一、前言

journalctl是一个功能强大的命令行工具，用于查看和管理系统日志，可以深入了解系统的运行状况、故障信息和关键事件。它也是systemd日志记录器（systemd-journald）的一部分，systemd-journald是systemd守护进程的一个组件，负责收集、存储和检索系统日志，甚至可以将**journalctl**的日志上报给ELK。

本文将介绍**journalctl**的基本概念、用法和常见的使用场景。将详细讨论如何使用**journalctl**来查看和过滤日志消息，以及如何通过搜索和格式化选项来定位特定的日志内容。此外还将探讨如何使用**journalctl**来追踪实时日志并进行分页浏览，以便及时监控系统的运行状态。

二、过滤选项及其作用

不指定来源日志来源选项默认会显示用户可以看到的所有日志记录。

1.指定日志来源(--system,--user)

--system，显示来自系统服务和内核的日志；

--user，显示来自当前用户可以看到的日志。

```
 ⊙ 22:40:42  ~  journalctl --system|tail -n 20
May 27 22:39:05 gentoo sshd[13305]: Accepted keyboard-interactive/pam for root from 192.168.1.3 port 57256 ssh2
May 27 22:39:05 gentoo sshd[13305]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 22:39:05 gentoo systemd-logind[1123]: New session 8 of user root.
May 27 22:39:05 gentoo systemd[1]: Created slice user-0.slice.
May 27 22:39:05 gentoo systemd[1]: Starting user-runtime-dir@0.service...
May 27 22:39:05 gentoo systemd[1]: Finished user-runtime-dir@0.service.
May 27 22:39:05 gentoo systemd[1]: Starting user@0.service...
May 27 22:39:05 gentoo (systemd)[13309]: pam_unix(systemd-user:session): session opened for user root(uid=0) by (uid=0)
May 27 22:39:05 gentoo systemd[13309]: Queued start job for default target default.target.
May 27 22:39:05 gentoo systemd[13309]: Created slice app.slice.
May 27 22:39:05 gentoo systemd[13309]: Reached target paths.target.
May 27 22:39:05 gentoo systemd[13309]: Reached target timers.target.
May 27 22:39:05 gentoo systemd[13309]: Starting dbus.socket...
May 27 22:39:05 gentoo systemd[13309]: Listening on dbus.socket.
May 27 22:39:05 gentoo systemd[13309]: Reached target sockets.target.
May 27 22:39:05 gentoo systemd[13309]: Reached target basic.target.
May 27 22:39:05 gentoo systemd[13309]: Reached target default.target.
May 27 22:39:05 gentoo systemd[13309]: Startup finished in 65ms.
May 27 22:39:05 gentoo systemd[1]: Started user@0.service.
May 27 22:39:05 gentoo systemd[1]: Started session-8.scope.
 ⊙ 22:40:50  ~  journalctl |tail -n 20
May 27 22:39:05 gentoo sshd[13305]: Accepted keyboard-interactive/pam for root from 192.168.1.3 port 57256 ssh2
May 27 22:39:05 gentoo sshd[13305]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 22:39:05 gentoo systemd-logind[1123]: New session 8 of user root.
May 27 22:39:05 gentoo systemd[1]: Created slice user-0.slice.
May 27 22:39:05 gentoo systemd[1]: Starting user-runtime-dir@0.service...
May 27 22:39:05 gentoo systemd[1]: Finished user-runtime-dir@0.service.
May 27 22:39:05 gentoo systemd[1]: Starting user@0.service...
May 27 22:39:05 gentoo (systemd)[13309]: pam_unix(systemd-user:session): session opened for user root(uid=0) by (uid=0)
May 27 22:39:05 gentoo systemd[13309]: Queued start job for default target default.target.
May 27 22:39:05 gentoo systemd[13309]: Created slice app.slice.
May 27 22:39:05 gentoo systemd[13309]: Reached target paths.target.
May 27 22:39:05 gentoo systemd[13309]: Reached target timers.target.
May 27 22:39:05 gentoo systemd[13309]: Starting dbus.socket...
May 27 22:39:05 gentoo systemd[13309]: Listening on dbus.socket.
May 27 22:39:05 gentoo systemd[13309]: Reached target sockets.target.
May 27 22:39:05 gentoo systemd[13309]: Reached target basic.target.
May 27 22:39:05 gentoo systemd[13309]: Reached target default.target.
May 27 22:39:05 gentoo systemd[13309]: Startup finished in 65ms.
May 27 22:39:05 gentoo systemd[1]: Started user@0.service.
May 27 22:39:05 gentoo systemd[1]: Started session-8.scope.
 ⊙ 22:40:58  ~
```

2.指定时间范围查找(-S,--since，-U,--until)

-S为从某个时间开始，-U为截止到某个时间。

时间格式为标准的年月日时分秒(YYYY-MM-DD HH:MM:SS)：“2023-05-27 18:00:00”。

如果不指定，则假定为从"00:00:00"开始，同时，还支持使用字符串的模式，比如"yesterday"、"today"、"tomorrow"分别表示昨天、今天、明天(当前时间的后一天)，详细用法可通过man systemd.time查阅。

比如，显示从昨天到现在的日志：

```
journalctl -S "yesterday"
```

```
 23:23:55  ~  journalctl -S "yesterday"  |head
May 26 06:47:33 gentoo sshd[1239]: pam_unix(sshd:session): session closed for user root
May 26 06:47:33 gentoo systemd-logind[1123]: Session 1 logged out. Waiting for processes to exit.
May 26 06:47:33 gentoo systemd[1]: session-1.scope: Deactivated successfully.
May 26 06:47:33 gentoo systemd[1]: session-1.scope: Consumed 59min 44.757s CPU time.
May 26 06:47:33 gentoo systemd-logind[1123]: Removed session 1.
May 26 06:48:28 gentoo sshd[3110]: pam_unix(sshd:session): session closed for user root
May 26 06:48:28 gentoo systemd-logind[1123]: Session 3 logged out. Waiting for processes to exit.
May 26 06:48:28 gentoo systemd[1]: session-3.scope: Deactivated successfully.
May 26 06:48:28 gentoo systemd[1]: session-3.scope: Consumed 7.909s CPU time.
May 26 06:48:28 gentoo systemd-logind[1123]: Removed session 3.
 23:24:06  ~
```

显示指定时间点到现在的日志：

```
journalctl -S "2023-05-21 18:00:00"
```

```
 23:26:07  ~  journalctl -S "2023-05-21 18:00:00" | head
May 21 18:38:53 gentoo systemd[1]: Starting systemd-tmpfiles-clean.service...
May 21 18:38:53 gentoo systemd[1]: systemd-tmpfiles-clean.service: Deactivated successfully.
May 21 18:38:53 gentoo systemd[1]: Finished systemd-tmpfiles-clean.service.
May 21 18:38:53 gentoo systemd[1]: run-credentials-systemd\x2dtmpfiles\x2dclean.service.mount: Deactivated successfully.
May 21 18:51:30 gentoo sshd[158771]: Accepted keyboard-interactive/pam for root from 192.168.1.3 port 19879 ssh2
May 21 18:51:30 gentoo sshd[158771]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 21 18:51:30 gentoo systemd-logind[1192]: New session 42 of user root.
May 21 18:51:30 gentoo systemd[1]: Started session-42.scope.
May 21 18:51:35 gentoo sz[158846]: [root] cka.pdf/ZMODEM: 471132 Bytes, 23661835 BPS
May 21 20:05:33 gentoo sshd[158771]: pam_unix(sshd:session): session closed for user root
 23:26:10  ~
```

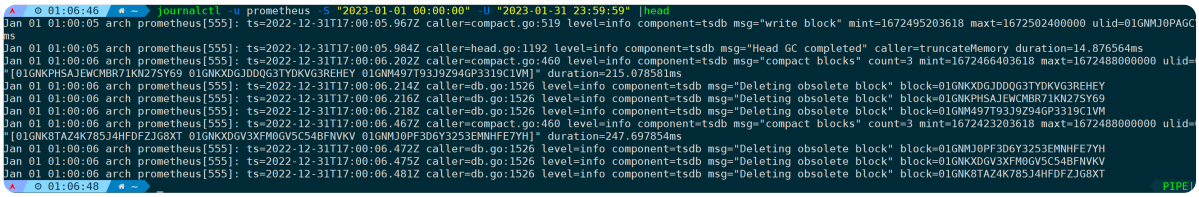
指定时间范围内的日志，比如查找从5月21到昨天的sshd服务日志：

```
journalctl -S "2023-05-21 18:00:00" -U yesterday -u sshd # -u 后面接服务名
```

```
 23:28:20  ~  journalctl -S "2023-05-21 18:00:00" -U yesterday -u sshd
May 22 15:47:05 arch sshd[14550]: Accepted password for root from [redacted] 197.48 port 11622 ssh2
May 22 15:47:05 arch sshd[14550]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 22 15:56:43 arch sshd[14647]: Accepted password for root from [redacted] 197.48 port 61242 ssh2
May 22 15:56:43 arch sshd[14647]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 22 21:57:42 arch sshd[14972]: Accepted password for root from [redacted].67.32 port 2492 ssh2
May 22 21:57:42 arch sshd[14972]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
-- Boot 91b890e06b654b49a54257184891e744 --
May 23 08:25:16 arch systemd[1]: Started OpenSSH Daemon.
May 23 08:25:20 arch sshd[990]: Server listening on 0.0.0.0 port 22.
May 23 08:25:20 arch sshd[990]: Server listening on :: port 22.
May 23 12:45:47 arch sshd[1539]: Accepted password for root from 192.168.1.3 port 9580 ssh2
May 23 12:45:47 arch sshd[1539]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 23 17:52:49 arch sshd[1880]: Accepted password for root from [redacted].179.137 port 49117 ssh2
May 23 17:52:49 arch sshd[1880]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 23 22:49:39 arch sshd[2252]: Accepted password for root from [redacted].67.32 port 2135 ssh2
May 23 22:49:39 arch sshd[2252]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 24 17:13:00 arch sshd[2741]: Accepted password for root from [redacted]179.137 port 19756 ssh2
May 24 17:13:00 arch sshd[2741]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 25 17:02:57 arch sshd[3338]: Accepted password for root from [redacted] 179.137 port 56503 ssh2
May 25 17:02:57 arch sshd[3338]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 25 18:32:26 arch sshd[3512]: Accepted password for root from [redacted].179.137 port 45545 ssh2
May 25 18:32:26 arch sshd[3512]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
 23:28:26  ~
```

查找2023年1月份的prometheus服务的日志：

```
journalctl -u prometheus -S "2023-01-01 00:00:00" -U "2023-01-31 23:59:59"
```



3.查询特定引导ID的日志(-b,--boot)

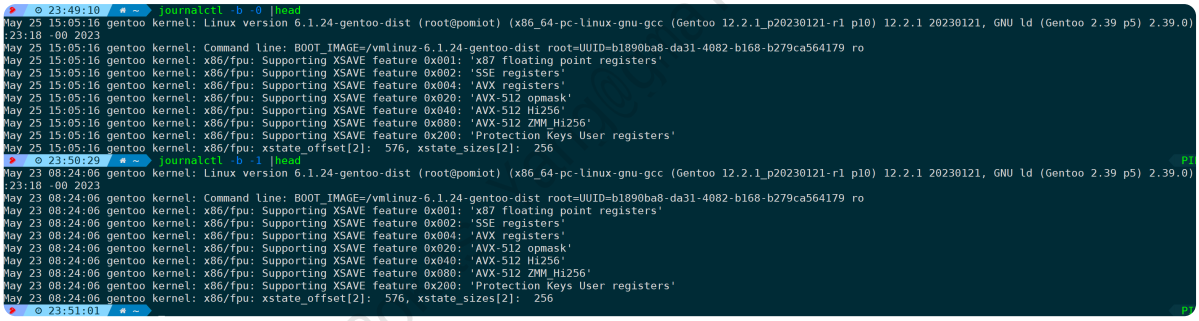
显示来自特定启动时的日志。

-0或者为空表示本次系统的日志：

```
 journalctl -b -0 # 显示本次系统启动时的日志
```

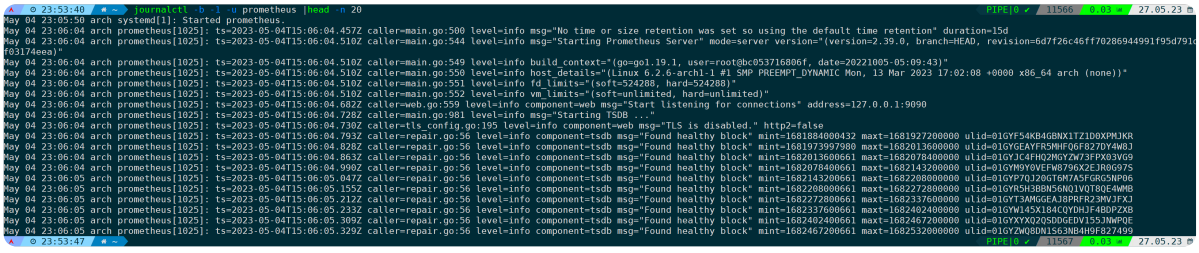
-1 表示上一次的系统启动的日志：

```
journalctl -b -1
```



