

博客园

cnblogs.com

首页

新闻

博问

专区

闪存

班级

代码改变世界

搜索

菜单

通知

sparkdev

博客园 首页 新随笔 联系 订阅 管理

随笔 - 231 文章 - 0 评论 - 1571

linux journalctl 命令

目录

- [Help](#)
- [输出所有的日志记录](#)
- [匹配\(match\)](#)
- [把日志保存到文件中](#)
- [限定日志所能占用的最高容量](#)
- [查看某次启动后的日志](#)
- [查看指定时间段的日志](#)
- [同时应用 match 和时间过滤条件](#)
- [按 unit 过滤日志](#)
- [通过日志级别进行过滤](#)
- [实时更新日志](#)
- [只显示最新的 n 行](#)
- [控制输出](#)
- [按可执行文件的路径过滤](#)
- [查看内核日志](#)
- [总结](#)

journalctl 用来查询 systemd-journald 服务收集到的日志。systemd-journald 服务是 systemd init 系统提供的收集系统日志的服务。

命令格式为：  
**journalctl [OPTIONS...] [MATCHES...]**

journalctl 命令的路径为：  
/bin/journalctl

## Help

可以通过 man page 和 -h 选项来获得最直接的帮助文档：

```
$ man journalctl
$ journalctl -h
```

## 输出所有的日志记录

不带任何选项时，journalctl 输出所有的日志记录：

```
$ sudo journalctl
```

```
-- Logs begin at Wed 2018-03-14 12:30:46 CST, end at Sat 2018-03-24 16:21:47 CST. --
3月 14 12:30:46 crab kernel: random: get_random_bytes called from start_kernel+0x42/0x
3月 14 12:30:46 crab kernel: Linux version 4.13.0-36-generic (buildd@lcy01-amd64-017)
3月 14 12:30:46 crab kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.13.0-36-generic
```

公告

昵称： sparkdev  
园龄： 4年7个月  
荣誉： 推荐博客  
粉丝： 905  
关注： 32  
[+加关注](#)

< 2020年12月 >

日	一	二	三	四	五	六
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索

找找看

谷歌搜索

最新随笔

1.创建 SysV 风格的 linux daemon 程序

2.Linux session(会话)

3.Linux job control

4.Linux 伪终端(pty)

5.Linux 终端(TTY)

6.Linux Capabilities 简介

7.用什么监控我们的容器？

8.Ubuntu Server：自动更新

9.Ubuntu：apt 命令

10.Ubuntu：apt-get 命令

我的标签

Linux(80)

docker(38)

Azure(28)

Golang(16)

PowerShell(15)

jenkins(12)

CI/CD(10)

ubuntu(10)

teamcity(9)

log(9)

更多

https://www.cnblogs.com/sparkdev/p/8795141.html#title12

1/8

这基本上没什么用处，因为你立即就被洪水般的日志记录给淹没了。所以，接下来我们学习如何高效的过滤出有价值的日志信息。

## 匹配(match)

我们可以通过 "FIELD=VALUE" 的格式来匹配具体的日志记录, 如：

```
_SYSTEMD_UNIT=cron.service
```

日志信息的定义也类似一个实体类型，具体的信息被保存在各个对应的字段中，比如 MESSAGE、MESSAGE\_ID、\_PID、\_UID、\_HOSTNAME、\_SYSTEMD\_UNIT 等等(通过 man 7 systemd.journal-fields 可以查看所有可用的 match 字段)。因此可以通过这些字段的内容匹配相关的日志记录：

```
nick@crab:~$ journalctl _SYSTEMD_UNIT=cron.service
-- Logs begin at Wed 2018-03-14 12:30:46 CST, end at Tue 2018-03-27 11:18:44 CST. --
3月 14 12:30:48 crab cron[687]: (CRON) INFO (pidfile fd = 3)
3月 14 12:30:48 crab cron[687]: (CRON) INFO (Running @reboot jobs)
3月 14 13:17:01 crab CRON[1179]: pam_unix(cron:session): session opened for user root
3月 14 13:17:01 crab CRON[1180]: (root) CMD ( cd / && run-parts --report /etc/cron.t
3月 14 13:17:01 crab CRON[1179]: pam_unix(cron:session): session closed for user root
```

