Системни логове

Thursday, August 27, 2020 4:39 PM

Лог файловете са такива файлове, чието съдържание има информация за състоянието на системата.

Тези съобщения ще се отнасят до услугите на ядрото или приложенията.

Има различни логове за различни задачи.

Например има логове, който проследява неуспешните влизания.

Има още един логове, само за cron и така нататък.

Повечето базирани на система D операционни системи имат две системи за вход **Rsyslog** log и **journald**.

Rsyslog е съвместим **sysklogd** и обработва постоянни логове.

Rsyslog или регистрира в текстови файлове в локалната машина, или влиза в мрежата към отдалечен сървър за регистриране, използвайки протоколи TCP или UDP.

JournalD обаче е част от **systemd** не е постоянен по подразбиране, така че логовете на **journald** не оцеляват при рестартиране.

Тъй като **journald** е двоичен и се съхранява само в паметта, много бързо се пише и много бързо се търси, за да се гарантира, че Rsyslog работи.

за да се уверите, че Rsyslog работи.

sudo systemctl status rsyslog

Ако не е включен:

sudo systemctl start rsyslog

systemctl is-enabled rsyslog.service
systemctl is-active rsyslog.service

Default conf file:

/etc/rsyslog.d в Debian based
/etc/rsyslog.conf RHL based

CentOS:

less -N /etc/rsyslog.conf

Това което търсим тук започва на линия 40 при мен. Затова ако се налага скролнете докато го видите. Това са правилата за къде всеки вид данни ще бъдат логнати във.

```
 1 # rsyslog configuration file
 2
 3 # For more information see /usr/share/doc/rsyslog-*/rsyslog_conf.html
 4 # or latest version online at http://www.rsyslog.com/doc/rsyslog_conf.html
 5 # If you experience problems, see http://www.rsyslog.com/doc/troubleshoot.html
 6
 7 #### MODULES ####
 8
 9 module(load="imuxsock"    # provides support for local system logging (e.g. via logger
 9  command)
10        SysSock.Use="off") # Turn off message reception via local log socket;
11                           # local messages are retrieved through imjournal now.
12 module(load="imjournal"             # provides access to the systemd journal
13        StateFile="imjournal.state") # File to store the position in the journal
14 #module(load="imklog") # reads kernel messages (the same are read from journald)
15 #module(load"immark") # provides --MARK-- message capability
16
17 # Provides UDP syslog reception
18 # for parameters see http://www.rsyslog.com/doc/imudp.html
19 #module(load="imudp") # needs to be done just once
20 #input(type="imudp" port="514")
21
22 # Provides TCP syslog reception
23 # for parameters see http://www.rsyslog.com/doc/imtcp.html
24 #module(load="imtcp") # needs to be done just once
25 #input(type="imtcp" port="514")
26
27 #### GLOBAL DIRECTIVES ####
```

```
28
29 # Where to place auxiliary files
30 global(workDirectory="/var/lib/rsyslog")
31
32 # Use default timestamp format
33 module(load="builtin:omfile" Template="RSYSLOG_TraditionalFileFormat")
34
35 # Include all config files in /etc/rsyslog.d/
36 include(file="/etc/rsyslog.d/*.conf" mode="optional")
37
38 #### RULES ####
39
40 # Log all kernel messages to the console.
41 # Logging much else clutters up the screen.
42 #kern.*                                                 /dev/console
43
44 # Log anything (except mail) of level info or higher.
45 # Don't log private authentication messages!
46 *.info;mail.none;authpriv.none;cron.none                /var/log/messages
47
48 # The authpriv file has restricted access.
48 # The authpriv file has restricted access.
49 authpriv.*                                              /var/log/secure
50
51 # Log all the mail messages in one place.
52 mail.*                                                  -/var/log/maillog
53
54
55 # Log cron stuff
56 cron.*                                                  /var/log/cron
57
58 # Everybody gets emergency messages
59 *.emerg                                                 :omusrmsg:*
60
61 # Save news errors of level crit and higher in a special file.
62 uucp,news.crit                                          /var/log/spooler
63
64 # Save boot messages also to boot.log
65 local7.*                                                /var/log/boot.log
66
67
68 # ### sample forwarding rule ###
69 #action(type="omfwd"
70 # An on-disk queue is created for this action. If the remote host is
71 # down, messages are spooled to disk and sent when it is up again.
72 #queue.filename="fwdRule1"       # unique name prefix for spool files
73 #queue.maxdiskspace="1g"         # 1gb space limit (use as much as possible)
74 #queue.saveonshutdown="on"       # save messages to disk on shutdown
75 #queue.type="LinkedList"         # run asynchronously
76 #action.resumeRetryCount="-1"    # infinite retries if host is down
77 # Remote Logging (we use TCP for reliable delivery)
78 # remote_host is: name/ip, e.g. 192.168.0.1, port optional e.g. 10514
79 #Target="remote_host" Port="XXX" Protocol="tcp")
```