-2就是上两次，顺序依次类推，那么同理，指定-b后，可以查找特定服务在上次启动后产生的日志，比如显示prometheus服务在上次系统启动后产生的日志可以是：

```
journalctl -b -1 -u prometheus
```



--list-boots可显示所有boot id列表：

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

```
 18:57:52  ~ journalctl --list-boots
IDX BOOT ID                          FIRST ENTRY                 LAST ENTRY
-29 bd872ff980c14071be90e2d6e43e46e2 Tue 2022-05-24 09:05:48 CST Sat 2022-05-28 13:49:32 CST
-28 6990345cb14549d88daa83bcfed9169b Sun 2022-05-29 11:44:16 CST Tue 2022-06-07 02:04:57 CST
-27 dc0b8d563e324ecba747f4c196aaa8d4 Tue 2022-06-07 02:46:32 CST Fri 2022-06-24 05:17:03 CST
-26 7510cd8e2e7b47619926f46093be4162 Fri 2022-06-24 17:26:08 CST Fri 2022-06-24 18:28:47 CST
-25 acf18b3add8a40909287e9d16e2c7a2c Fri 2022-06-24 18:28:55 CST Fri 2022-06-24 19:08:13 CST
-24 b2178c2e30904178908b94da546fb412 Fri 2022-06-24 19:09:57 CST Thu 2022-09-22 02:42:56 CST
-23 f6fced3a28424c52ac609e1baf6bb805 Thu 2022-09-22 02:47:00 CST Sat 2022-10-01 21:27:47 CST
-22 53a9a1c3f27246fd87e37b0fb1e4850a Sat 2022-10-01 21:45:59 CST Mon 2022-10-03 13:07:07 CST
-21 f4ece52b515e44a9ad10308008d46e7e Mon 2022-10-03 13:22:37 CST Thu 2022-10-06 21:58:46 CST
-20 6bcbe7f703074b03a4c806e3ffd60e80 Thu 2022-10-06 22:00:43 CST Sat 2022-10-08 05:54:49 CST
-19 874906559b7f4583b6150a5616f3bfe6 Sat 2022-10-08 05:57:35 CST Fri 2022-10-14 22:07:18 CST
-18 008dfccde22947449bd0384505376742 Fri 2022-10-14 22:09:16 CST Wed 2022-11-02 21:33:31 CST
-17 fdbb0dfab2cd453aa7b473eb7e7a153d Wed 2022-11-02 22:05:48 CST Fri 2022-11-11 13:40:31 CST
-16 2651b144594544955895d5c6801721648 Fri 2022-11-11 13:49:36 CST Sun 2022-11-20 05:38:07 CST
-15 a0b5baf657a148aaaf2b64feefbee21c Sat 2022-11-19 21:38:20 CST Tue 2023-01-17 20:37:20 CST
-14 821e72ec38154ff3b7c34d0db99f37b1 Tue 2023-01-17 20:37:38 CST Wed 2023-01-18 18:29:45 CST
-13 709bc21614a64157b6d1eebe36c0cf3f Wed 2023-01-18 18:32:19 CST Wed 2023-01-25 11:44:22 CST
-12 4a197eb813a44cac9a75464640e0b1b0 Wed 2023-01-25 20:55:47 CST Wed 2023-01-25 20:59:22 CST
-11 132652d2d3bd4a0ab732205025562c30 Wed 2023-01-25 21:09:42 CST Wed 2023-01-25 21:18:22 CST
-10 81e669da42904059b2272b4e75c3e84c Wed 2023-01-25 21:38:55 CST Thu 2023-01-26 18:01:22 CST
 -9 b8f41f5a560c4a5bb851368faae9e898 Thu 2023-01-26 18:25:22 CST Fri 2023-03-17 15:51:26 CST
 -8 f99a93283b104b16967783e160f2e014 Fri 2023-03-17 15:51:45 CST Sat 2023-04-15 10:11:57 CST
 -7 056213455b5e4f6f9dea07c233c808ec Sat 2023-04-15 10:12:18 CST Sun 2023-04-16 12:09:19 CST
 -6 dc9b2e2a8a9e4166a699a63c748e9322 Sun 2023-04-16 12:47:46 CST Tue 2023-04-18 20:34:45 CST
 -5 dd7a8c34b9a84d598a90658faf9767bd Tue 2023-04-18 20:36:13 CST Thu 2023-04-20 00:22:33 CST
 -4 7b34bd6a8ccc4b7886606477d3fa880b Thu 2023-04-20 15:57:58 CST Sat 2023-04-29 12:01:33 CST
 -3 282b653f8aff45989654220e708a9001 Sat 2023-04-29 12:48:31 CST Sat 2023-04-29 12:52:33 CST
 -2 06dd1eebfbbe4970bb59559cd3fcca51 Sat 2023-04-29 13:06:27 CST Thu 2023-05-04 23:03:22 CST
 -1 1ab5a0b32c7249ad9eace156dfa47211 Thu 2023-05-04 23:04:20 CST Tue 2023-05-23 07:39:22 CST
  0 91b890e06b654b49a54257184891e744 Tue 2023-05-23 08:23:46 CST Sun 2023-05-28 18:57:33 CST
 18:57:53  ~
```

4.查询指定服务的日志(-u, --unit)

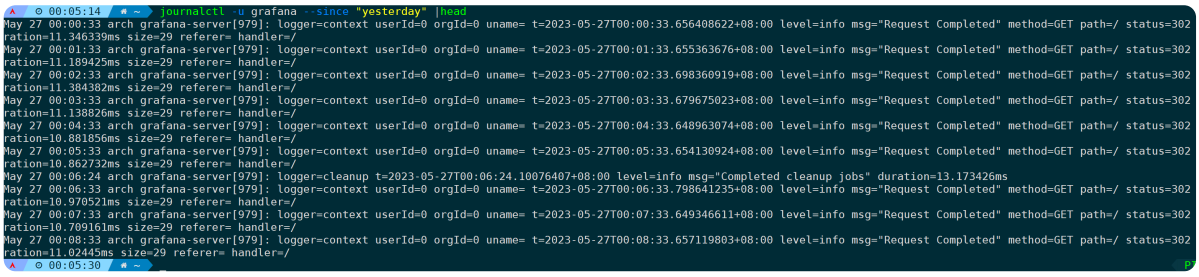
-u将指定systemd unit服务单元。

查找sshd服务的日志：

```
journalctl -u sshd
```

查询grafana服务的日志，并指定时间范围为今天到现在：

```
journalctl -u grafana --since "yesterday"
```



5.查找用户级别的服务日志(--user-unit)

这部分服务在systemd的user unit下管理，可通过**systemctl --user list-units**来展示用户级别下单元服务。

> 不指定这个--user-unit参数也没问题，默认是系统级别的日志，都会展示出来。

查找dbus服务日志：

```
journalctl --user-unit=dbus.socke
```

```
 00:18:37  ~  journalctl --user-unit=dbus.socket|head
May 24 20:52:12 arch systemd[71034]: Starting D-Bus User Message Bus Socket...
May 24 20:52:12 arch systemd[71034]: Listening on D-Bus User Message Bus Socket.
May 24 21:08:44 arch systemd[71034]: Closed D-Bus User Message Bus Socket.
May 24 21:18:16 arch systemd[71909]: Starting D-Bus User Message Bus Socket...
May 24 21:18:16 arch systemd[71909]: Listening on D-Bus User Message Bus Socket.
May 24 21:36:20 arch systemd[71909]: Closed D-Bus User Message Bus Socket.
May 25 15:22:37 arch systemd[103746]: Starting D-Bus User Message Bus Socket...
May 25 15:22:37 arch systemd[103746]: Listening on D-Bus User Message Bus Socket.
May 25 15:37:51 arch systemd[103746]: Closed D-Bus User Message Bus Socket.
May 25 16:39:22 arch systemd[106056]: Starting D-Bus User Message Bus Socket...
 00:18:42  ~
```

6.查找特定标识符的日志(-t, --identifier)

如果是自己写的程序，这个标识符是可自定义的，比如下面这段代码：

```
import logging

logger = logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)
logger.addHandler(logging.StreamHandler())
logger.setSyslogIdent("app")
```

假设这个程序服务名为"my-test-app"，但最后一条设置的日志标识符是"app"，那么通过-t参数查找标识符时应该指定"app"，而不是"my-test-app"。

默认情况下，日志标识符大多数都是程序名本身，比如查找标识符为prometheus的日志：

```
journalctl -t 'prometheus'
```

```
 00:30:23  ~  journalctl -t 'prometheus' |head
Oct 07 19:08:15 arch prometheus[1909]: ts=2022-10-07T11:08:15.598Z caller=main.go:456 level=error msg="Error loading config (--config.file=/usr/local/prometheus.yml)" file=/usr/lo
ometheus.yml: no such file or directory"
Oct 07 19:08:58 arch prometheus[1949]: ts=2022-10-07T11:08:58.628Z caller=main.go:456 level=error msg="Error loading config (--config.file=/usr/local/prometheus.yml)" file=/usr/lo
ometheus.yml: no such file or directory"
Oct 07 19:09:09 arch prometheus[1987]: ts=2022-10-07T11:09:09.890Z caller=main.go:500 level=info msg="No time or size retention was set so using the default time retention" durati
Oct 07 19:09:09 arch prometheus[1987]: ts=2022-10-07T11:09:09.890Z caller=main.go:544 level=info msg="Starting Prometheus Server" mode=server version="(version=2.39.0, branch=HEAD
f03174eea)"
Oct 07 19:09:09 arch prometheus[1987]: ts=2022-10-07T11:09:09.890Z caller=main.go:549 level=info build_context="(go=go1.19.1, user=root@bc053716806f, date=20221005-05:09:43)"
Oct 07 19:09:09 arch prometheus[1987]: ts=2022-10-07T11:09:09.890Z caller=main.go:550 level=info host_details="(Linux 5.19.13-arch1-1 #1 SMP PREEMPT_DYNAMIC Tue, 04 Oct 2022 14:36
Oct 07 19:09:09 arch prometheus[1987]: ts=2022-10-07T11:09:09.890Z caller=main.go:551 level=info fd_limits="(soft=524288, hard=524288)"
Oct 07 19:09:09 arch prometheus[1987]: ts=2022-10-07T11:09:09.890Z caller=main.go:552 level=info vm_limits="(soft=unlimited, hard=unlimited)"
Oct 07 19:09:09 arch prometheus[1987]: ts=2022-10-07T11:09:09.891Z caller=web.go:559 level=info component=web msg="Start listening for connections" address=127.0.0.1:9090
Oct 07 19:09:09 arch prometheus[1987]: ts=2022-10-07T11:09:09.892Z caller=main.go:981 level=info msg="Starting TSDB ..."
 00:35:38  ~  journalctl -u 'prometheus' |head
Oct 07 19:08:15 arch systemd[1]: Started prometheus.
Oct 07 19:08:15 arch prometheus[1909]: ts=2022-10-07T11:08:15.598Z caller=main.go:456 level=error msg="Error loading config (--config.file=/usr/local/prometheus.yml)" file=/usr/lo
ometheus.yml: no such file or directory"
Oct 07 19:08:15 arch systemd[1]: prometheus.service: Main process exited, code=exited, status=2/INVALIDARGUMENT
Oct 07 19:08:15 arch systemd[1]: prometheus.service: Failed with result 'exit-code'.
Oct 07 19:08:58 arch systemd[1]: Started prometheus.
Oct 07 19:08:58 arch prometheus[1949]: ts=2022-10-07T11:08:58.628Z caller=main.go:456 level=error msg="Error loading config (--config.file=/usr/local/prometheus.yml)" file=/usr/lo
ometheus.yml: no such file or directory"
Oct 07 19:08:58 arch systemd[1]: prometheus.service: Main process exited, code=exited, status=2/INVALIDARGUMENT
Oct 07 19:08:58 arch systemd[1]: prometheus.service: Failed with result 'exit-code'.
Oct 07 19:09:09 arch systemd[1]: Started prometheus.
Oct 07 19:09:09 arch prometheus[1987]: ts=2022-10-07T11:09:09.890Z caller=main.go:500 level=info msg="No time or size retention was set so using the default time retention" durati
 00:35:44  ~
```