上图中的输出是 cron.service 服务相关的日志记录。

可以同时添加多个字段进行匹配，它们之间是与的关系，就是同时符合多个条件的记录才会被匹配，比如添加 PRIORITY 字段的匹配条件：

```
$ journalctl _SYSTEMD_UNIT=cron.service PRIORITY=6
```

```
nick@crab:~$ journalctl _SYSTEMD_UNIT=cron.service PRIORITY=6
-- Logs begin at Wed 2018-03-14 12:30:46 CST, end at Tue 2018-03-27 12:03:36 CST. --
3月 14 12:30:48 crab cron[687]: (CRON) INFO (pidfile fd = 3)
3月 14 12:30:48 crab cron[687]: (CRON) INFO (Running @reboot jobs)
3月 14 13:17:01 crab CRON[1179]: pam_unix(cron:session): session opened for user root
3月 14 13:17:01 crab CRON[1180]: (root) CMD ( cd / && run-parts --report /etc/cron.t
3月 14 13:17:01 crab CRON[1179]: pam_unix(cron:session): session closed for user root
3月 14 14:17:01 crab CRON[1792]: pam_unix(cron:session): session opened for user root
```

注意各个字段的取值，比如为 PRIORITY 设置 debug、info 是不工作的，必须设置为对应的数字。可以通过 -F 选项来查看某个字段的可选值：

```
$ journal -F PRIORITY
```

```
nick@crab:~$ journalctl -F PRIORITY
2
3
4
7
6
5
```

具体的对应方式如下：

- 0: emerg
- 1: alert
- 2: crit
- 3: err
- 4: warning
- 5: notice
- 6: info
- 7: debug

对同一个字段应用多个 match 条件的情况，比如：

```
$ journalctl _SYSTEMD_UNIT=cron.service _SYSTEMD_UNIT=prometheus.service
```

此时 cron.service 和 prometheus.service 的日志都会输出。

### 多个 match 条件的或操作

使用 "+" 号可以对多个匹配字段执行或操作：

```
$ journalctl _SYSTEMD_UNIT=cron.service + _PID=28097
```

上面的命令会输出 cron.service 的日志和进程 28097 的日志。

下面是一个更复杂的例子：

积分与排名

积分 - 692520  
排名 - 362

随笔分类 (346)

- AI(2)
- Ansible(3)
- Azure(28)
- Bash(8)
- C#(9)
- cgroups(2)
- CI/CD(20)
- DevOps(16)
- Docker(39)
- ElasticSearch(1)
- elk(9)
- Git(4)
- Golang(16)
- https(1)
- Jenkins(12)
- 更多

随笔档案 (231)

- 2020年4月(1)
- 2020年1月(1)
- 2019年12月(1)
- 2019年9月(2)
- 2019年8月(7)
- 2019年7月(7)
- 2019年6月(7)
- 2019年5月(7)
- 2019年4月(7)
- 2019年3月(7)
- 2019年2月(7)
- 2019年1月(7)
- 2018年12月(7)
- 2018年11月(7)
- 2018年10月(7)
- 更多

最新评论

- 1. Re:linux dig 命令  
楼主目前整理的是我目前看到最牛逼的!  
--Geeksongs
- 2. Re:Linux 特殊权限 SUID,SGID,SBIT  
centos的/etc/shadow文件权限是000  
1 root root 1.2K Apr 29 2020 /etc/shadow 连超级用户root对它都没有读取权限，这个又是通过什么权限来控制...  
--ImWillis
- 3. Re:Windows OpenSSH 基本用法  
不安全  
--不如做正确的事
- 4. Re:Linux Socket 编程简介  
帮助很大，赞👍  
--快乐张
- 5. Re:Linux 特殊权限 SUID,SGID,SBIT  
开始我还以为你搬运的这篇：Linux special permissions SUID, SGID, SBIT，结果是他搬运的你哈哈  
--ImWillis

```
$ journalctl _SYSTEMD_UNIT=avahi-daemon.service _PID=28097 + _SYSTEMD_UNIT=dbus.service
```

前面的两个条件是与的关系，最后一个条件与前面的两个条件是或的关系，也就是相对于用小括号把前面的两个条件括起来。

## 把日志保存到文件中

systemd-journald 服务收集到的日志默认保存在 /run/log 目录中，重启系统会丢掉以前的日志信息。我们可以通过两种方式让 systemd-journald 服务把所有的日志都保存到文件中，这样重新启动后就不会丢掉以前的日志。

方法一：创建目录 /var/log/journal，然后重启日志服务 systemd-journald.service。

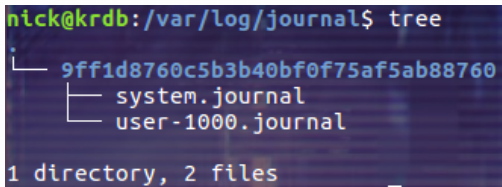
方法二：修改配置文件 /etc/systemd/journald.conf，把 Storage=auto 改为 Storage=persistent，并取消注释，然后重启日志服务 systemd-journald.service。

