# journalctl工具是CentOS-7才有的工具

好处：Systemd 统一管理所有 Unit 的启动日志。带来的好处就是 ，可以只用journalctl一个命令，查看所有日志（内核日志和 应用日志）。日志的配置文件/etc/systemd/journald.conf

## journalctl用法

### 1查看所有日志（默认情况下 ，只保存本次启动的日志）

$journalctl

## **2查看内核日志（不显示应用日志）**

$journalctl -k

## **3查看系统本次启动的日志**

$journalctl -b

## **4查看上一次启动的日志（需更改设置）**

在该[Journal]部分下，将该Storage=选项设置为“persistent”以启用持久记录：

vim /etc/systemd/journald.conf

. . .

[Journal]

Storage=persistent

在您的服务器上启用了保存以前的引导时，journalctl提供了一些命令来帮助您将引导作为分割单位来使用。要查看journald知道的引导，请使用以下--list-boots选项journalctl：

[root@centos7 ~]# journalctl --list-boots

-1 00d066e11cb3412a912cb804cee123b5 Thu 2018-02-22 17:01:47 CST—Thu 2018-02-22 17:09:15 CST

0 63f75abbe94c4087bc2cc3cdb3b57100 Thu 2018-02-22 17:09:10 CST—Thu 2018-02-22 17:10:19 CST

这将为每次启动显示一行。第一列是启动的偏移量，可用于轻松引用启动journalctl。如果您需要绝对参考，则启动ID位于第二列。您可以通过在结束时列出的两个时间规范来指出引导会话引用的时间。

要显示来自这些引导的信息，您可以使用来自第一列或第二列的信息。

例如，要查看上一次启动的日志，请使用-1带有该-b标志的相对指针：

$journalctl -b -1

## **5查看指定时间的日志**

可以使用--since和--until选项过滤任意时间限制，这些限制分别显示给定时间之前或之后的条目。

例如： #"显示2017年10月30号，18点10分30秒到当前时间之间的所有日志信息"

journalctl --since="2017-10-30 18:10:30"

另外，journal还能够理解部分相对值及命名简写。例如，大家可以使用“yesterday”、“today”、“tomorrow”或者“now”等表达。另外，我们也可以使用“-”或者“+”设定相对值，或者使用“ago”之前的表达。

例如获取昨天的日志如下：

$journalctl –since yesterday

获取某一个时间段到当前时间的前一个小时的日志

$journalctl --since 09:00 --until "1 hour ago"

获取当前时间的前20分钟的日志

$journalctl --since "20 min ago"

获取某一天到某一个时间段的日志信息

$journalctl --since "2017-01-10" --until "2017-01-11 03:00"

如您所见，定义灵活的时间窗口来过滤您希望看到的条目相对容易。

## **6按消息兴趣过滤**

我们在上面学习了一些可以使用时间限制来过滤日记数据的方法。在本节中，我们将讨论如何根据您感兴趣的服务或组件来进行过滤。systemd日记提供了多种方法来执行此操作。

### 按服务

也许最有用的过滤方式是你感兴趣的单位。我们可以使用这个-u选项来过滤。

例如，查看httpd服务的日志信息

[root@centos7 ~]# journalctl -u httpd.service

-- Logs begin at Thu 2018-02-22 17:01:47 CST, end at Thu 2018-02-22 17:30:01 CST. --Feb 22 17:29:27 centos7.localdomain systemd[1]: Starting The Apache HTTP Server...Feb 22 17:29:27 centos7.localdomain httpd[1610]: AH00558: httpd: Could not reliably determine tFeb 22 17:29:28 centos7.localdomain systemd[1]: Started The Apache HTTP Server.

也可以查看httpd服务当天的运行状况

journalctl -u httpd.service --since today

### 按进程、用户或者群组ID

由于某些服务当中包含多个子进程，因此如果我们希望通过进程ID实现查询，也可以使用相关过滤机制。

这里需要指定\_PID字段。例如，如果PID为8088，则可输入：

$journalctl \_PID=8088

有时候我们可能希望显示全部来自特定用户或者群组的日志条目，这就需要使用\_UID或者\_GID。例如，如果大家的Web服务器运行在www-data用户下，则可这样找到该用户ID：