Искам да погледнете **authpriv.*** правилото на **49** линия.
Има две колони в едно правило:
От ляво е избирателя и действието на избирателя в дясно.

Избирателя е разделен на какво един вид се търси, а това след точката е един вид какъв тип съобщения се издирват за да се съобщи за тях. Тоест приоритета е променен към критични съобщения задължително да се логват в **/var/log/secure**

Линиите от **44** до **46** също са важни, това което правят е:
Логват всяко нещо с каквото и да е ниво на информация или по високо в **/var/log/messages**
Погледнете **wildcard** символа в началото.

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

Това което реално прави е логва върнати мейли, информационни логове, достъп и хрон.

Ако се интересувате от това да запишете съобщение в логовете;

Използвайте **logger "Text here"**

==========
==========

Другият метод за логване в Линукс е със сервиза **journald**.

Байнъри файл запазен в **/var/run**

/var/run е виртуална файлова система в RAM на компютъра.

- Има фиксиран размер и не е толкова постоянен.
- Данните са структурирани и индексирани.

Ако е необходимо може да бъде нагласен да бъде постоянен и дори да препраща информация към **syslog** това също може да ни даде качеството да можем да изпращаме информация от логве дори в на външен сървър Логинг сървър.

За да видим какво е вписано в **journal** ползваме **journalctl**.

Пишейки тази команда ще ни даде възможността да видим всички journal записи. Ако искаме да сме по точни можем да напишем **jorunalctl –k** за да видим съобщения от ядрото. За да видим за крон сервиза можем да видим по следният начин.

journalctl /sbin/cron

```
[nick@localhost ~]$ journalctl /sbin/crond
-- Logs begin at Wed 2020-09-23 15:24:23 EEST, end at Wed 2020-09-23 15:43:29 EEST. --
Sep 23 15:24:32 localhost.localdomain crond[1231]: (CRON) STARTUP (1.5.2)
Sep 23 15:24:32 localhost.localdomain crond[1231]: (CRON) INFO (Syslog will be used instead o>
Sep 23 15:24:32 localhost.localdomain crond[1231]: (CRON) INFO (RANDOM_DELAY will be scaled w>
Sep 23 15:24:33 localhost.localdomain crond[1231]: (CRON) INFO (running with inotify support)
```

