

Логирование в Linux

Цель и назначения логирования

В ходе работы приложения операционной системы могут создавать разные типы сообщений, которые записываются в различные журналы. Для создания, хранения и повторного использования таких сообщений (логирования) операционная система *Linux* использует механизм логирования (создания и ведения журналов) по средствам набора специально предназначенных конфигурационных файлов, команд, программ (утилит) и сервисов, также называемых даемон (программ, работающих в фоновом режиме без контакта с пользователем).

Все программы *Linux* ведут лог путем отправки сообщений об ошибках или своем состоянии с помощью записывая все сообщения в файл, который будет находиться в каталоге */var/log/*. Например, общие сообщения ядра и программ сохраняются в */var/log/messages*. В *CentOs* в данном каталоге можно найти следующие журналы:

- *wtm*
- *utmp*
- *dmesg*
- *messages*
- *maillog* или *mail.log*
- *spooler*
- *auth.log* или *secure*

Просмотрите каталог */var/log/*.

Для просмотра списка журналов, находящихся в данном каталоге, используйте команду *ls -l /var/log*.

В системе *CentOS* это выглядит приблизительно так:

```
[r@TestLinux ~]# ls -l /var/log
total 1472
-rw-----. 1 root root 4524 Nov 15 16:04 anaconda.ifcfg.log
-rw-----. 1 root root 59041 Nov 15 16:04 anaconda.log
-rw-----. 1 root root 42763 Nov 15 16:04 anaconda.program.log
-rw-----. 1 root root 299910 Nov 15 16:04 anaconda.storage.log
-rw-----. 1 root root 40669 Nov 15 16:04 anaconda.syslog
-rw-----. 1 root root 57061 Nov 15 16:04 anaconda.xlog
-rw-----. 1 root root 1829 Nov 15 16:04 anaconda.yum.log
drwxr-x---. 2 root root 4096 Nov 15 16:11 audit
-rw-r--r--. 1 root root 2252 Dec 9 10:27 boot.log
-rw-----. 1 root utmp 384 Dec 9 10:31 bttmp
-rw-----. 1 root utmp 1920 Nov 28 09:28 bttmp-20131202
drwxr-xr-x. 2 root root 4096 Nov 29 15:47 ConsoleKit
-rw-----. 1 root root 2288 Dec 9 11:01 cron
-rw-----. 1 root root 8809 Dec 2 17:09 cron-20131202
-rw-r--r--. 1 root root 165665 Nov 15 16:04 dracut.log
-rw-r--r--. 1 root root 146876 Dec 9 10:44 lastlog
```

```

-rw----- 1 root root 950 Dec 9 10:27 maillog
-rw-----. 1 root root 4609 Dec 2 17:00 maillog-20131202
-rw----- 1 root root 123174 Dec 9 10:27 messages
-rw-----. 1 root root 458481 Dec 2 17:00 messages-20131202
-rw----- 1 root root 2644 Dec 9 10:44 secure
-rw-----. 1 root root 15984 Dec 2 17:00 secure-20131202
-rw-----. 1 root root 0 Nov 15 16:02 tallylog
-rw-rw-r--. 1 root utmp 89856 Dec 9 10:44 wtmp
-rw----- 1 root root 3778 Dec 6 16:48 yum.log

```

Попробуйте посмотреть файл `messages` в каталоге `/var/log`.

Большинство журналов могут быть просмотрены при помощи команд `cat`, `head`, `tail` а также, например, таких редакторов как `vi`.

Следует отметить, что содержимое файлов `wtmp` и `utmp` (отслеживают пользователей, вошедших и покинувших систему) нельзя читать с помощью простой команды «`cat`» - данные файлы представлены в бинарном формате.

Для просмотра содержимого `/var/log/wtmp` используется: «`last`», для просмотра содержимого `/var/log/btmp` используется: «`lastb`» для просмотра содержимого `/var/run/utmp` используется: «`who`».

Прочитайте последние 20 записей в `messages`

```
tail -n 20 /var/log/messages
```

Следует также отметить команду `tail -f /var/log/messages`, позволяющую просматривать лог в реальном времени.