$id -u www-data

33

1

2

3

4

接下来，我们可以使用该ID返回过滤后的journal结果：

$journalctl \_UID=33 --since today

Systemd journal拥有多种可实现过滤功能的字段。其中一些来自被记录的进程，有些则由journald用于自系统中收集特定时间段内的日志。

之前提到的\_PID属于后一种。Journal会自动记录并检索进程PID，以备日后过滤之用。大家可以查看当前全部可用journal字段：

$man systemd.journal-fields

下面来看针对这些字段的过滤机制。-F选项可用于显示特定journal字段内的全部可用值。

例如，要查看systemd journal拥有条目的群组ID，可使用以下命令：

[root@centos7 ~]# journalctl -F \_GID

995

42

40

70

172

998

81

0

其将显示全部journal已经存储至群组ID字段内的值，并可用于未来的过滤需求。

### 按优先级

管理员们可能感兴趣的另一种过滤机制为信息优先级。尽管以更为详尽的方式查看日志也很有必要，不过在理解现有信息时，低优先级日志往往会分散我们的注意力并导致理解混乱。

大家可以使用journalctl配合-p选项显示特定优先级的信息，从而过滤掉优先级较低的信息。

例如，只显示错误级别或者更高的日志条目：

[root@centos7 ~]# journalctl -p err -b

-- Logs begin at Thu 2018-02-22 17:01:47 CST, end at Thu 2018-02-22 17:40:02 CST. --

Feb 22 17:09:10 centos7.localdomain kernel: sd 0:0:0:0: [sda] Assuming drive cache: write throu

Feb 22 17:09:12 centos7.localdomain kernel: piix4\_smbus 0000:00:07.3: SMBus Host Controller not

Feb 22 17:09:15 centos7.localdomain rsyslogd[593]: error during parsing file /etc/rsyslog.conf,

Feb 22 17:09:47 centos7.localdomain pulseaudio[1232]: [alsa-sink-ES1371/1] alsa-sink.c: ALSA wo

Feb 22 17:09:47 centos7.localdomain pulseaudio[1232]: [alsa-sink-ES1371/1] alsa-sink.c: Most li

Feb 22 17:09:47 centos7.localdomain pulseaudio[1232]: [alsa-sink-ES1371/1] alsa-sink.c: We were

Feb 22 17:09:48 centos7.localdomain spice-vdagent[1274]: Cannot access vdagent virtio channel /

lines 1-8/8 (END)

这将只显示被标记为错误、严重、警告或者紧急级别的信息。Journal的这种实现方式与标准syslog信息在级别上是一致的。大家可以使用优先级名称或者其相关量化值。以下各数字为由最高到最低优先级：

0: emerg

1: alert

2: crit

3: err

4: warning

5: notice

6: info

7: debug

例如：

[root@centos7 ~]# journalctl -p 3 -b

-- Logs begin at Thu 2018-02-22 17:01:47 CST, end at Thu 2018-02-22 17:50:01 CST. --

Feb 22 17:09:10 centos7.localdomain kernel: sd 0:0:0:0: [sda] Assuming drive cache: write throu

Feb 22 17:09:12 centos7.localdomain kernel: piix4\_smbus 0000:00:07.3: SMBus Host Controller not

Feb 22 17:09:15 centos7.localdomain rsyslogd[593]: error during parsing file /etc/rsyslog.conf,

Feb 22 17:09:47 centos7.localdomain pulseaudio[1232]: [alsa-sink-ES1371/1] alsa-sink.c: ALSA wo

Feb 22 17:09:47 centos7.localdomain pulseaudio[1232]: [alsa-sink-ES1371/1] alsa-sink.c: Most li

Feb 22 17:09:47 centos7.localdomain pulseaudio[1232]: [alsa-sink-ES1371/1] alsa-sink.c: We were

Feb 22 17:09:48 centos7.localdomain spice-vdagent[1274]: Cannot access vdagent virtio channel /

lines 1-8/8 (END)

### 修改journal显示内容

到这里，过滤部分已经介绍完毕。我们也可以使用多种方式对输出结果进行修改，从而调整journalctl的显示内容。

分页显示（默认）或者改为正常标准输出

分页显示，其中插入省略号以代表被移除的信息，使用–no-full选

$journalctl --no-full

. . .