和-u指定prometheus服务是有区别的，-u会记录整个服务生命周期开始到结束产生的日志，而-t只查找指定标识符产生的日志。

7.查找特定优先级的日志(-p, --priority)

-p可以精准的将各个优先级日志分门别类筛选出来，按消息优先级或优先级范围过滤输出。

取一个单一的数字或文本日志级别（即在0/"emerg "和7/"debug "之间），或一个数字/文本日志级别的范围，形式为FROM..TO，比如0..3表示取0到3级的日志。

日志等级一共分为如下8个级别：

数值	优先级	参数值	描述	示例
0	Emergency	emerg	System is unusable（系统不可用）	Severe Kernel BUG, systemd dumped core.This level should not be used by applications.
1	Alert	alert	Should be corrected immediately（应立即纠正）	Vital subsystem goes out of work. Data loss. **kernel: BUG: unable to handle kernel paging request at ffffc90403238ffc**
2	Critical	crit	Critical conditions（危机状态）	Crashes, coredumps. Like familiar flash: `systemd-coredump[25319]: Process 25310 (plugin-containe) of user 1000 dumped core` Failure in the system primary application, like X11.
3	Error	err	Error conditions（错误状态）	Not fatal error reported: `kernel: usb 1-3: 3:1: cannot get freq at ep 0x84`, `systemd[1]: Failed unmounting /var.`, `libvirtd[1720]: internal error: Failed to initialize a valid firewall backend`).
4	Warning	warning	Warning conditions（警告状态）	A non-root file system has only 1GB free. `org.freedesktop.Notifications[1860]: (process:5999): Gtk-WARNING **: Locale not supported by C library. Using the fallback 'C' locale`.
5	Notice	notice	Normal but significant condition（正常但值得注意的情况）	`systemd[1]: var.mount: Directory /var to mount over is not empty, mounting anyway`. `gcr-prompter[4997]: Gtk: GtkDialog mapped without a transient parent. This is discouraged`.
6	Informational	info	Normal operational messages that require no action（无需任何操作的正常信息）	`lvm[585]: 7 logical volume(s) in volume group "archvg" now active`.

数值	优先级	参数值	描述	示例
7	Debug	debug	debug-level messages (debug调试级别)	kdeinit5[1900]: powerdevil: Scheduling inhibition from ":1.14" "firefox" with cookie 13 and reason "screen"

日志等级详细可参考 RFC 5424 6.2.1 。

查找sshd服务Error级别的日志：

```
journalctl -p 3 -u sshd
```

```
 ⊙ 01:12:27  ~ journalctl -p 3 -u sshd |head
May 24 22:50:21 arch sshd[74736]: error: kex_exchange_identification: read: Connection reset by peer
May 25 22:40:07 arch sshd[117376]: fatal: Timeout before authentication for [redacted] 27.136 port 42974
May 25 23:37:43 arch sshd[119029]: fatal: Timeout before authentication for [redacted].27.136 port 59708
May 26 02:30:30 arch sshd[124063]: fatal: Timeout before authentication for [redacted].27.136 port 39320
May 26 05:47:57 arch sshd[124418]: error: kex_exchange_identification: read: Connection reset by peer
May 26 09:50:23 arch sshd[124485]: error: kex_exchange_identification: Connection closed by remote host
May 26 09:54:38 arch sshd[124495]: error: kex_exchange_identification: Connection closed by remote host
May 26 12:16:09 arch sshd[124772]: error: kex_exchange_identification: Connection closed by remote host
May 26 22:50:42 arch sshd[127066]: error: kex_exchange_identification: banner line contains invalid characters
May 27 04:09:40 arch sshd[127186]: error: kex_exchange_identification: Connection closed by remote host
 ⊙ 01:12:31  ~
```

查找Emergency级别的所有日志：

```
journalctl -p emerg
```

查找fail2ban服务0(Emergency)到5(Notice)等级的日志：

```
journalctl -p 0..5 -u fail2ban
```

```
 ⊙ 01:13:40  ~ journalctl -p 0..5 -u fail2ban
Jul 29 21:16:15 arch systemd[1]: fail2ban.service: Main process exited, code=exited, status=255/EXCEPTION
Jul 29 21:16:15 arch systemd[1]: fail2ban.service: Failed with result 'exit-code'.
Sep 22 02:42:55 arch systemd[1]: fail2ban.service: Consumed 41min 33.867s CPU time.
-- Boot 2651b14459454495895d5c6801721648 --
Nov 20 05:38:06 arch systemd[1]: fail2ban.service: Consumed 13min 15.772s CPU time.
-- Boot b8f41f5a560c4a5bb851368faae9e898 --
Mar 17 15:50:48 arch systemd[1]: fail2ban.service: Consumed 1h 26min 33.828s CPU time.
-- Boot f99a93283b104b16967783e160f2e014 --
Apr 15 10:11:55 arch systemd[1]: fail2ban.service: Consumed 48min 34.718s CPU time.
 ⊙ 01:15:14  ~ _
```

查找上一次系统启动，标识符为"kernel"的0到2级以及第4级的日志：

```
journalctl -b -1 -p 0..2 -p 4 -t 'kernel'
```

```
 ⊙ 01:18:47  ~ journalctl -b -1 -p 0..2 -p 4 -t 'kernel'
May 04 23:04:20 arch kernel: core: CPUID marked event: 'cpu cycles' unavailable
May 04 23:04:20 arch kernel: core: CPUID marked event: 'instructions' unavailable
May 04 23:04:20 arch kernel: core: CPUID marked event: 'bus cycles' unavailable
May 04 23:04:20 arch kernel: core: CPUID marked event: 'cache references' unavailable
May 04 23:04:20 arch kernel: core: CPUID marked event: 'cache misses' unavailable
May 04 23:04:20 arch kernel: core: CPUID marked event: 'branch instructions' unavailable
May 04 23:04:20 arch kernel: core: CPUID marked event: 'branch misses' unavailable
May 04 23:04:20 arch kernel:   #9 #10 #11 #12 #13 #14 #15 #16
May 04 23:04:20 arch kernel:  #17 #18 #19 #20 #21 #22 #23 #24
May 04 23:04:20 arch kernel:  #25 #26 #27 #28 #29 #30 #31 #32
May 04 23:04:20 arch kernel:  #33 #34 #35 #36 #37 #38 #39 #40
May 04 23:04:20 arch kernel:  #41 #42 #43 #44 #45 #46 #47 #48
May 04 23:04:20 arch kernel:  #49 #50 #51 #52 #53 #54 #55 #56
May 04 23:04:20 arch kernel:  #57 #58 #59 #60 #61 #62 #63
May 04 23:04:34 arch kernel: piix4_smbus 0000:00:07.3: SMBus Host Controller not enabled!
May 04 23:04:37 arch kernel: platform regulatory.0: Direct firmware load for regulatory.db failed with error -2
May 10 20:17:24 arch kernel: watchdog: BUG: soft lockup - CPU#11 stuck for 28s! [swapper/11:0]
May 10 20:17:25 arch kernel: Modules linked in: tls xt_conntrack xt_MASQUERADE nf_conntrack_netlink nfnetlink iptable_nat nf_nat nf_conntrack nf_defra
May 10 20:17:27 arch kernel: CPU: 11 PID: 0 Comm: swapper/11 Not tainted 6.2.6-arch1-1 #1 bdb4a56fad97b891ecbccb5d194884721c85b4d2
May 10 20:17:27 arch kernel: Hardware name: VMware, Inc. VMware Virtual Platform/440BX Desktop Reference Platform, BIOS 6.00 11/12/2020
May 10 20:17:27 arch kernel: RIP: 0010:_raw_spin_unlock_irqrestore+0x1d/0x40
May 10 20:17:27 arch kernel: Code: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 f3 0f 1e fa 0f 1f 44 00 00 c6 07 00 0f 1f 00 f7 c6 00 02 00 00 74 06 fb 0f 1
May 10 20:17:27 arch kernel: RSP: 0018:ffffbd13804fce60 EFLAGS: 00000206
May 10 20:17:27 arch kernel: RAX: 0000000000000040 RBX: 0000000000000002 RCX: 0000000000000040
May 10 20:17:27 arch kernel: RDX: 0000000000000040 RSI: 0000000000000202 RDI: ffff9293cc64f730
May 10 20:17:27 arch kernel: RBP: ffff9293cc64f730 R08: 000000000000003f R09: ffff929adf0e31a8
May 10 20:17:27 arch kernel: R10: 0000000000000002 R11: 00000000d1b75ea4 R12: ffffffffa8e02ae0
May 10 20:17:27 arch kernel: R13: 0000000000000202 R14: ffffbd13804fcf08 R15: ffff929adf0e3180
May 10 20:17:27 arch kernel: FS:  0000000000000000(0000) GS:ffff929adf0c0000(0000) knlGS:0000000000000000
May 10 20:17:27 arch kernel: CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
May 10 20:17:27 arch kernel: CR2: 000055f5e95688b0 CR3: 00000001049b8006 CR4: 0000000000770ee0
May 10 20:17:27 arch kernel: PKRU: 55555554
May 10 20:17:27 arch kernel: Call Trace:
May 10 20:17:27 arch kernel:  <IRQ>
May 10 20:17:27 arch kernel:  __percpu_counter_sum_mask+0x5b/0x70
May 10 20:17:27 arch kernel:  ? __pfx_writeout_period+0x10/0x10
May 10 20:17:27 arch kernel:  fprop_new_period+0x10/0x60
May 10 20:17:27 arch kernel:  writeout_period+0x3d/0x80
May 10 20:17:27 arch kernel:  call_timer_fn+0x24/0x130
May 10 20:17:27 arch kernel:  ? __pfx_writeout_period+0x10/0x10
```

8.按照日志设备进行过滤和查询(--facility)

这里的设备是指生成日志消息的系统组件或服务。

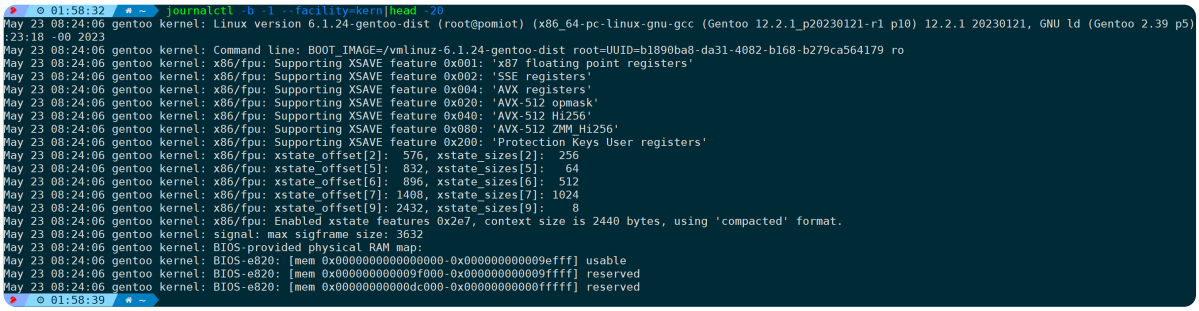
通过**journalctl --facility=help**可以看到当前有哪些设备。