### 方法一的详细操作

在 /var/log/ 下面创建名为 journal 的目录，并设置权限即可：

```
$ sudo mkdir /var/log/journal
$ sudo chown root:systemd-journal /var/log/journal
$ sudo chmod 2775 /var/log/journal
$ sudo systemctl restart systemd-journald.service
```

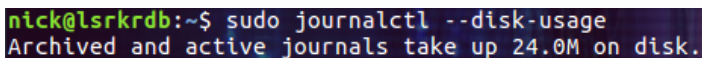
之后 /run/log 下面就没有 journal 的日志了，日志文件被保存在 /var/log/journal 目录下：



```
nick@krdb:/var/log/journal$ tree
.
├── 9ff1d8760c5b3b40bf0f75af5ab88760
│   ├── system.journal
│   └── user-1000.journal
└── 1 directory, 2 files
```

### 查看日志占据的磁盘空间

```
$ sudo journalctl --disk-usage
```



```
nick@lsrkrdb:~$ sudo journalctl --disk-usage
Archived and active journals take up 24.0M on disk.
```

注意：无论是否设置把日志存储到文件，都会得到 disk-usage。

### 清理日志数据

如果大家打算对 journal 记录进行清理，则可使用两种不同方式。

- 使用 -vacuum-size 选项
- 使用 -vacuum-time 选项

如果使用 -vacuum-size 选项，则可硬性指定日志的总体积，意味着其会不断删除旧有记录直到所占容量符合要求：

```
$ sudo journalctl --vacuum-size=1G
```

另一种方式则是使用 -vacuum-time 选项。任何早于这一时间点的条目都将被删除。例如，去年之后的条目才能保留：

```
$ sudo journalctl --vacuum-time=1years
```

## 限定日志所能占用的最高容量

我们可以通过 /etc/systemd/journald.conf 文件来配置 systemd-journald 服务的行为。以下条目可用于限定日志数据可以占用的最大存储数量和日志数据体积的膨胀速度：

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 systemd-journald 服务对服务器空间的消耗及保留方式。

### 阅读排行榜

1. Dockerfile 中的 COPY 与 ADD 命令 (172045)
2. SSH 远程执行任务(113852)
3. linux sudo 命令(108484)
4. Docker: 限制容器可用的内存(91124)
5. Docker: 限制容器可用的 CPU(89403)
6. Windows 支持 OpenSSH 了! (78220)
7. linux useradd 命令基本用法(77039)
8. Linux mount 命令(65971)
9. SSH 端口转发(63015)
10. Docker Compose 引用环境变量(62250)

### 评论排行榜

1. Windows 支持 OpenSSH 了! (42)
2. Docker Machine 详解(37)
3. Docker Machine 简介(34)
4. 用 Docker Machine 创建 Azure 虚拟主机(29)
5. SSH 远程执行任务(29)
6. C# 创建压缩文件(28)
7. 局域网内部署 Docker Registry(27)
8. Python 操作 Azure Blob Storage(24)
9. PowerShell 远程执行任务(22)
10. PowerShell 脚本中的密码(21)

### 推荐排行榜

1. SSH 远程执行任务(59)
2. Windows 支持 OpenSSH 了! (51)
3. Dockerfile 中的 CMD 与 ENTRYPOINT(43)
4. Docker Machine 详解(43)
5. 从 docker 到 runC(41)

## 查看某次启动后的日志

默认情况下 systemd-journald 服务只保存本次启动后的日志(重新启动后丢掉以前的日志)。此时 -b 选项是没啥用的。当我们把 systemd-journald 服务收集到的日志保存到文件中之后，就可以通过下面的命令查看系统的重启记录：

```
$ journalctl --list-boots
```

```
nick@krdb:~$ journalctl --list-boots
-2 bc3e886e9b454bb082bc1bee0b5719ce Mon 2018-03-26 17:57:09 CST-Mon 2018-03-26 17:57:23 CST
-1 9eaabbc25fe343999ef1024e6a16fb58 Mon 2018-03-26 17:57:32 CST-Mon 2018-03-26 18:00:38 CST
0 4cc595bd2f9e4a41a8c613883d6a496f Mon 2018-03-26 18:00:48 CST-Mon 2018-03-26 18:01:51 CST
```