Feb 04 20:54:13 journalme sshd[937]: Failed password for root from 83.234.207.60...h2Feb 04 20:54:13 journalme sshd[937]: Connection closed by 83.234.207.60 [preauth]

大家也可以要求其显示全部信息，无论其是否包含不可输出的字符。具体方式为添加-a标记：

$journalctl -a

默认情况下，journalctl会在pager内显示输出结果以便于查阅。如果大家希望利用文本操作工具对数据进行处理，则可能需要使用标准格式。在这种情况下，我们需要使用–no-pager选项：

$journalctl --no-pager

这样就可以用一些工具过滤出自己感兴趣的信息了

**命令定义输出格式**

如果大家需要对journal条目进行处理，则可能需要使用更易使用的格式以简化数据解析工作。幸运的是，journal能够以多种格式进行显示，只须添加-o选项加格式说明即可。

例如，我们可以将journal条目输出为JSON格式：

[root@centos7 ~]# journalctl -b -u httpd -o json

{ "\_\_CURSOR" : "s=8fa6a8a1c6264c7b938e4d23584ae602;i=149d;b=63f75abbe94c4087bc2cc3cdb3b57100;m=46edf6e6;t=565c9ae1d38f7;x=b3a1eaebceb26d5b", "\_\_REALTIME\_TIMESTAMP" : "1519291767535863", "\_\_MONOTONIC\_TIMESTAMP"

{ "\_\_CURSOR" : "s=8fa6a8a1c6264c7b938e4d23584ae602;i=149e;b=63f75abbe94c4087bc2cc3cdb3b57100;m=46f3506d;t=565c9ae22927d;x=91ef081943191196", "\_\_REALTIME\_TIMESTAMP" : "1519291767886461", "\_\_MONOTONIC\_TIMESTAMP"

{ "\_\_CURSOR" : "s=8fa6a8a1c6264c7b938e4d23584ae602;i=149f;b=63f75abbe94c4087bc2cc3cdb3b57100;m=46f7a7e4;t=565c9ae26e9f5;x=1f0dc6e3105af151", "\_\_REALTIME\_TIMESTAMP" : "1519291768170997", "\_\_MONOTONIC\_TIMESTAMP"

这种方式对于工具解析非常重要。大家也可以使用json-pretty格式以更好地处理数据结构，这种方法易读性，显示的内容也比较全面：

[root@centos7 ~]# journalctl -u httpd -o json-pretty

{

"\_\_CURSOR" : "s=8fa6a8a1c6264c7b938e4d23584ae602;i=149d;b=63f75abbe94c4087bc2cc3cdb3b57

"\_\_REALTIME\_TIMESTAMP" : "1519291767535863",

"\_\_MONOTONIC\_TIMESTAMP" : "1190000358",

"\_BOOT\_ID" : "63f75abbe94c4087bc2cc3cdb3b57100",

"PRIORITY" : "6",

"\_UID" : "0",

"\_GID" : "0",

"\_MACHINE\_ID" : "534ca72579bb44b4b5c707ba441967eb",

"\_HOSTNAME" : "centos7.localdomain",

"SYSLOG\_FACILITY" : "3",

"SYSLOG\_IDENTIFIER" : "systemd",

"\_TRANSPORT" : "journal",

"\_PID" : "1",

"\_COMM" : "systemd",

"\_EXE" : "/usr/lib/systemd/systemd",

"\_CAP\_EFFECTIVE" : "1fffffffff",

"\_SYSTEMD\_CGROUP" : "/",

"CODE\_FILE" : "src/core/unit.c",

"CODE\_LINE" : "1417",

"CODE\_FUNCTION" : "unit\_status\_log\_starting\_stopping\_reloading",

"MESSAGE\_ID" : "7d4958e842da4a758f6c1cdc7b36dcc5",

"\_CMDLINE" : "/usr/lib/systemd/systemd --switched-root --system --deserialize 21",

"\_SELINUX\_CONTEXT" : "system\_u:system\_r:init\_t:s0",

"UNIT" : "httpd.service",

"MESSAGE" : "Starting The Apache HTTP Server...",

"\_SOURCE\_REALTIME\_TIMESTAMP" : "1519291767533650"

}

以下为可用于显示的各类格式：

cat: 只显示信息字段本身。

export: 适合传输或备份的二进制格式。

json: 标准JSON，每行一个条目。

json-pretty: JSON格式，适合人类阅读习惯。

json-sse: JSON格式，经过打包以兼容server-sent事件。

short: 默认syslog类输出格式。

short-iso: 默认格式，强调显示ISO 8601挂钟时间戳。

short-monotonic: 默认格式，提供普通时间戳。

short-precise: 默认格式，提供微秒级精度。

verbose: 显示该条目的全部可用journal字段，包括通常被内部隐藏的字段。

**活动进程监控**

Journalctl命令还能够帮助管理员以类似于tail的方式监控活动或近期进程。这项功能内置于journalctl当中，允许大家在无需借助其它工具的前提下实现访问。