常用设备解读：

- **kernel**：内核产生的日志消息。
- **user**：与用户操作和登录相关的日志消息。
- **mail**：与邮件系统相关的日志消息。
- **auth**：与身份验证和授权相关的日志消息。
- **syslog**：由 syslog 守护程序生成的日志消息。
- **lpr**：与打印系统相关的日志消息。
- **news**：与新闻服务器相关的日志消息。
- **uucp**：与 UUCP（Unix to Unix Copy）系统相关的日志消息。
- **cron**：与定时任务（cron）相关的日志消息。
- **authpriv**：与身份验证和授权的私有信息相关的日志消息。
- **ftp**：与文件传输协议（FTP）服务器相关的日志消息。
- **ntp**：与网络时间协议（NTP）服务器相关的日志消息。

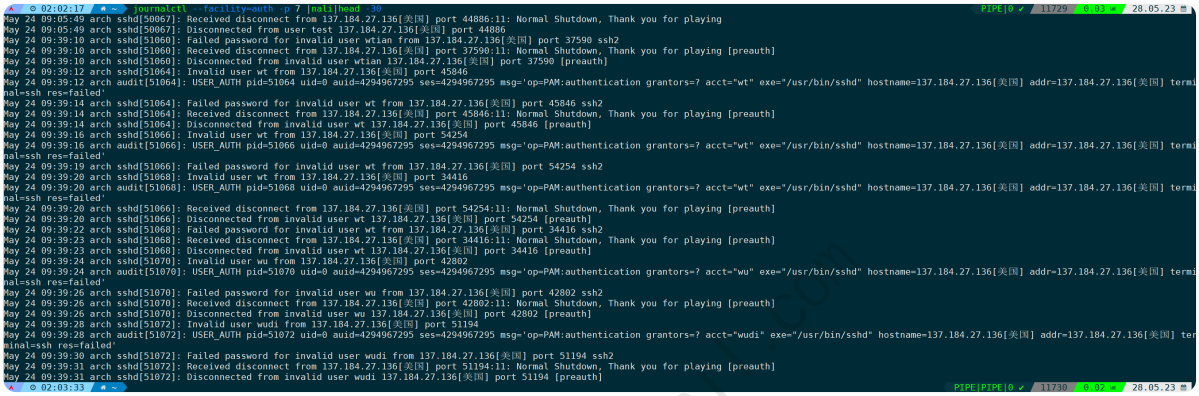
比如，我想知道上一次启动内核产生的日志：

```
journalctl -b -1 --facility=kern
```



又或者想知道身份验证和授权相关的日志，并且debug级别输出：

```
journalctl --facility=auth -p 7
```

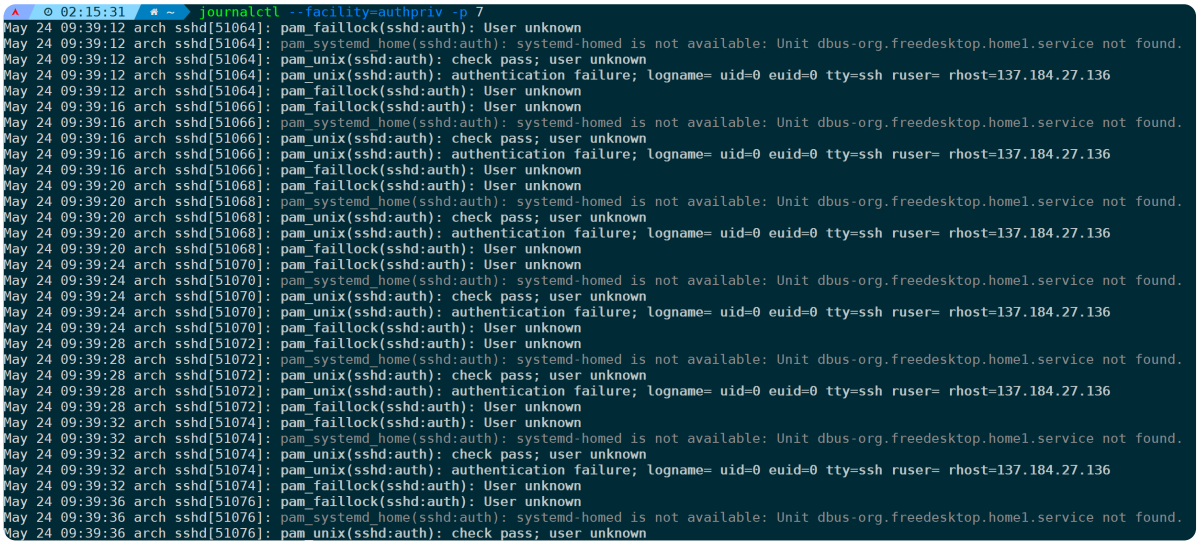


可以看到很多sshd登录失败的日志，此系统被ssh暴力穷举过，但显然都失败了，如果有安全防护需求，可以参考我写的 **fail2ban配置说明** 。

也可以通过**authpriv**查找与身份验证和授权的私有信息相关的日志消息：

```
journalctl --facility=authpriv -p 7
```

> 会显示系统内部一些私有日志信息，比如pam模块的验证日志。



从输出可以看到，在用户名这一层已经校验未找到，不会继续往下校验密码字段了。

9.使用正则表达式过滤日志(-g,--grep)

作用域是MESSAGE字段的内容，支持Perl正则，可以通过**man pcre2pattern**来查看具体语法。

同时，如果写的表达式都是小写，那就不区分大小写，如果包含大写就会区分大小写，如果不想区分大小写可以使用**--case-sensitive=false**参数来生效，比如下面的几种情况：

- --grep "abc"，是不区分大小写的；
- --grep "Abc"，区分大小写，只过滤匹配Abc的日志；
- --grep "Abc" --case-sensitive=false，仍然不区分大小写。

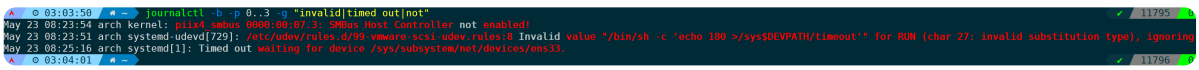
过滤sshd服务错误用户名密码的日志：

```
journalctl -u sshd --grep 'Failed password'
```

```
02:43:09 ~ journalctl -u sshd --grep 'Failed password'
May 24 09:39:10 arch sshd[51060]: Failed password for invalid user wtian from 137.184.27.136 port 37590 ssh2
May 24 09:39:14 arch sshd[51064]: Failed password for invalid user wt from 137.184.27.136 port 45846 ssh2
May 24 09:39:19 arch sshd[51066]: Failed password for invalid user wt from 137.184.27.136 port 54254 ssh2
May 24 09:39:22 arch sshd[51068]: Failed password for invalid user wt from 137.184.27.136 port 34416 ssh2
May 24 09:39:26 arch sshd[51070]: Failed password for invalid user wu from 137.184.27.136 port 42802 ssh2
May 24 09:39:30 arch sshd[51072]: Failed password for invalid user wudi from 137.184.27.136 port 51194 ssh2
May 24 09:39:34 arch sshd[51074]: Failed password for invalid user wugang from 137.184.27.136 port 59616 ssh2
May 24 09:39:38 arch sshd[51076]: Failed password for invalid user wuhong from 137.184.27.136 port 39754 ssh2
May 24 09:39:43 arch sshd[51078]: Failed password for invalid user wuhz from 137.184.27.136 port 48140 ssh2
May 24 09:39:46 arch sshd[51080]: Failed password for invalid user wuhz from 137.184.27.136 port 56520 ssh2
May 24 09:39:50 arch sshd[51082]: Failed password for invalid user wujie from 137.184.27.136 port 36734 ssh2
May 24 09:39:54 arch sshd[51084]: Failed password for invalid user wujun from 137.184.27.136 port 45110 ssh2
May 24 09:39:57 arch sshd[51086]: Failed password for invalid user wunian from 137.184.27.136 port 53524 ssh2
May 24 09:40:02 arch sshd[51088]: Failed password for invalid user wuning from 137.184.27.136 port 33716 ssh2
May 24 09:40:06 arch sshd[51090]: Failed password for invalid user wuning from 137.184.27.136 port 42074 ssh2
May 24 09:40:09 arch sshd[51092]: Failed password for invalid user wuning from 137.184.27.136 port 50458 ssh2
May 24 09:40:14 arch sshd[51094]: Failed password for invalid user wupeng from 137.184.27.136 port 58908 ssh2
May 24 09:40:18 arch sshd[51098]: Failed password for invalid user wupeng from 137.184.27.136 port 39046 ssh2
May 24 09:40:21 arch sshd[51100]: Failed password for invalid user wuql from 137.184.27.136 port 47434 ssh2
May 24 09:40:26 arch sshd[51102]: Failed password for invalid user wuql from 137.184.27.136 port 55832 ssh2
May 24 09:40:30 arch sshd[51104]: Failed password for invalid user wuql from 137.184.27.136 port 35996 ssh2
May 24 09:40:33 arch sshd[51106]: Failed password for invalid user wuql from 137.184.27.136 port 44400 ssh2
May 24 09:40:38 arch sshd[51108]: Failed password for invalid user wuyongkun from 137.184.27.136 port 52784 ssh2
May 24 09:40:41 arch sshd[51110]: Failed password for invalid user wuzhikun from 137.184.27.136 port 32926 ssh2
May 24 09:40:45 arch sshd[51112]: Failed password for invalid user ww from 137.184.27.136 port 41354 ssh2
May 24 09:40:50 arch sshd[51114]: Failed password for invalid user wwb from 137.184.27.136 port 49750 ssh2
May 24 09:40:53 arch sshd[51116]: Failed password for invalid user wwb from 137.184.27.136 port 58150 ssh2
May 24 09:40:57 arch sshd[51118]: Failed password for invalid user wwh from 137.184.27.136 port 38308 ssh2
May 24 09:41:01 arch sshd[51120]: Failed password for invalid user wwj from 137.184.27.136 port 46716 ssh2
May 24 09:41:05 arch sshd[51122]: Failed password for invalid user wwj from 137.184.27.136 port 55108 ssh2
May 24 09:41:09 arch sshd[51124]: Failed password for invalid user wwj from 137.184.27.136 port 35288 ssh2
May 24 09:41:14 arch sshd[51126]: Failed password for invalid user www from 137.184.27.136 port 43680 ssh2
May 24 09:41:17 arch sshd[51128]: Failed password for invalid user wwwadmin from 137.184.27.136 port 52072 ssh2
May 24 09:41:21 arch sshd[51130]: Failed password for invalid user www from 137.184.27.136 port 60464 ssh2
May 24 09:41:26 arch sshd[51132]: Failed password for invalid user www-data from 137.184.27.136 port 40642 ssh2
May 24 09:41:29 arch sshd[51134]: Failed password for invalid user wwwroot from 137.184.27.136 port 49038 ssh2
May 24 09:41:33 arch sshd[51136]: Failed password for invalid user www from 137.184.27.136 port 57438 ssh2
May 24 09:41:37 arch sshd[51138]: Failed password for invalid user www from 137.184.27.136 port 37588 ssh2
May 24 09:41:41 arch sshd[51140]: Failed password for invalid user wwx from 137.184.27.136 port 46002 ssh2
```

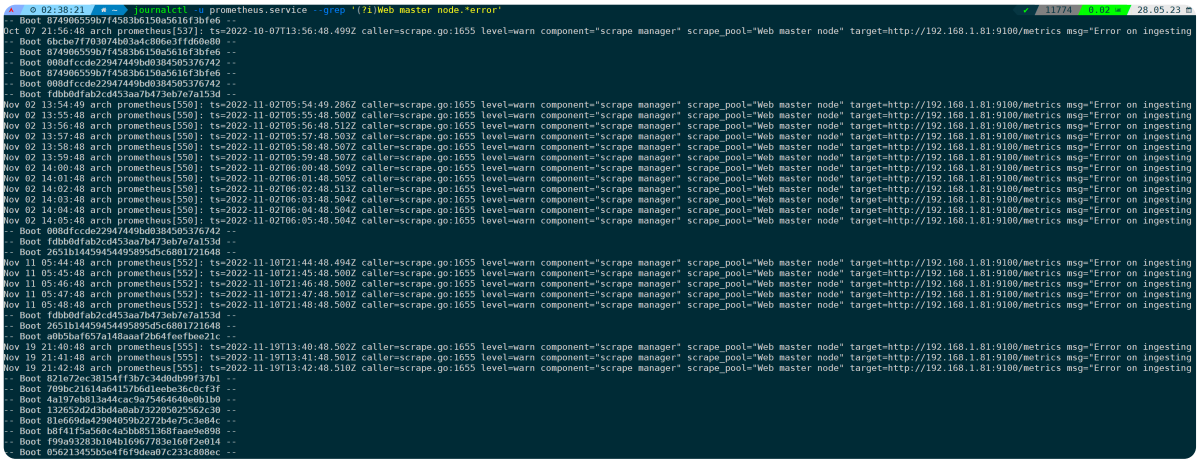
过滤优先级为0-3并且包含关键词"invalid"、"timed out"、"not"的日志:

```
journalctl -b -p 0..3 -g "invalid|timed out|not"
```