Одной из важных функций логирования является слежение за входом и выходом из системы. Так, например, чтобы узнать, кто в текущий момент находится на сервере, нужно использовать команду «`who`». Данная команда извлекает информацию из `/var/run/utmp`.

Запустите команду «`who`».

Пример работы команды «`who`» в CentOS:

```

[r@Linux ~]# who
root    tty1      2013-12-09 10:44
root    pts/0      2013-12-09 10:29 (10.0.2.2)
sysadmin pts/1      2013-12-09 10:31 (10.0.2.2)
joeblog pts/2      2013-12-09 10:39 (10.0.2.2)

```

Чтобы получить историю входов конкретного пользователя используйте команду

«`last | grep`» выводит историю входа пользователей:

```

[r@Linux ~]# last | grep username
username pts/1    10.0.2.2      Mon Dec 9 10:31 still logged in
username pts/0    10.0.2.2      Fri Nov 29 15:42 - crash (00:01)

```

```

username pts/0      10.0.2.2      Thu Nov 28 17:06 - 17:13 (00:06)
username pts/0      10.0.2.2      Thu Nov 28 16:17 - 17:05 (00:48)
username pts/0      10.0.2.2      Thu Nov 28 09:29 - crash (06:04)
username pts/0      10.0.2.2      Wed Nov 27 16:37 - down (00:29)
username ttyl       Wed Nov 27 14:05 - down (00:36)
username ttyl       Wed Nov 27 13:49 - 14:04 (00:15)

```

В данном примере нужно было получить историю входа пользователя *username*.
Как можно видеть, было пару случаев, когда он приводил к сбою системы (*crash*).

Чтобы узнать время последней перезагрузки системы, используйте следующую команду: *last reboot*

Результат имеет примерно такой вид:

```

reboot system boot 2.6.32-358.el6.x Mon Dec 9 10:27 - 10:47 (00:19)
reboot system boot 2.6.32-358.el6.x Fri Dec 6 16:37 - 10:47 (2+18:10)
reboot system boot 2.6.32-358.el6.x Fri Dec 6 16:28 - 16:36 (00:08) reboot system
boot 2.6.32-358.el6.x Fri Dec 6 11:06 - 16:36 (05:29)
reboot system boot 2.6.32-358.el6.x Mon Dec 2 17:00 - 16:36 (3+23:36)
reboot system boot 2.6.32-358.el6.x Fri Nov 29 16:01 - 16:36 (7+00:34)
reboot system boot 2.6.32-358.el6.x Fri Nov 29 15:43 - 16:36 (7+00:53)
...
...
wtmp begins Fri Nov 15 16:11:54 2013

```

Чтобы узнать время последнего входа в систему, используйте *lastlog*:

Результат выглядит примерно так:

<i>Username</i>	<i>Port</i>	<i>From</i>	<i>Latest</i>
<i>root</i>	<i>ttyl</i>		<i>Mon Dec 9 10:44:30 +1100 2013</i>
<i>bin</i>			<i>**Never logged in**</i>
<i>daemon</i>			<i>**Never logged in**</i>
<i>adm</i>			<i>**Never logged in**</i>
<i>lp</i>			<i>**Never logged in**</i>
<i>sync</i>			<i>**Never logged in**</i>
<i>shutdown</i>			<i>**Never logged in**</i>
<i>halt</i>			<i>**Never logged in**</i>
<i>mail</i>			<i>**Never logged in**</i>
<i>uucp</i>			<i>**Never logged in**</i>
<i>operator</i>			<i>**Never logged in**</i>
<i>games</i>			<i>**Never logged in**</i>
<i>gopher</i>			<i>**Never logged in**</i>
<i>ftp</i>			<i>**Never logged in**</i>
<i>nobody</i>			<i>**Never logged in**</i>
<i>vcsa</i>			<i>**Never logged in**</i>
<i>saslauth</i>			<i>**Never logged in**</i>
<i>postfix</i>			<i>**Never logged in**</i>
<i>sshd</i>			<i>**Never logged in**</i>
<i>sysadmin</i>	<i>pts/1</i>	<i>10.0.2.2</i>	<i>Mon Dec 9 10:31:50 +1100 2013</i>
<i>dbus</i>			<i>**Never logged in**</i>
<i>joeblog</i>	<i>pts/2</i>	<i>10.0.2.2</i>	<i>Mon Dec 9 10:39:24 +1100 2013</i>

Функции и назначение *rsyslog*

При контроле конфигурации операционной системы, а также поиске и исправлении ошибок важно контролировать, как и куда сохраняются файлы журналов. На протяжении многих лет в операционной системе *Linux* используется сервис *Syslog* для управления логами. В современных версиях применяется его модификация - *rsyslog*.