Можем да видим подобна информация като тази от **systemctl** като напишем **journalctl –u crond**
u=unit в ситуацията.

```
[nick@localhost ~]$ journalctl -u crond
-- Logs begin at Wed 2020-09-23 15:24:23 EEST, end at Wed 2020-09-23 16:01:47 EEST. --
Sep 23 15:24:32 localhost.localdomain systemd[1]: Started Command Scheduler.
Sep 23 15:24:32 localhost.localdomain crond[1231]: (CRON) STARTUP (1.5.2)
Sep 23 15:24:32 localhost.localdomain crond[1231]: (CRON) INFO (Syslog will be used instead o>
Sep 23 15:24:32 localhost.localdomain crond[1231]: (CRON) INFO (RANDOM_DELAY will be scaled w>
Sep 23 15:24:33 localhost.localdomain crond[1231]: (CRON) INFO (running with inotify support)
Sep 23 16:01:01 localhost.localdomain CROND[3299]: (root) CMD (run-parts /etc/cron.hourly)
Sep 23 16:01:01 localhost.localdomain anacron[3310]: Anacron started on 2020-09-23
Sep 23 16:01:01 localhost.localdomain anacron[3310]: Will run job `cron.daily' in 5 min.
Sep 23 16:01:01 localhost.localdomain anacron[3310]: Will run job `cron.weekly' in 25 min.
Sep 23 16:01:01 localhost.localdomain anacron[3310]: Will run job `cron.monthly' in 45 min.
Sep 23 16:01:01 localhost.localdomain anacron[3310]: Jobs will be executed sequentially
lines 1-12/12 (END)
```

```
[nick@localhost ~]$ systemctl status crond
● crond.service - Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2020-09-23 15:24:32 EEST; 38min ago
 Main PID: 1231 (crond)
    Tasks: 2 (limit: 11324)
   Memory: 2.1M
   CGroup: /system.slice/crond.service
           ├─1231 /usr/sbin/crond -n
           └─3310 /usr/sbin/anacron -s

Sep 23 15:24:32 localhost.localdomain crond[1231]: (CRON) STARTUP (1.5.2)
Sep 23 15:24:32 localhost.localdomain crond[1231]: (CRON) INFO (Syslog will be used instead o>
Sep 23 15:24:32 localhost.localdomain crond[1231]: (CRON) INFO (RANDOM_DELAY will be scaled w>
Sep 23 15:24:33 localhost.localdomain crond[1231]: (CRON) INFO (running with inotify support)
Sep 23 16:01:01 localhost.localdomain CROND[3299]: (root) CMD (run-parts /etc/cron.hourly)
Sep 23 16:01:01 localhost.localdomain anacron[3310]: Anacron started on 2020-09-23
Sep 23 16:01:01 localhost.localdomain anacron[3310]: Will run job `cron.daily' in 5 min.
Sep 23 16:01:01 localhost.localdomain anacron[3310]: Will run job `cron.weekly' in 25 min.
Sep 23 16:01:01 localhost.localdomain anacron[3310]: Will run job `cron.monthly' in 45 min.
Sep 23 16:01:01 localhost.localdomain anacron[3310]: Jobs will be executed sequentially
```

journalctl –f следва лога като **tail –f**

За да направим **journalctl** постоянен трябва да направим директория и да рестартираме сервиза:

sudo mkdir /var/log/journal
sudo systemctl restart systemd-journald

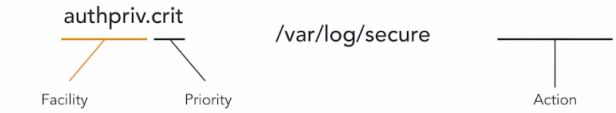
Можем също да сме по точни в **journalctl**:

journalctl --since "2020-01-10 17:00:00"
Journalctl --until "2020-10-10"
Journalctl --since "date time" --until "date time"

journalctl --since time --until "1 hour ago"

==========
==========

Малко по рано поглендахме rsyslog.conf файла и грубо разбрахме как правилата работят и как се създават.



Разбрахме, че филтрите са направени от филтри и акции.
Филтъра уточнява чии предмети във един лог файл да работят.
Състои се от един вид съоръжение и даден приоритет и, че акциите потвърждават какво действие лога ще извърши.

Съоръжението е системата която създава системното съобщение и се състои от видовете .