过滤prometheus服务的master节点的错误日志：

```
journalctl -u prometheus.service --grep '(?i)Web master node.*error'
```

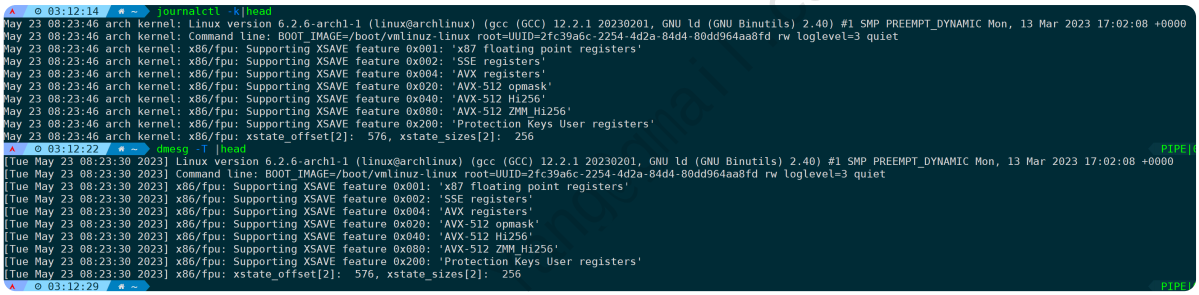


这里用到了(?i)，正则里面的不区分大小写的作用，不要误以为上面的大小写逻辑有问题。

10.显示内核日志(-k, --dmesg)

此参数将只显示内核级别的日志：

```
journalctl -k
```



仔细看和dmesg打印的日志结果是一样的，但机制来源不一样，**journalctl -k会从systemd-journald服务**收集的日志中过滤出内核日志，而dmesg是直接访问内核缓存区(**kernel ring buffer**)并把日志输出出来。

三、输出选项详解

1.控制日志输出格式(-o, --output)

格式	含义
short	默认，产生的输出与传统的syslog文件的格式基本相同，每条日志显示一行。
short-full	和short非常相似，但显示的是--since=和--until=选项接受的格式的时间戳，与短输出模式下显示的时间戳信息不同，该模式在输出中包括工作日、年份和时区信息。
short-iso	和short非常相似，但显示的是ISO 8601标准的时间戳（YYYY-MM-DDThh:mm:ss）。
short-iso-precise	如同short-iso，但包括完整的微秒级精度。

格式	含义
short-monotonic	单调递增时间，时间格式为相对时间。
short-delta	与short-monotonic一样，但包括与前一条的时间差，不可靠的时间差会用 "*"来标记。
short-unix	显示的是自1970年1月1日UTC以来的秒数，即UNIX时间戳，精度为微秒。
verbose	显示所有字段的完整结构的条目项目。
export	将日志序列化为适合备份和网络传输的二进制（但主要是基于文本的）流。要将二进制流导入到journald格式使用**man systemd-journal-remote**查看用法。
json	json格式输出，可通过**man Journal JSON Format**查看用法。
json-pretty	将条目格式化为JSON数据结构，但将其格式化为多行，以便使其更易读。
json-sse	将条目格式化为JSON数据结构，但将其包装成适合服务器发送事件的格式。
json-seq	将条目格式化为JSON数据结构，但前缀为ASCII记录分隔符（0x1E），后缀为ASCII换行符（0x0A），符合"application/json-seq"。
cat	生成一个非常简洁的输出，只显示每个日志条目的实际信息，没有元数据，甚至没有时间戳。如果与--output-fields选项结合使用，将为每条日志记录输出指定的字段。
with-unit	与short-full类似，但在单元和用户单元名称前加上前缀，而不是传统的syslog标识符。在使用模板化实例时很有用，因为它将在单元名称中包括参数名称。

1）short

默认情况下输出格式，比如输出今天的sshd服务日志：

```
journalctl -u sshd -S today -o short
```

```
 ⊙ 14:40:49  ~ journalctl -u sshd -S today
May 28 01:46:29 arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
May 28 01:46:29 arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 01:47:13 arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
May 28 01:47:15 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
May 28 01:47:19 arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
May 28 14:01:03 arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
May 28 14:01:03 arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
 ⊙ 14:41:00  ~ journalctl -u sshd -S today -o short
May 28 01:46:29 arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
May 28 01:46:29 arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 01:47:13 arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
May 28 01:47:15 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
May 28 01:47:19 arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
May 28 14:01:03 arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
May 28 14:01:03 arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
 ⊙ 14:41:02  ~
```

2）short-full

在short的基础上补全年份、时区信息，和--since=和--until=选项的时间戳适配：

```
journalctl -u sshd -S today -o short-full
```

```
⊙ 14:41:02 ~ journalctl -u sshd -S today -o short-full
Sun 2023-05-28 01:46:29 CST arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
Sun 2023-05-28 01:46:29 CST arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
Sun 2023-05-28 01:47:13 CST arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
Sun 2023-05-28 01:47:15 CST arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
Sun 2023-05-28 01:47:19 CST arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
Sun 2023-05-28 14:01:03 CST arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
Sun 2023-05-28 14:01:03 CST arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
⊙ 14:44:25 ~
```

3）short-iso

以ISO 8601标准时间戳（YYYY-MM-DDThh:mm:ss）显示：

```
journalctl -u sshd -S today -o short-iso
```

```
⊙ 14:44:25 ~ journalctl -u sshd -S today -o short-iso
2023-05-28T01:46:29+0800 arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
2023-05-28T01:46:29+0800 arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
2023-05-28T01:47:13+0800 arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
2023-05-28T01:47:15+0800 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
2023-05-28T01:47:19+0800 arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
2023-05-28T14:01:03+0800 arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
2023-05-28T14:01:03+0800 arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
⊙ 14:46:55 ~
```

4）short-iso-precise

在short-iso基础上加上完整的微秒级精度：

```
journalctl -u sshd -S today -o short-iso-precise
```

```
⊙ 14:48:23 ~ journalctl -u sshd -S today -o short-iso-precise
2023-05-28T01:46:29.419024+0800 arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
2023-05-28T01:46:29.421466+0800 arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
2023-05-28T01:47:13.906563+0800 arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
2023-05-28T01:47:15.777475+0800 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
2023-05-28T01:47:19.817946+0800 arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
2023-05-28T14:01:03.725509+0800 arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
2023-05-28T14:01:03.726649+0800 arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
⊙ 14:48:24 ~
```

5）short-monotonic

单调递增时间，时间格式为相对时间：

```
journalctl -u sshd -S today -o short-monotonic
```

```
⊙ 14:48:24 ~ journalctl -u sshd -S today -o short-monotonic
[408178.607941] arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
[408178.610222] arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
[408223.095480] arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
[408224.966429] arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
[408229.006792] arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
[452252.914314] arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
[452252.915384] arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
⊙ 14:50:24 ~
```

6）short-delta

与short-monotonic一样，但包括与前一条的时间差，不可靠的时间差会用 * 来标记：

```
journalctl -u sshd -S today -o short-delta
```

```
⊙ 14:50:54 ~ journalctl -u sshd -S today -o short-delta
[408178.607941              ] arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
[408178.610222 <    0.002281 >] arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
[408223.095480 <   44.485258 >] arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
[408224.966429 <    1.870949 >] arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
[408229.006792 <    4.040363 >] arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
[452252.914314 <44023.907522 >] arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
[452252.915384 <    0.001070 >] arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
⊙ 14:53:29 ~
```

7）short-unix

显示的是自1970年1月1日UTC以来的秒数，即UNIX时间戳，精度为微秒：

```
journalctl -u sshd -S today -o short-unix
```

```
 ⊙ 14:53:29  # ~ journalctl -u sshd -S today -o short-unix
1685209589.419024 arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
1685209589.421466 arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
1685209633.906563 arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
1685209635.777475 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
1685209639.817946 arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
1685253663.725509 arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
1685253663.726649 arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
 ⊙ 14:58:51  # ~
```

8）verbose

显示所有字段的完整结构的条目项目：

```
journalctl -u sshd -S today -o verbose
```

```
 ⊙ 15:02:21  # ~ journalctl -u sshd -S today -o verbose
Sun 2023-05-28 01:46:29.419024 CST [s=9de5c4069e3b4a5182f62465ddd61d5b;i=193f61;b=91b890e06b654b49a54257184891e744;m=5f09574745;t=5fcb06ef30ce4;x=401ab83427cda50]
    _BOOT_ID=91b890e06b654b49a54257184891e744
    _MACHINE_ID=b8fd7a061aca4427b8f4f27e474d4989
    _HOSTNAME=arch
    _RUNTIME_SCOPE=system
    PRIORITY=6
    _UID=0
    _GID=0
    _SYSTEMD_SLICE=system.slice
    _CAP_EFFECTIVE=1ffffffffff
    SYSLOG_FACILITY=4
    _TRANSPORT=syslog
    SYSLOG_IDENTIFIER=sshd
    _COMM=sshd
    _EXE=/usr/bin/sshd
    _SYSTEMD_CGROUP=/system.slice/sshd.service
    _SYSTEMD_UNIT=sshd.service
    _SYSTEMD_INVOCATION_ID=21174b37b37c4cd5b4d20c828cbffaa3
    _CMDLINE="sshd: root [priv]"
    SYSLOG_PID=7052
    SYSLOG_TIMESTAMP=May 28 01:46:29
    MESSAGE=Accepted password for root from 192.168.1.3 port 56322 ssh2
    _PID=7052
    _SOURCE_REALTIME_TIMESTAMP=1685209589419024
Sun 2023-05-28 01:46:29.421466 CST [s=9de5c4069e3b4a5182f62465ddd61d5b;i=193f62;b=91b890e06b654b49a54257184891e744;m=5f0957502e;t=5fcb06ef315cc;x=1f2b32c96c965b6e]
    _BOOT_ID=91b890e06b654b49a54257184891e744
    _MACHINE_ID=b8fd7a061aca4427b8f4f27e474d4989
    _HOSTNAME=arch
    _RUNTIME_SCOPE=system
    PRIORITY=6
    _UID=0
    _GID=0
    _SYSTEMD_SLICE=system.slice
    _CAP_EFFECTIVE=1ffffffffff
    _TRANSPORT=syslog
    SYSLOG_IDENTIFIER=sshd
    _COMM=sshd
    _EXE=/usr/bin/sshd
    _SYSTEMD_CGROUP=/system.slice/sshd.service
    _SYSTEMD_UNIT=sshd.service
    _SYSTEMD_INVOCATION_ID=21174b37b37c4cd5b4d20c828cbffaa3
    _CMDLINE="sshd: root [priv]"
    SYSLOG_FACILITY=10
```