Сервис *rsyslog* является центром механизма журналирования. Данный сервис отвечает за прослушивание зарегистрированных сообщений различных частей системы *Linux* и маршрутизацию сообщения к соответствующему журналу в каталоге */var/log*. Он также может передавать зарегистрированные сообщения другому серверу *Linux*.

Вот основные возможности *rsyslog*:

- Многопоточность;
- *TCP, SSL, TLS, RELP*;
- Поддержка *MySQL, PostgreSQL, Oracle*;
- Фильтрация журналов;
- Полностью настраиваемый формат вывода.

Все настройки *rsyslog* находятся в файле */etc/rsyslog.conf* и других конфигурационных файлах из */etc/rsyslog.d*. Вы можете посмотреть существуют ли у вас эти файлы выполнив:

```
ls /etc/rsys*  
rsyslog.conf rsyslog.d/
```

Основной конфигурационный файл - */etc/rsyslog.conf*, в нем подключены все файлы из папки *rsyslog.d* с помощью директивы *IncludeConfig* в самом начале файла:

```
IncludeConfig /etc/rsyslog.d/*.conf
```

В этих файлах из *rsyslog.d* могут содержаться дополнительные настройки, например, аутентификация на *Rsyslog* сервере. В главном конфигурационном файле содержится очень много полезных настроек. Обычно он обеспечивает управление локальными логами по умолчанию (говорит где находится то или иное сообщение).

Посмотрите файл rsyslog.conf

Пример:

```
# rsyslog v5 configuration file  
# Include all config files in /etc/rsyslog.d/  
IncludeConfig /etc/rsyslog.d/*.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                                         *
# 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

```

Информация в файле `rsyslog.conf` имеет вид серии строк, состоящих из двух частей - селектор и действие (selector и action), например:

```
mail.*      -/var/log/maillog
```

Селектор (`mail.*`) указывает на источник и приоритет сообщения; действие (`/var/log/maillog`) говорит, что нужно сделать с данным сообщением. Селектор также разделен на 2 части символом точки (.). Часть перед символом точки называется объектом (источник сообщения), а часть за символом называется приоритетом (степень важности сообщения).

Комбинация объекта-приоритета и действия говорит `rsyslog`, что делать, если сообщение соответствует указанным параметрам.

Система *Linux* распознает следующие объекты:

- *auth* or *authpriv*: Сообщения, от сервисов авторизации и безопасности;
- *kern*: сообщения ядра *Linux*;
- *mail*: сообщения подсистемы почты;
- *cron*: сообщения даемона *Cron*;
- *daemon*: сообщения от сервисов;
- *news*: сообщения подсистемы новостей сети;
- *lpr*: сообщения, связанные с печатью;
- *user*: сообщения пользовательских программ;
- *local0* до *local7*: Зарезервировано для локального использования.

Система *Linux* допускает следующие приоритеты:

- *Debug*: Отладочная информация от программ;
- *info*: простое информационное сообщение – никакого вмешательства не требуется;
- *notice*: состояние, которое может потребовать внимания;
- *warn*: Предупреждение;
- *err*: ошибка;
- *crit*: критическое состояние;
- *alert*: состояние, требующее немедленного вмешательства;
- *emerg*: аварийное состояние.

Следует отметить, что каждый приоритет подразумевает свой уровень подробности логирования.

Рассмотрим, например, следующую строку из файла:

```
cron.* /var/log/cron
```

Данная строчка означает, что rsyslog сохранять все сообщения, приходящие от сервиса *cron*, в файле */var/log/cron*. Звездочка (*) после точки значит, что зарегистрированы будут сообщения всех приоритетов. Аналогичным образом, если объект был определен звездочкой, это объединяет все источники (например **.info*).

