате **XFS**, а также смонтированные в настоящий момент разделы подкачки. Например, в следующем списке перечислены разделы виртуального диска на основе **KVM**:

XFS (vda1): Mounting Filesystem
Adding 1023996k swap on /dev/mapper/rhel-swap.
Priority:-1 extents:1 across:1023996k
XFS (vda1): Ending clean mount
SELinux: initialized (dev vda1, type xfs), uses xattr

Как вы, надеюсь, обнаружили, файл **/var/log/maillog** не содержит никакой информации о почте клиента, только серверы.

Red Hat включила инструмент настройки GUI в RHEL 7. Автоматическая настройка для Аппаратная графика теперь достаточно надежна, но если у вас возникнут какие-либо проблемы, вы можете посмотреть в **/var/log/Xorg.0.log**.