字段含义可通过man 7 systemd.journal-fields来查看。

9）export

将日志序列化为适合备份和网络传输的二进制（但主要是基于文本的）流：

```
journalctl -u sshd -S today -o export
```

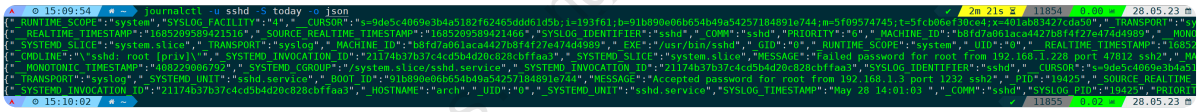
```
⊙ 15:07:30  ~  journalctl -u sshd -S today -o export
__CURSOR=s=9de5c4069e3b4a5182f62465ddd61d5b;i=193f61;b=91b890e06b654b49a54257184891e744;m=5f09574745;t=5fcb06ef30ce4;x=401ab83427cda50
__REALTIME_TIMESTAMP=1685209589419236
__MONOTONIC_TIMESTAMP=408178607941
_BOOT_ID=91b890e06b654b49a54257184891e744
_MACHINE_ID=b8fd7a061aca4427b8f4f27e474d4989
_HOSTNAME=arch
_RUNTIME_SCOPE=system
PRIORITY=6
_UID=0
_GID=0
_SYSTEMD_SLICE=system.slice
_CAP_EFFECTIVE=1ffffffffff
SYSLOG_FACILITY=4
_TRANSPORT=syslog
SYSLOG_IDENTIFIER=sshd
_COMM=sshd
_EXE=/usr/bin/sshd
_SYSTEMD_CGROUP=/system.slice/sshd.service
_SYSTEMD_UNIT=sshd.service
_SYSTEMD_INVOCATION_ID=21174b37b37c4cd5b4d20c828cbffaa3
_CMDLINE="sshd: root [priv]"
SYSLOG_PID=7052
SYSLOG_TIMESTAMP=May 28 01:46:29
MESSAGE=Accepted password for root from 192.168.1.3 port 56322 ssh2
_PID=7052
_SOURCE_REALTIME_TIMESTAMP=1685209589419024

__CURSOR=s=9de5c4069e3b4a5182f62465ddd61d5b;i=193f62;b=91b890e06b654b49a54257184891e744;m=5f0957502e;t=5fcb06ef315cc;x=1f2b32c96c965b6e
__REALTIME_TIMESTAMP=1685209589421516
__MONOTONIC_TIMESTAMP=408178610222
_BOOT_ID=91b890e06b654b49a54257184891e744
_MACHINE_ID=b8fd7a061aca4427b8f4f27e474d4989
_HOSTNAME=arch
_RUNTIME_SCOPE=system
PRIORITY=6
_UID=0
_GID=0
_SYSTEMD_SLICE=system.slice
_CAP_EFFECTIVE=1ffffffffff
_TRANSPORT=syslog
SYSLOG_IDENTIFIER=sshd
_COMM=sshd
_EXE=/usr/bin/sshd
```

10）json

json格式输出：

```
journalctl -u sshd -S today -o json
```



11）json-pretty

将条目格式化为JSON数据结构，但将其格式化为多行，以便使其更易读。

比如查询上次启动时优先级为2(Critical)的错误日志，json-pretty格式输出：

```
journalctl -b -1 -p 2 -o json-pretty
```

```
⊙ 15:14:20  ~  journalctl -b -1 -p 2 -o json-pretty
{
        "SYSLOG_FACILITY" : "0",
        "SYSLOG_IDENTIFIER" : "kernel",
        "__REALTIME_TIMESTAMP" : "1683721044744751",
        "_TRANSPORT" : "kernel",
        "_HOSTNAME" : "arch",
        "_SOURCE_MONOTONIC_TIMESTAMP" : "508393877655",
        "_RUNTIME_SCOPE" : "system",
        "__CURSOR" : "s=9de5c4069e3b4a5182f62465ddd61d5b;i=1727e9;b=1ab5a0b32c7249ad9eace156dfa47211;m=765ec30bfa;t=5fb55dac5da2f;x=b56b6b890bea3378",
        "MESSAGE" : "watchdog: BUG: soft lockup - CPU#11 stuck for 28s! [swapper/11:0]",
        "PRIORITY" : "0",
        "__MONOTONIC_TIMESTAMP" : "508395981818",
        "_MACHINE_ID" : "b8fd7a061aca4427b8f4f27e474d4989",
        "_BOOT_ID" : "1ab5a0b32c7249ad9eace156dfa47211"
}
⊙ 15:14:32  ~  journalctl -b -1 -p 2 -o json | jq
{
  "_TRANSPORT": "kernel",
  "MESSAGE": "watchdog: BUG: soft lockup - CPU#11 stuck for 28s! [swapper/11:0]",
  "__REALTIME_TIMESTAMP": "1683721044744751",
  "__MONOTONIC_TIMESTAMP": "508395981818",
  "_HOSTNAME": "arch",
  "_BOOT_ID": "1ab5a0b32c7249ad9eace156dfa47211",
  "SYSLOG_FACILITY": "0",
  "_SOURCE_MONOTONIC_TIMESTAMP": "508393877655",
  "__CURSOR": "s=9de5c4069e3b4a5182f62465ddd61d5b;i=1727e9;b=1ab5a0b32c7249ad9eace156dfa47211;m=765ec30bfa;t=5fb55dac5da2f;x=b56b6b890bea3378",
  "_MACHINE_ID": "b8fd7a061aca4427b8f4f27e474d4989",
  "PRIORITY": "0",
  "_RUNTIME_SCOPE": "system",
  "SYSLOG_IDENTIFIER": "kernel"
}
⊙ 15:14:42  ~
```

类似于将json输出递交给jq命令格式化输出一遍。

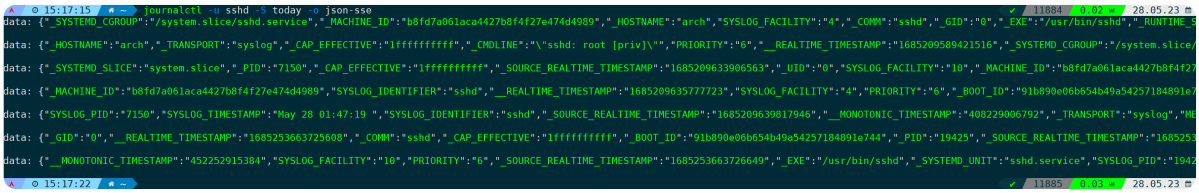
当MESSAGE长度过大，在终端展示不全时，可以加上--all参数并重定向写入到文件，查看文件的方式来看完整的MESSAGE字段信息，比如：

```
journalctl -b -1 -p 2 -o json-pretty --all > boot-1.json
```

12）json-sse

将条目格式化为JSON数据结构，但将其包装成适合服务器发送事件的格式：

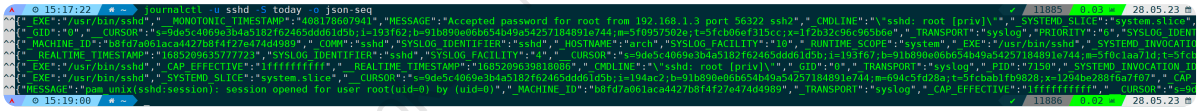
```
journalctl -u sshd -S today -o json-sse
```



13）json-seq

将条目格式化为JSON数据结构，但前缀为ASCII记录分隔符（0x1E），后缀为ASCII换行符（0x0A），符合"application/json-seq"格式标准：

```
journalctl -u sshd -S today -o json-seq
```



14）cat

生成一个非常简洁的输出，只显示每个日志条目的实际信息，没有元数据，甚至没有时间戳：

```
journalctl -u sshd -S today -o cat
```

```
journalctl -u sshd -S today -o cat
Accepted password for root from 192.168.1.3 port 56322 ssh2
pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
Failed password for root from 192.168.1.228 port 47812 ssh2
Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
Accepted password for root from 192.168.1.3 port 1232 ssh2
pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
```

15）with-unit

与short-full类似，但在单元和用户单元名称前加上前缀，而不是传统的syslog标识符：

```
journalctl -u sshd -S today -o with-unit
```

```
 ⊙ 15:21:18  ~ journalctl -u sshd -S today -o with-unit
Sun 2023-05-28 01:46:29 CST arch sshd.service[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
Sun 2023-05-28 01:46:29 CST arch sshd.service[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
Sun 2023-05-28 01:47:13 CST arch sshd.service[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
Sun 2023-05-28 01:47:15 CST arch sshd.service[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
Sun 2023-05-28 01:47:19 CST arch sshd.service[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
Sun 2023-05-28 14:01:03 CST arch sshd.service[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
Sun 2023-05-28 14:01:03 CST arch sshd.service[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
 ⊙ 15:22:26  ~
```

与short-full类似，但在单元和用户单元名称前加上前缀，而不是传统的syslog标识符。在使用模板化实例时很有用，因为它将在单元名称中展示参数名称。

2.指定输出的字段列表(--output-fields)

逗号分隔的字段列表，这些字段应该包括在输出中。这只对通常会显示所有字段的输出模式有影响（verbose、export、json、 json-pretty、json-sse和 json-seq）；同时，"CURSOR"、"REALTIME_TIMESTAMP"、"__MONOTONIC_TIMESTAMP"和"_BOOT_ID"字段是固定输出的。

字段含义可通过man 7 systemd.journal-fields来查看。

比如输出字段指定为SYSLOG_IDENTIFIER，其它默认字段也会被强制打印出来：

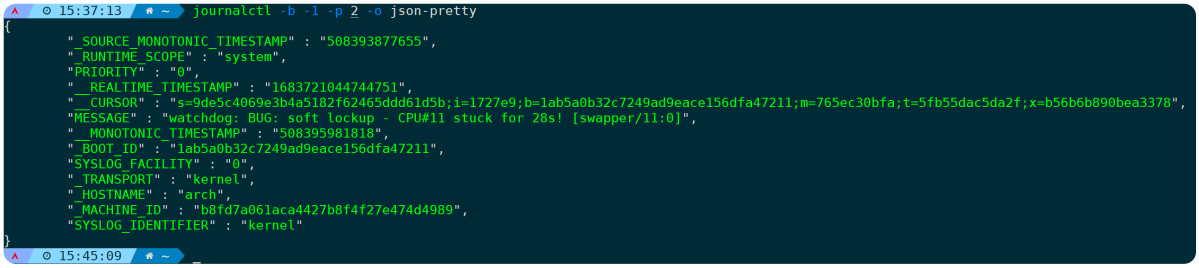
```
journalctl -b -1 -p 0 -o json-pretty --output-fields=SYSLOG_IDENTIFIER
```

```
 ⊙ 15:36:36  ~ journalctl -b -1 -p 0 -o json-pretty --output-fields=SYSLOG_IDENTIFIER
{
        "__REALTIME_TIMESTAMP" : "1683721044744751",
        "__MONOTONIC_TIMESTAMP" : "508395981818",
        "SYSLOG_IDENTIFIER" : "kernel",
        "_BOOT_ID" : "1ab5a0b32c7249ad9eace156dfa47211",
        "__CURSOR" : "s=9de5c4069e3b4a5182f62465ddd61d5b;i=1727e9;b=1ab5a0b32c7249ad9eace156dfa47211;m=765ec30bfa;t=5fb55dac5da2f;x=b56b6b890bea3378"
}
 ⊙ 15:37:13  ~
```

同时注意，如果本身日志里面没有这个字段，不能凭空生成出来，指定的--output-fields也是要在日志记录的范围内的，比如下面这条json-pretty的输出：

```
$ journalctl -b -1 -p 2 -o json-pretty
{
        "_SOURCE_MONOTONIC_TIMESTAMP" : "508393877655",
        "_RUNTIME_SCOPE" : "system",
        "PRIORITY" : "0",
        "__REALTIME_TIMESTAMP" : "1683721044744751",
        "__CURSOR" :
"s=9de5c4069e3b4a5182f62465ddd61d5b;i=1727e9;b=1ab5a0b32c7249ad9eace156dfa47211;m=765e
c30bfa;t=5fb55dac5da2f;x=b56b6b890bea3378",
        "MESSAGE" : "watchdog: BUG: soft lockup - CPU#11 stuck for 28s!
[swapper/11:0]",
        "__MONOTONIC_TIMESTAMP" : "508395981818",
        "_BOOT_ID" : "1ab5a0b32c7249ad9eace156dfa47211",
        "SYSLOG_FACILITY" : "0",
        "_TRANSPORT" : "kernel",
        "_HOSTNAME" : "arch",
```

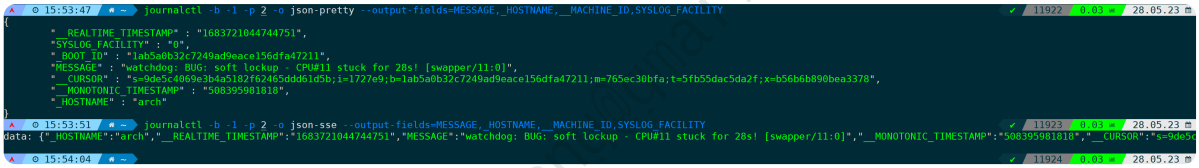
```
    "_MACHINE_ID" : "b8fd7a061aca4427b8f4f27e474d4989",
    "SYSLOG_IDENTIFIER" : "kernel"
}
```



json-pretty默认行为是打印所有字段，如果字段内容为空就不打印，很显然这条日志只有上面这些字段有内容，那么**--output-fields**的取值也只能从上面这些字段中取，不能凭空产生。