### **显示近期日志**

要显示特定数量的记录，大家可以使用-n选项，类似为tail -n功能。默认情况下只显示最后发生的10条日志，但是也可以指定。  
例如：

[root@centos7 ~]# journalctl -n20

-- Logs begin at Thu 2018-02-22 17:01:47 CST, end at Thu 2018-02-22 18:20:01 CST. --

Feb 22 17:40:01 centos7.localdomain systemd[1]: Started Session 5 of user root.

Feb 22 17:40:02 centos7.localdomain systemd[1]: Starting Session 5 of user root.

Feb 22 17:40:02 centos7.localdomain CROND[1754]: (root) CMD (/usr/lib64/sa/sa1 1 1)

Feb 22 17:50:01 centos7.localdomain systemd[1]: Started Session 6 of user root.

Feb 22 17:50:01 centos7.localdomain systemd[1]: Starting Session 6 of user root.

Feb 22 17:50:01 centos7.localdomain CROND[1855]: (root) CMD (/usr/lib64/sa/sa1 1 1)

Feb 22 18:00:01 centos7.localdomain systemd[1]: Started Session 7 of user root.

Feb 22 18:00:01 centos7.localdomain CROND[1962]: (root) CMD (/usr/lib64/sa/sa1 1 1)

Feb 22 18:00:01 centos7.localdomain systemd[1]: Starting Session 7 of user root.

Feb 22 18:01:01 centos7.localdomain systemd[1]: Started Session 8 of user root.

Feb 22 18:01:01 centos7.localdomain systemd[1]: Starting Session 8 of user root.

Feb 22 18:01:01 centos7.localdomain CROND[1983]: (root) CMD (run-parts /etc/cron.hourly)

Feb 22 18:01:01 centos7.localdomain run-parts(/etc/cron.hourly)[1989]: starting 0anacron

Feb 22 18:01:01 centos7.localdomain run-parts(/etc/cron.hourly)[1996]: finished 0anacron

Feb 22 18:10:01 centos7.localdomain systemd[1]: Started Session 9 of user root.

Feb 22 18:10:01 centos7.localdomain CROND[2092]: (root) CMD (/usr/lib64/sa/sa1 1 1)

Feb 22 18:10:01 centos7.localdomain systemd[1]: Starting Session 9 of user root.

Feb 22 18:20:01 centos7.localdomain systemd[1]: Started Session 10 of user root.

Feb 22 18:20:01 centos7.localdomain CROND[2197]: (root) CMD (/usr/lib64/sa/sa1 1 1)

Feb 22 18:20:01 centos7.localdomain systemd[1]: Starting Session 10 of user root.

### **追踪日志**

要主动追踪当前正在编写的日志，大家可以使用-f标记。同样功能类似为tail -f，只要不终止，会一直监控

$journalctl -f

### **Journal维护**

存储这么多数据当然会带来巨大压力，因此我们还需要了解如何清理部分陈旧日志以释放存储空间。

#### **查看当前日志占用磁盘的空间的总大小**

[root@centos7 ~]# journalctl --disk-usage

Archived and active journals take up 8.0M on disk.

#### **指定日志文件最大空间**

$journalctl --vacuum-size=1G

#### **指定日志文件保存多久**

$journalctl --vacuum-time=1years

## **journalctl相关配置**

大家可以配置自己的服务器以限定journal所能占用的最高容量。要实现这一点，我们需要编辑/etc/systemd/journald.conf文件。

以下条目可用于限定journal体积的膨胀速度：

SystemMaxUse=: 指定journal所能使用的最高持久存储容量。

SystemKeepFree=: 指定journal在添加新条目时需要保留的剩余空间。

SystemMaxFileSize=: 控制单一journal文件大小，符合要求方可被转为持久存储。

RuntimeMaxUse=: 指定易失性存储中的最大可用磁盘容量（/run文件系统之内）。

RuntimeKeepFree=: 指定向易失性存储内写入数据时为其它应用保留的空间量（/run文件系统之内）。

RuntimeMaxFileSize=: 指定单一journal文件可占用的最大易失性存储容量（/run文件系统之内）。

通过设置上述值，大家可以控制journald对服务器空间的消耗及保留方式。