此时我们就可以通过 -b 选项来选择查看某次运行过程中的日志：

```
$ sudo journalctl -b -1
或
$ sudo journalctl -b 9eaabbc25fe343999ef1024e6a16fb58
```

下面的命令都会输出最后一次启动后的日志信息：

```
$ sudo journalctl -b
$ sudo journalctl -b 0
```

## 查看指定时间段的日志

利用 --since 与 --until 选项设定时间段，二者分别负责指定给定时间之前与之后的日志记录。时间值可以使用多种格式，比如下面的格式：

```
YYYY-MM-DD HH:MM:SS
```

如果我们要查询 2018 年 3 月 26 日下午 8:20 之后的日志：

```
$ journalctl --since "2018-03-26 20:20:00"
```

如果以上格式中的某些组成部分未进行填写，系统会直接进行默认填充。例如，如果日期部分未填写，则会直接显示当前日期。如果时间部分未填写，则缺省使用 "00:00:00"(午夜)。秒字段亦可留空，默认值为 "00"，比如下面的命令：

```
$ journalctl --since "2018-03-26" --until "2018-03-26 03:00"
```

另外，journalctl 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。比如获取昨天的日志数据可以使用下面的命令：

```
$ journalctl --since yesterday
```

要获得早上 9:00 到一小时前这段时间内的日志，可以使用下面的命令：

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

## 同时应用 match 和时间过滤条件

实际的使用中更常见的用例是同时应用 match 和时间条件，比如要过滤出某个时间段中 cron 服务的日志记录：

```
$ sudo journalctl _SYSTEMD_UNIT=cron.service --since "2018-03-27" --until "2018-03-27 0:"
```

```
kr@krjpstress:~$ sudo journalctl _SYSTEMD_UNIT=cron.service --since "2018-03-27"
--until "2018-03-27 01:00"
-- Logs begin at Mon 2018-03-26 14:34:12 CST, end at Tue 2018-03-27 17:08:58 CST
Mar 27 00:17:01 krjpstress CRON[10670]: pam_unix(cron:session): session opened for
Mar 27 00:17:01 krjpstress CRON[10671]: (root) CMD ( cd / && run-parts --report
Mar 27 00:17:01 krjpstress CRON[10670]: pam_unix(cron:session): session closed for
lines 1-4/4 (END)
```

## 按 unit 过滤日志

systemd 把几乎所有的任务都抽象成了 unit，因此我们可以方便的使用 -u 选项通过 unit 的名称来过滤器日志记录。查看某个 unit 的日志：

```
$ sudo journalctl -u nginx.service
$ sudo journalctl -u nginx.service --since today
```

还可以使用多个 -u 选项同时获得多个 unit 的日志：

```
$ journalctl -u nginx.service -u php-fpm.service --since today
```

## 通过日志级别进行过滤

除了通过 PRIORITY= 的方式，还可以通过 -p 选项来过滤日志的级别。可以指定的优先级如下：

```
# 0: emerg
# 1: alert
# 2: crit
# 3: err
# 4: warning
# 5: notice
# 6: info
# 7: debug
```

```
$ sudo journalctl -p err
```

注意，这里指定的是优先级的名称。

## 实时更新日志

与 tail -f 类似，journalctl 支持 -f 选项来显示实时的日志：

```
$ sudo journalctl -f
```

如果要查看某个 unit 的实时日志，再加上 -u 选项就可以了：

```
$ sudo journalctl -f -u prometheus.service
```

## 只显示最新的 n 行

命令行选项 -n 用来控制只显示最新的 n 行日志，默认是显示尾部的最新 10 行日志：

```
$ sudo journalctl -n
```

也可以显示尾部指定行数的日志：

```
$ sudo journalctl -n 20
```

下面则是显示 cron.service 服务最新的三行日志：

```
$ journalctl -u cron.service -n 3
```

## 控制输出

### 把结果重定向到标准输出

默认情况下，journalctl 会在 pager 内显示输出结果。如果大家希望利用文本操作工具对数据进行处理，则需要使用标准输出。在这种情况下，我们需要使用 --no-pager 选项。

```
$ sudo journalctl --no-pager
```

这样就可以把结果重定向到我们需要的地方(一般是磁盘文件或者是文本工具)。

### 格式化输出的结果

如果大家需要对日志记录进行处理，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，journalctl 能够以多种格式进行显示，只须添加 -o 选项即可。-o 选项支持的类型如下：