Филтрите могат да бъдат:

1. **kern**
2. **user**
3. **mail**
4. **daemon**
5. **auth**
6. **syslog**
7. **lpr**
8. **news**

9. **uucp**
10. **cron**
11. **authpriv**
12. **ftp**
13. **local0 –local7**

Приоритетите могат да бъдат:

1. Debug най ниско
2. Info
3. Notice
4. Warning ако сложим това ще показва всичко от 5 до 8
5. Err
6. Crit
7. Alert
8. Emerg най високо

Когато това бъде вписано ще логовете ще се филтрират само съобщения от този тип.
Ако искам да центрирам само един приоритет ще използвам знака равно след него:

mail.=crit
Примера даден ще насочи системата да вади съобщения които са критични без никакви други.

Ако искаме да извадим даден приоритет ще ползваме удивителна, като примера по долу:

mail.!crit

Това което ще става сега е следното - всички други освен **crit** ще се показват.

Ако искаме да сложим всички приоритети ще сложим:

mail.*

Можем да филтрираме съобщенията базирани на техните свойства, тоест специфичен текст:

:msg, contains, "string"

Където sting е съобщението, което искаме да филтрираме.

Други възможни са:

1. Contains
2. Isequal
3. Startswith
4. Regex
5. Eregex
6. Isempty

Примерно:

:hostname, isequal, "server1"

Можем да направим така че лога да се филтрира базирано на това което съдържа:

Примерно:

```
if $programname == word' then {
        action(type="omfile" file="/var/log/openvpn.log")
if $msg contains 'init' then
        action(type="omfile" file="/var/log/init.log")
else
        action(type="omfile" file="/var/log/init.log")
        }
```

https://www.monitis.com/blog/how-to-filter-logs-with-syslog-ng/

==========
==========
Филтриране на данните по даден критерий.

Логовете трябва да бъдат един вид завъртени, за да не станат твърде големи.

Конфигурационният файл на **logrotate** е:

/etc/logroate.conf

less -N /etc/logrotate.conf

```
 # see "man logrotate" for details
                                                    # rotate log files weekly
                                                    weekly
                                                    Това означава че седмично ще се архивират логовете

                                                    # keep 4 weeks worth of backlogs
                                                    rotate 4
                                                    Колко седмици трябва да минат


                                                    # create new (empty) log files after rotating old ones
                                                    create
                                                    Създава новите логове след ротацията.

                                                    #use date as suffix of the rotated file
                                                    dateext
                                                    Ползва дата на завъртяният файл
                                                    # uncomment this if you want your log files compressed
                                                    #compress компресира лог файловете ако се откоментира

                                                    # RPM packages drop log rotation information into this directory
                                                    include /etc/logrotate.d включва всички конфигурационни файлове в
                                                    /etc/logrotate.d ако искаме да направим специфични правила ще
                                                    трябва да се случат в тази папка
                                                    # 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.
```

cd /etc/logrotate.d/

ls -lah /etc/logrotate.d/

==========
==========

За всеки филтър си има дадено действие:

Действията свързани с логове, обикновено включват запазване на съобщения към файл.

Например:

cron* /var/log/cron.log

Всички съобщения свързани с **cron** се записват в **cron.log**
Със записването на всяка линия се синхронизира пътя към този файл. Това се нарича синхронизиран статичен път.

За да предпазим системата от презаписване и пре-синхронизиране със логове можем да направим следното - да го запазим в Рама.

cron* -/var/log/cron.log

Тази методика се нарича не-синхронизиран статичен път. Един вид след като изключим системата лога изчезва и всичко започва на ново.

Това разгледахме беше статичният път, нека си поговорим за динамичният път.

С динамичният път можем да си направим един шаблон.

cron* ?dynamiccronlog

След като направим шаблона ще можем да ползваме името с въпросителна в правилото което нагласяме.

Шаблона може да изглежда ето така:

```
template(name="dynamiccronlog"
type="list") {
constant(value="/var/log/cron_logs/")
        property(name="timegenerated")
        constant(value"-cron.log")
        }
```

Когато създавате шаблони може да използвате качествата на съобщения например:

%msg%
%HOSTNAME%
%PRI%

И много други.