比如基于上面这条日志，除了固定的几个字段我不能控制输出，我只想输出MESSAGE、_HOSTNAME、__MACHINE_ID、SYSLOG_FACILITY字段的内容，并且以json-pretty和json-sse格式输出：

```
journalctl -b -1 -p 2 -o json-pretty --output-
fields=MESSAGE,_HOSTNAME,__MACHINE_ID,SYSLOG_FACILITY
journalctl -b -1 -p 2 -o json-sse --output-
fields=MESSAGE,_HOSTNAME,__MACHINE_ID,SYSLOG_FACILITY
```



3.通过指定字段列表筛选日志(<FILED>=)

如果你已经确定了想要的字段的对应日志条目，那么可以通过指定字段的值来过滤匹配。

比如筛选sshd服务**SYSLOG_PID**为7052并且**_SYSTEMD_INVOCATION_ID**为指定字符串的日志，并通过json-pretty格式输出：

```
journalctl SYSLOG_PID=7052 _SYSTEMD_INVOCATION_ID=21174b37b37c4cd5b4d20c828cbffaa3 -u
sshd -o json-pretty
```

```
⊙ 16:04:54  ~ journalctl SYSLOG_PID=7052 _SYSTEMD_INVOCATION_ID=21174b37b37c4cd5b4d20c828cbffaa3 -u sshd -o json-pretty
{
        "_SYSTEMD_INVOCATION_ID" : "21174b37b37c4cd5b4d20c828cbffaa3",
        "_SYSTEMD_UNIT" : "sshd.service",
        "PRIORITY" : "6",
        "_COMM" : "sshd",
        "SYSLOG_IDENTIFIER" : "sshd",
        "_MACHINE_ID" : "b8fd7a061aca4427b8f4f27e474d4989",
        "_SOURCE_REALTIME_TIMESTAMP" : "1685209589419024",
        "_SYSTEMD_CGROUP" : "/system.slice/sshd.service",
        "_PID" : "7052",
        "_RUNTIME_SCOPE" : "system",
        "_HOSTNAME" : "arch",
        "__CURSOR" : "s=9de5c4069e3b4a5182f62465ddd61d5b;i=193f61;b=91b890e06b654b49a54257184891e744;m=5f09574745;t=5fcb06ef30ce4;x=401ab83427cda50",
        "SYSLOG_FACILITY" : "4",
        "SYSLOG_TIMESTAMP" : "May 28 01:46:29 ",
        "_CMDLINE" : "\"sshd: root [priv]\"",
        "_BOOT_ID" : "91b890e06b654b49a54257184891e744",
        "_UID" : "0",
        "MESSAGE" : "Accepted password for root from 192.168.1.3 port 56322 ssh2",
        "_SYSTEMD_SLICE" : "system.slice",
        "_GID" : "0",
        "_TRANSPORT" : "syslog",
        "_EXE" : "/usr/bin/sshd",
        "SYSLOG_PID" : "7052",
        "__REALTIME_TIMESTAMP" : "1685209589419236",
        "_CAP_EFFECTIVE" : "1ffffffffff",
        "__MONOTONIC_TIMESTAMP" : "408178607941"
}
{
        "_BOOT_ID" : "91b890e06b654b49a54257184891e744",
        "SYSLOG_TIMESTAMP" : "May 28 01:46:29 ",
        "SYSLOG_IDENTIFIER" : "sshd",
        "_RUNTIME_SCOPE" : "system",
        "_CMDLINE" : "\"sshd: root [priv]\"",
        "_SOURCE_REALTIME_TIMESTAMP" : "1685209589421466",
        "_PID" : "7052",
        "_COMM" : "sshd",
        "_UID" : "0",
        "__MONOTONIC_TIMESTAMP" : "408178610222",
        "_SYSTEMD_UNIT" : "sshd.service",
        "PRIORITY" : "6",
        "SYSLOG_FACILITY" : "10",
        "_TRANSPORT" : "syslog",
        "__CURSOR" : "s=9de5c4069e3b4a5182f62465ddd61d5b;i=193f62;b=91b890e06b654b49a54257184891e744;m=5f0957502e;t=5fcb06ef315cc;x=1f2b32c96c965b6e",
        "__REALTIME_TIMESTAMP" : "1685209589421516",
        "_SYSTEMD_CGROUP" : "/system.slice/sshd.service",
        "_MACHINE_ID" : "b8fd7a061aca4427b8f4f27e474d4989",
        "_SYSTEMD_INVOCATION_ID" : "21174b37b37c4cd5b4d20c828cbffaa3",
        "_HOSTNAME" : "arch",
        "_GID" : "0",
        "MESSAGE" : "pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)",
        "SYSLOG_PID" : "7052",
        "_CAP_EFFECTIVE" : "1ffffffffff",
        "_EXE" : "/usr/bin/sshd",
        "_SYSTEMD_SLICE" : "system.slice"
}
⊙ 16:04:57  ~
```

又或者指定_BOOT_ID为某个值，筛选优先级为3的内核日志信息，short-iso格式输出：

```
journalctl -k _BOOT_ID=91b890e06b654b49a54257184891e744 -p 3 -o short-iso
```

```
⊙ 16:08:49  ~ journalctl -k _BOOT_ID=91b890e06b654b49a54257184891e744 -p 3 -o short-iso
2023-05-23T08:23:54+0800 arch kernel: piix4_smbus 0000:00:07.3: SMBus Host Controller not enabled!
⊙ 16:08:54  ~
```

journalctl --list-boots会列出系统已记录的所有_BOOT_ID：

```
 16:13:13  ~  journalctl --list-boots
IDX BOOT ID                          FIRST ENTRY                 LAST ENTRY
-29 bd872ff980c14071be90e2d6e43e46e2 Tue 2022-05-24 09:05:48 CST Sat 2022-05-28 13:49:32 CST
-28 6990345cb14549d88daa83bcfed9169b Sun 2022-05-29 11:44:16 CST Tue 2022-06-07 02:04:57 CST
-27 dc0b8d563e324ecba747f4c196aaa8d4 Tue 2022-06-07 02:46:32 CST Fri 2022-06-24 05:17:03 CST
-26 7510cd8e2e7b47619926f46093be4162 Fri 2022-06-24 17:26:08 CST Fri 2022-06-24 18:28:47 CST
-25 acf18b3add8a40909287e9d16e2c7a2c Fri 2022-06-24 18:28:55 CST Fri 2022-06-24 19:08:13 CST
-24 b2178c2e30904178908b94da546fb412 Fri 2022-06-24 19:09:57 CST Thu 2022-09-22 02:42:56 CST
-23 f6fced3a28424c52ac609e1baf6bb805 Thu 2022-09-22 02:47:00 CST Sat 2022-10-01 21:27:47 CST
-22 53a9a1c3f27246fd87e37b0fb1e4850a Sat 2022-10-01 21:45:59 CST Mon 2022-10-03 13:07:07 CST
-21 f4ece52b515e44a9ad10308008d46e7e Mon 2022-10-03 13:22:37 CST Thu 2022-10-06 21:58:46 CST
-20 6bcbe7f703074b03a4c806e3ffd60e80 Thu 2022-10-06 22:00:43 CST Sat 2022-10-08 05:54:49 CST
-19 874906559b7f4583b6150a5616f3bfe6 Sat 2022-10-08 05:57:35 CST Fri 2022-10-14 22:07:18 CST
-18 008dfccde22947449bd0384505376742 Fri 2022-10-14 22:09:16 CST Wed 2022-11-02 21:33:31 CST
-17 fdbb0dfab2cd453aa7b473eb7e7a153d Wed 2022-11-02 22:05:48 CST Fri 2022-11-11 13:40:31 CST
-16 2651b14459454495895d5c6801721648 Fri 2022-11-11 13:49:36 CST Sun 2022-11-20 05:38:07 CST
-15 a0b5baf657a148aaaf2b64feefbee21c Sat 2022-11-19 21:38:20 CST Tue 2023-01-17 20:37:20 CST
-14 821e72ec38154ff3b7c34d0db99f37b1 Tue 2023-01-17 20:37:38 CST Wed 2023-01-18 18:29:45 CST
-13 709bc21614a64157b6d1eebe36c0cf3f Wed 2023-01-18 18:32:19 CST Wed 2023-01-25 11:44:22 CST
-12 4a197eb813a44cac9a75464640e0b1b0 Wed 2023-01-25 20:55:47 CST Wed 2023-01-25 20:59:22 CST
-11 132652d2d3bd4a0ab732205025562c30 Wed 2023-01-25 21:09:42 CST Wed 2023-01-25 21:18:22 CST
-10 81e669da42904059b2272b4e75c3e84c Wed 2023-01-25 21:38:55 CST Thu 2023-01-26 18:01:22 CST
 -9 b8f41f5a560c4a5bb851368faae9e898 Thu 2023-01-26 18:25:22 CST Fri 2023-03-17 15:51:26 CST
 -8 f99a93283b104b16967783e160f2e014 Fri 2023-03-17 15:51:45 CST Sat 2023-04-15 10:11:57 CST
 -7 056213455b5e4f6f9dea07c233c808ec Sat 2023-04-15 10:12:18 CST Sun 2023-04-16 12:09:19 CST
 -6 dc9b2e2a8a9e4166a699a63c748e9322 Sun 2023-04-16 12:47:46 CST Tue 2023-04-18 20:34:45 CST
 -5 dd7a8c34b9a84d598a90658faf9767bd Tue 2023-04-18 20:36:13 CST Thu 2023-04-20 00:22:33 CST
 -4 7b34bd6a8ccc4b7886606477d3fa880b Thu 2023-04-20 15:57:58 CST Sat 2023-04-29 12:01:33 CST
 -3 282b653f8aff45989654220e708a9001 Sat 2023-04-29 12:48:31 CST Sat 2023-04-29 12:52:33 CST
 -2 06dd1eebfbbe4970bb59559cd3fcca51 Sat 2023-04-29 13:06:27 CST Thu 2023-05-04 23:03:22 CST
 -1 1ab5a0b32c7249ad9eace156dfa47211 Thu 2023-05-04 23:04:20 CST Tue 2023-05-23 07:39:22 CST
  0 91b890e06b654b49a54257184891e744 Tue 2023-05-23 08:23:46 CST Sun 2023-05-28 16:12:33 CST
 16:13:19  ~
```

4.显示最近的日志的指定行数(-n, --lines)

指定此参数后，默认显示最近10行日志：

```
journalctl -u sshd -n
```

显示最近20行日志：

```
journalctl -u sshd -n 20
```

```
 18:38:17  ~  journalctl -u sshd -n
May 28 01:46:29 arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 01:47:13 arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
May 28 01:47:15 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
May 28 01:47:19 arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
May 28 14:01:03 arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
May 28 14:01:03 arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 16:39:25 arch sshd[20039]: error: kex_exchange_identification: banner line contains invalid characters
May 28 16:39:25 arch sshd[20039]: banner exchange: Connection from 89.248.165.103 port 34220: invalid format
May 28 18:36:01 arch sshd[20082]: Accepted password for root from 192.168.1.3 port 21902 ssh2
May 28 18:36:01 arch sshd[20082]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
 18:38:34  ~  journalctl -u sshd -n 20
May 26 15:16:04 arch sshd[4272]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 00:18:42 arch sshd[4608]: Accepted password for root from 192.168.1.3 port 8798 ssh2
May 27 00:18:42 arch sshd[4608]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 11:43:58 arch sshd[5041]: Accepted password for root from 192.168.1.3 port 7228 ssh2
May 27 11:43:58 arch sshd[5041]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 12:38:52 arch sshd[5153]: Accepted password for root from 192.168.1.3 port 13610 ssh2
May 27 12:38:52 arch sshd[5153]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 22:47:54 arch sshd[5465]: Accepted password for root from 192.168.1.3 port 58238 ssh2
May 27 22:47:54 arch sshd[5465]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 01:46:29 arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
May 28 01:46:29 arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 01:47:13 arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
May 28 01:47:15 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
May 28 01:47:19 arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
May 28 14:01:03 arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
May 28 14:01:03 arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 16:39:25 arch sshd[20039]: error: kex_exchange_identification: banner line contains invalid characters
May 28 16:39:25 arch sshd[20039]: banner exchange: Connection from 89.248.165.103 port 34220: invalid format
May 28 18:36:01 arch sshd[20082]: Accepted password for root from 192.168.1.3 port 21902 ssh2
May 28 18:36:01 arch sshd[20082]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
 18:38:35  ~
```