#### short

这是默认的格式，即经典的 syslog 输出格式。

#### short-iso

与 short 类似，强调 ISO 8601 时间戳。

#### short-precise

与 short 类似，提供微秒级精度。

#### short-monotonic



与 short 类似, 强调普通时间戳。

#### verbose

显示全部字段, 包括通常被内部隐藏的字段。

#### export

适合传输或备份的二进制格式。

#### json

标准 json 格式, 每行一条记录。

#### json-pretty

适合阅读的 json 格式。

#### json-sse

经过包装可以兼容 server-sent 事件的 json 格式。

#### cat

只显示信息字段本身。

比如我们要以 json 格式输出 cron.service 的最后一条日志:

```
$ sudo journalctl -u cron.service -n 1 --no-pager -o json
```

```
nick@tiger:~$ sudo journalctl -u cron.service -n 1 --no-pager -o json
{"__CURSOR": "s=e95d7c8a8fe0481b99c2d64ea4339859;i=1b19;b=99f2b199f7cf4b50809a46077b615230;m=12ab7e3f84;t=56888bf845b18;x=e7ea1158427adcce", "__REALTIME_TIMESTAMP": "1522311421516568", "__MONOTONIC_TIMESTAMP": "80186589060", "__BOOT_ID": "99f2b199f7cf4b50809a46077b615230", "PRIORITY": "6", "__SYSTEMD_SLICE": "system.slice", "__MACHINE_ID": "61e8c3574d524990b78b20e85549e6a7", "__HOSTNAME": "tiger", "__TRANSPORT": "syslog", "__UID": "0", "__GID": "0", "__CAP_EFFECTIVE": "3ffffffff", "SYSLOG_FACILITY": "10", "SYSLOG_IDENTIFIER": "CRON", "COMM": "cron", "EXE": "/usr/sbin/cron", "CMDLINE": "/usr/sbin/CRON -f", "AUDIT_LOGINUID": "0", "SYSTEMD_CGROUP": "/system.slice/cron.service", "SYSTEMD_UNIT": "cron.service", "MESSAGE": "pam_unix(cron:session): session closed for user root", "SYSLOG_PID": "17247", "PID": "17247", "AUDIT_SESSION": "23", "SOURCE_REALTIME_TIMESTAMP": "1522311421516371" }
```

而 json-pretty 的格式为:

```
nick@tiger:~$ sudo journalctl -u cron.service -n 1 --no-pager -o json-pretty
{
  "__CURSOR": "s=e95d7c8a8fe0481b99c2d64ea4339859;i=1b19;b=99f2b199f7cf4b50809a46077b615230;m=12ab7e3f84;t=56888bf845b18;x=e7ea1158427adcce",
  "__REALTIME_TIMESTAMP": "1522311421516568",
  "__MONOTONIC_TIMESTAMP": "80186589060",
  "BOOT_ID": "99f2b199f7cf4b50809a46077b615230",
  "PRIORITY": "6",
  "SYSTEMD_SLICE": "system.slice",
  "MACHINE_ID": "61e8c3574d524990b78b20e85549e6a7",
  "HOSTNAME": "tiger",
  "TRANSPORT": "syslog",
  "UID": "0",
  "GID": "0",
  "CAP_EFFECTIVE": "3ffffffff",
  "SYSLOG_FACILITY": "10",
  "SYSLOG_IDENTIFIER": "CRON",
  "COMM": "cron",
  "EXE": "/usr/sbin/cron",
  "CMDLINE": "/usr/sbin/CRON -f",
  "AUDIT_LOGINUID": "0",
  "SYSTEMD_CGROUP": "/system.slice/cron.service",
  "SYSTEMD_UNIT": "cron.service",
  "MESSAGE": "pam_unix(cron:session): session closed for user root",
  "SYSLOG_PID": "17247",
  "PID": "17247",
  "AUDIT_SESSION": "23",
  "SOURCE_REALTIME_TIMESTAMP": "1522311421516371"
}
```

## 按可执行文件的路径过滤

如果在参数中指定某个可执行文件(二进制文件或脚本文件), 则 journalctl 会显示与该可执行文件相关的全部条目。比如可以显示 /usr/lib/systemd/systemd 程序产生的日志:

```
$ sudo journalctl /usr/lib/systemd/systemd
```

也可以显示 /usr/bin/bash 程序产生的日志:

```
$ sudo journalctl /usr/bin/bash
```

## 查看内核日志

如果我们需要查看内核日志, 可以指定 -k 选项, 这样输出的结果中就只有内核日志了。-k 选项是通过 -b 选项加上匹配条件 "\_TRANSPORT=kernel" 实现的。下面是基本的用法:

```
$ sudo journalctl -k
```