За да разберете всички качества може да видите **man page на rsyslog.conf** търсеки **Available Properties.**

Можем да сложим и обхвати на знаците примерно:

%msg:1:10%
%msg:1:10:lowercase%

Ето пример за логване с шаблон който архивира съобщение, тежест, съоръжение, времева щампа , име на хост, текст на съобщението и нов ред :

```
template(name"dynamiccronlog"
type="list") {
property(name"syslogserverity")
property(name="syslogfacility")
property(name"HOSTNAME")
property(name="msg")
property(name="\n")
}
```

==========

==========

По дизайн systemdjournal е направен да работи използвайки само рама на машината.
Ние можем да го направим да записва нещата на хард диска но това може да изземе много пространство от машината.

sudo mkdir /var/log/journal
sudo systemctl restart systemd-journald

За да проверим колко място заемат тези логове можем да използваме:

```
[nick@localhost logrotate.d]$ sudo journalctl --disk-usage
[sudo] password for nick:
Archived and active journals take up 8.0M in the file system.
```

sudo journalctl –disk-usage

Както виждате примера ми е доста лек и има само 8 мегабайта в употреба, но представете си, че е в продукция този сървър. Това могат да бъдат гигабайти.

Друг начин да видите колко се ползват е:

sudo du -sh /var/log/journal

Имайте в предвид, че **du** показва колко се ползва, а не колко ще има.

Препоръката е да триете логовете базирано на тяхната възраст за да не се задръства диска.

ls -l /var/log/journal/*

```
[nick@localhost logrotate.d]$ ls -l /var/log/journal/*
total 8192
-rw-r-----. 1 root root 8388608 Sep 23 20:19 system.journal
```

Както виждате имаме дата на създаването на лога.

Ако имаме архивирани журнали можем да ги трием базирано на тяхната възраст например с командата:

sudo journalctl --vacuum-time=30d

Може и когато стигнат определен размер:

sudo journalctl --vacuum-size=1G

Ще изтрие всички логове по големи от 1гб.

Може и по файл:

sudo jorunalctl –vacuum-files=2

==========

==========

Има времена в които диска пропада и ние трябва да прочетем данните от възстановеният диск. За нормалното логване с нашият **syslog** или други еквиваленти е лесно, просто **mount** на диска на възстановената файлова система в нашата файлова система. После ползваме **grep** за да намерим това което търсим. Това работи защото syslog съобщенията са в текстови формат.
grep failure /media/recovered/var/log/messages.log

Обаче, на **systemd-journald** са байнърита и неможем да правим това. За да можем да четем журнали от възстановена система те трябва да са статични:

mkdir /var/log/journal
ls /var/log/journal
F836e38657ba4bf7895c79874a004b5f == UID

След като се рестартира **journald** се създава папка с името на **UID**.
Когато ползваме **journalctl** автоматично той чете какво има в **/var/log/UID**

```
[nick@localhost log]$ ls -lah /var/log/journal/f836e38657ba4bf7895c79874a004b5f/
total 16M
drwxr-xr-x. 2 root root   53 Sep 23 20:25 .
drwxr-xr-x. 3 root root   46 Sep 23 16:20 ..
-rw-r-----. 1 root root 8.0M Sep 23 20:41 system.journal
-rw-r-----+ 1 root root 8.0M Sep 23 20:36 user-1000.journal
```

Ако възстановим журнал от възстановена система с друго **UID** тогава **journalctl** няма да може да чете автоматично съхраненото вътре.
За да накараме **journalctl** да чете журнала в друга папка ще трябва да ползваме следното:

Представете си, че диска е на закачен на **/media/recovered/** . Това посочва, че възстановеният системен журнал има следният път **/media/recovered/var/log/journal**.
За да го отворим:

sudo journalctl –directory=/media/recovered/var/log/journal/

За да накараме journalctl да чете друг файл където и да е било ползваме, насочването към файл:

sudo journalctl --file=/var/log/journal/recovered.journal

Другата опция е да пъхнете двете папки с UID в една папка в ситуацията - **/var/log/journal** и да ги съберем с:

sudo journalctl -m