Таким образом запись

```
mail.warn /var/log/mail.warn
```

будут регистрировать все сообщения с таким же или высшим, чем *warn*, приоритетом и пропускать все остальное (только сообщения с приоритетом *err*, *crit*, *alert* и *emerg* будут внесены в файл).

Запись

```
mail.=info /var/log/mail.info
```

имеет знак равенности (=) после точки, что указывает регистрировать только сообщения с указанным приоритетом. То есть, если нужно регистрировать только сообщения от почтовой подсистемы с приоритетом *info*. Если нужно регистрировать все сообщения почтовой подсистемы, кроме сообщений с приоритетом *info*, строка будет выглядеть так:

```
mail.!info /var/log/mail.info
```

или так:

```
mail.!=info /var/log/mail.info
```

В первом случае файл *mail.info* содержал бы все сообщения с приоритетом ниже *info*. Во втором случае он содержал бы все сообщения с приоритетом выше *info*.

Также есть возможность указания нескольких объектов или селекторов в одной строке, при этом они должны будут разделены запятыми:

```
Uucp,news.crit /var/log/spooler
```

Конфигурации для rsyslog могут исходить также от других пользовательских файлов. Эти файлы пользовательских конфигураций, как правило, расположены в разных каталогах в */etc/rsyslog.d*. Файл *rsyslog.conf* включает эти каталоги, используя директиву «*\$IncludeConfig*». Для этого в файле *rsyslog* имеется строчка

```
Include all config files in etc.rsyslog.d/  
Include(file="/etc/rsyslog.d/*.conf" mode="optional")
```

Сервис rsyslog может не только хранить зарегистрированные сообщения локально, но и передавать их по сети на другие серверы Linux, а также действовать как репозиторий для других систем. Прослушивание сообщений происходит через UDP-порт 514. В приведенном ниже примере он пересылает критические сообщения ядра на сервер под названием «texas»:

```
kern.crit      @texas
```

Задание: Попробуйте сами создать сообщение.

Для этого нужно будет сделать следующее:

- Задать спецификацию в файле /etc/rsyslog.conf;
- Перезапустить демон rsyslog;
- Проверить конфигурацию с помощью утилиты «logger».

Редактирование файла может быть выполнено при помощи команды

```
vi /etc/rsyslog.conf
```

При использовании редактора vi для перехода в режим редактирования с места расположения курсора нажимайте «i» для выхода из данного режима используйте «Esc». Для перехода в режим ввода команд используйте «:», команда сохранения «w» (после этого нажмите «Enter» – должно появиться сообщение о том сколько строк записано), для выхода используется команда q.

Внесите в файл rsyslog.conf две строки:

```
# New lines added for testing log message generation
local4.crit      /var/log/local4crit.log
local4.=info     /var/log/local4info.log
```

Данные команды для rsyslog исходят от объекта local4 и имеют разные приоритеты.

Затем можно перезапустить сервис, чтобы обновить данные файла:

```
/ systemctl restart rsyslog
```

Для того чтобы отправить сообщение объекту local4 нужно вызвать приложение logger

Создайте следующие сообщения:

```
logger -p local4.info " This is a info message from local 4"
```

Проверьте новые данные в каталоге /var/log

```
TestLinux root: This is a info message from local 4
```

Ротация журналов

Со временем журналы становятся больше, поскольку в них появляется новая информация. Это создает потенциальную проблему производительности. Кроме того, управление файлами становится затруднительным. Для решения этой проблемы *Linux* использует понятие «ротации» журналов (используется вместо их очистки или удаления). При ротации создается новый каталог, а старый переименуется и при необходимости сжимается. Таким образом, журналы имеют несколько старых версий. Эти файлы будут возвращаться в течение определенного периода времени в виде так называемых *backlog*-ов. Как только будет получено определенное количество *backlog*-ов, новая ротация удалит самый старый журнал.

Ротация выполняется при помощи утилиты «*logrotate*». Данная утилита зависит от конфигурационного файла *logrotate.conf*, который находится в */etc*.

Вот что, примерно, находится в данном файле:

```
cat /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
# uncomment this if you want your log files compressed
#compress
# RPM packages drop log rotation information into this directory
include /etc/logrotate.d
# system-specific logs may be configured here
```

По умолчанию журналы ротируются еженедельно, оставляя 4 *backlog*-а. При запуске программы создается новый пустой журнал, а старые при необходимости будут сжаты. Файлы *wtmp* и *btmp* являются исключениями. *wtmp* отслеживает вход в систему, а *btmp* содержит информацию о неудавшихся попытках входа. Эти журнальные файлы ротируются каждый месяц, и ошибки не возвращаются, если можно найти один из предыдущих файлов *wtmp* или *btmp*.

Пользовательские конфигурации ротации журналов содержатся в каталоге «*etc/logrotate.d*». также они включены в *logrotate.conf* с помощью директивы *include*.

Утилиту *logrotate* можно запустить вручную для ротации одного или нескольких файлов. Чтобы это сделать, нужно просто указать соответствующий конфигурационный файл как аргумент.

Задание: Чтобы продемонстрировать, как работает *logrotate* посмотрите содержимое каталога */var/log* (используйте *ls -l /var/log*)

Затем запустите команду *logrotate*:

```
logrotate -fv /etc/logrotate.conf
```

Затем попробуйте проверить новые журнальные файлы почты, безопасности и сообщений:

```
ls -l /var/log/mail*  
ls -l /var/log/messages*  
ls -l /var/log/secure*
```

Как можно видеть, все три новых журнала были созданы. Почтовый журнал и журнал безопасности все еще пусты, но новый журнал сообщений уже содержит некоторые данные.

Утилита *journalctl*

В современных версиях Linux сервис обеспечения работы системы (*systemd*) предоставляет свои инструменты логирования *Journal*. Данная утилита автоматически собирает все системные сообщения (сообщения ядра, различных служб и приложений) с сохранением их в специализированной бинарной базе. Утилита *Journal* может работать как совместно с *rsyslog*, так и полностью заменить его. Одним из достоинств *Journal* по сравнению с *rsyslog* является возможность контроля часовых поясов. Посмотреть текущий пояс можно при помощи запроса

```
timedatectl status
```

поменять при помощи запроса

```
timedatectl set-timezone <часовой пояс>
```

список часовых поясов можно получить по запросу *timedatectl list-timezones*)

Функционал *Journal* представлен набором команд *journalctl*.

Примеры таких команд:

- *--full, -l* - отображать все доступные поля;
- *--lines, -n* - отображать n строк, по умолчанию 10;
- *--boot, -b* - показать сообщения с момента определенной загрузки системы. По умолчанию используется последняя загрузка;
- *--list-boots* - показать список сохраненных загрузок системы;
- *--identifier, -t* - показать сообщения с выбранным идентификатором;
- *--unit, -u* - показать сообщения от выбранного сервиса;
- *--priority, -p* - фильтровать сообщения по их приоритету. Есть восемь уровней приоритета, от 0 до 7;
- *--grep, -g* - фильтрация по тексту сообщения;
- *--system* - выводить только системные сообщения;

- *--user* - выводить только сообщения пользователя;
- *--disk-usage* - вывести общий размер лог файлов на диске;
- *--flush* - перенести все данные из каталога */run/log/journal* в */var/log/journal*;
- *--rotate* - запустить ротацию логов;
- *-f* - выводить новые сообщения в реальном времени, как в команде *tail*;
- *--vacuum-time* - очистить логи, давностью больше указанного периода;
- *--vacuum-size* - очистить логи, чтобы размер хранилища соответствовал указанному.

Утилита *journalctl* позволяет использовать несколько форматов вывода:

- *short* - используется по умолчанию;
- *verbose* - также, как и *short*, только выводится намного больше информации;
- *json* - вывод в формате *json*, одна строка лога в одной строке вывода;
- *json-pretty* - форматированный вывод *json* для более удобного восприятия;
- *cat* - отображать только сообщения, без метаданных.

Чтобы указать нужный формат используйте опцию *-o*. Например:

```
sudo journalctl -o json-pretty
```

Задание: Попробуйте посмотреть логи, просмотренные *rsyslog* при помощи *journalctl*, сделайте выводы о различиях и сходствах данных утилит.