当配合--grep一起使用时，默认会有--reverse的效果，根据时间降序：

```
⊙ 18:38:35  ~  journalctl -u sshd -n 20 --grep 'pam'
May 28 18:36:01 arch sshd[20082]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 14:01:03 arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 01:47:13 arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
May 28 01:46:29 arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 22:47:54 arch sshd[5465]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 12:38:52 arch sshd[5153]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 11:43:58 arch sshd[5041]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 00:18:42 arch sshd[4608]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 26 15:16:04 arch sshd[4272]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 26 15:14:43 arch sshd[4169]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 25 18:32:26 arch sshd[3512]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 25 17:02:57 arch sshd[3338]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 24 17:13:00 arch sshd[2741]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 23 22:49:39 arch sshd[2252]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 23 17:52:49 arch sshd[1880]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 23 12:45:47 arch sshd[1539]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
-- Boot 1ab5a0b32c7249ad9eace156dfa47211 --
May 22 21:57:42 arch sshd[14972]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 22 15:56:43 arch sshd[14647]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 22 15:47:05 arch sshd[14550]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 20 02:36:55 arch sshd[13291]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
⊙ 18:39:47  ~  _
```

5.反向显示(-r, --reverse)

不指定时默认是根据日志先后顺序排序，指定后会将顺序反过来显示。

比如筛选prometheus服务今天最近20行的日志，从后往前排序：

```
journalctl -u prometheus -S today -r -n 20
```

再比如显示身份授权相关且等级为4(warning)到7(debug)级最近30行日志反向排序：

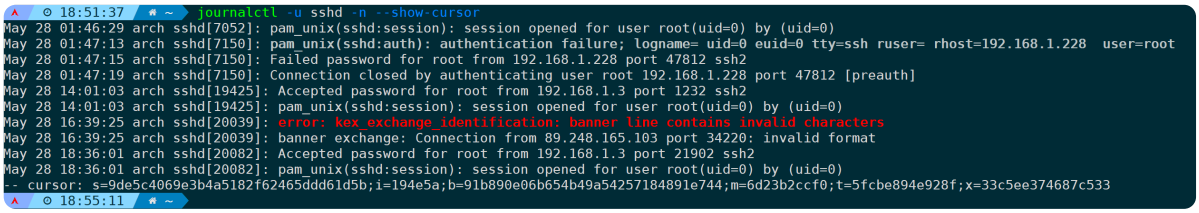
```
journalctl -p 4..7 --facility=auth -r -n 30
```

```
⊙ 18:46:10  ~  journalctl -p 4..7 --facility=auth -r -n 30
May 28 18:36:01 arch systemd-logind[932]: New session 21 of user root.
May 28 18:36:01 arch sshd[20082]: Accepted password for root from 192.168.1.3 port 21902 ssh2
May 28 16:59:55 arch systemd-logind[932]: Removed session 19.
May 28 16:59:55 arch systemd-logind[932]: Session 19 logged out. Waiting for processes to exit.
May 28 16:39:25 arch sshd[20039]: banner exchange: Connection from 89.248.165.103 port 34220: invalid format
May 28 14:01:03 arch systemd-logind[932]: New session 19 of user root.
May 28 14:01:03 arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
May 28 06:48:15 arch systemd-logind[932]: Removed session 11.
May 28 06:48:14 arch systemd-logind[932]: Session 11 logged out. Waiting for processes to exit.
May 28 06:47:47 arch systemd-logind[932]: Removed session 13.
May 28 06:47:47 arch systemd-logind[932]: Session 13 logged out. Waiting for processes to exit.
May 28 03:23:07 arch systemd-logind[932]: Removed session 18.
May 28 03:23:07 arch systemd-logind[932]: Removed session 17.
May 28 03:23:07 arch systemd-logind[932]: Session 18 logged out. Waiting for processes to exit.
May 28 03:23:07 arch systemd-logind[932]: Session 17 logged out. Waiting for processes to exit.
May 28 01:47:19 arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
May 28 01:47:15 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
May 28 01:46:29 arch systemd-logind[932]: New session 18 of user root.
May 28 01:46:29 arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
May 27 22:47:54 arch systemd-logind[932]: New session 17 of user root.
May 27 22:47:54 arch sshd[5465]: Accepted password for root from 192.168.1.3 port 58238 ssh2
May 27 16:06:45 arch systemd-logind[932]: Removed session 16.
May 27 16:06:45 arch systemd-logind[932]: Session 16 logged out. Waiting for processes to exit.
May 27 12:38:52 arch systemd-logind[932]: New session 16 of user root.
May 27 12:38:52 arch sshd[5153]: Accepted password for root from 192.168.1.3 port 13610 ssh2
May 27 11:44:13 arch systemd-logind[932]: Removed session 15.
May 27 11:44:13 arch systemd-logind[932]: Session 15 logged out. Waiting for processes to exit.
May 27 11:43:58 arch systemd-logind[932]: New session 15 of user root.
May 27 11:43:58 arch sshd[5041]: Accepted password for root from 192.168.1.3 port 7228 ssh2
May 27 03:58:36 arch systemd-logind[932]: Removed session 14.
⊙ 18:46:30  ~
```

6.显示游标位置(--show-cursor)

显示sshd服务最近10行日志，并输出最后一行的游标位置：

```
journalctl -u sshd -n --show-cursor
```

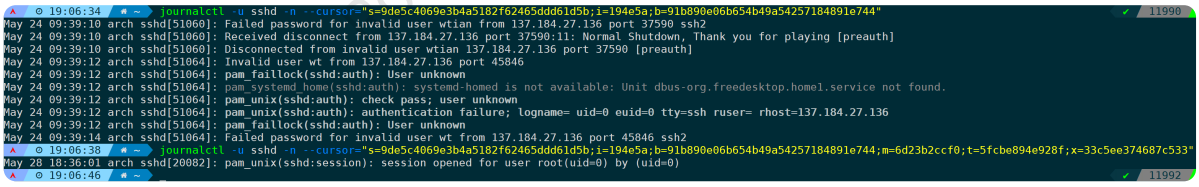


- `s`：代表序列号（sequence），它是一个标识日志消息序列的字符串。序列号用于标记日志消息的顺序，确保它们按照正确的顺序显示。
- `i`：代表日志文件索引号（file index），它指示了包含当前日志消息的日志文件的索引位置。每个日志文件都有一个唯一的索引号。
- `b`：代表引导 ID（boot ID），它标识了启动会话（boot session）。每次系统启动都会生成一个唯一的引导 ID，用于区分不同的启动会话。
- `m`：代表日志文件位置（monotonic），它表示日志消息在日志文件中的位置。它是一个递增的数值，用于确保日志消息在日志文件中的唯一性和顺序。

7.根据游标位置过滤日志(-c, --cursor)

拿到游标位置后，可以通过指定游标位置来定位日志：

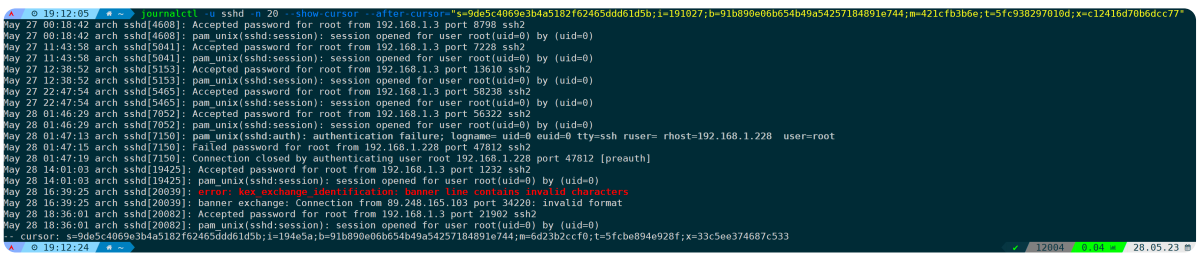
```
journalctl -u sshd -n --cursor="<CURSOR>"
```



8.显示某个光标之后的日志(--after-cursor)

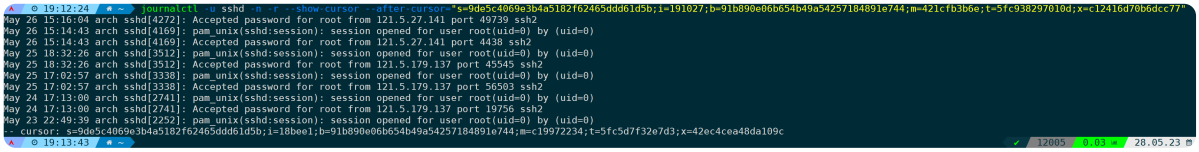
显示指定光标后的日志，并显示20行，并且在最后展示光标位置：

```
journalctl -u sshd -n 20 --show-cursor --after-cursor="<CURSOR>"
```



显示某个光标位置之后的10行日志，并反向输出：

```
journalctl -u sshd -n -r --show-cursor --after-cursor="<CURSOR>"
```



9.以UTC时间格式输出(--utc)

输出prometheus服务今天最近10行的日志，并以UTC时间输出：

```
journalctl --utc -u prometheus -S today -n
```

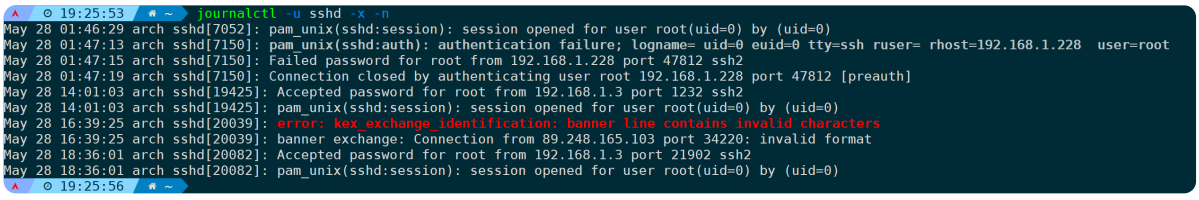
可以看到对比正常时间刚好差8个小时时区。

10.显示相关联的消息目录信息(-x, --catalog)

在systemd-journald的日志系统中，每条日志消息可以与一个或多个消息目录（message catalog）相关联。

使用 -x 或 --catalog 选项，journalctl 命令会尝试显示与每条日志消息相关联的消息目录信息：

```
journalctl -u sshd -x -n
```



如果没有关联的日志，则还是正常显示。

11.不显示主机名字段(--no-hostname)

顾名思义，不显示主机名，但这个参数只作用于日志格式为short开头的类别。

显示最近10行sshd服务日志，并且不显示主机名：

```
journalctl -u sshd --no-hostname -n
```

```
 19:31:55  ~  journalctl -u sshd --no-hostname -n
May 28 01:46:29 sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 01:47:13 sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
May 28 01:47:15 sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
May 28 01:47:19 sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
May 28 14:01:03 sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
May 28 14:01:03 sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 16:39:25 sshd[20039]: error: kex_exchange_identification: banner line contains invalid characters
May 28 16:39:25 sshd[20039]: banner exchange: Connection from 89.248.165.103 port 34220: invalid format
May 28 18:36:01 sshd[20082]: Accepted password for root from 192.168.1.3 port 21902 ssh2
May 28 18:36:01 sshd[20082]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
 19:31:58  ~  journalctl -u sshd -n
May 28 01:46:29 arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 01:47:13 arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
May 28 01:47:15 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
May 28 01:47:19 arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
May 28 14:01:03 arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
May 28 14:01:03 arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 16:39:25 arch sshd[20039]: error: kex_exchange_identification: banner line contains invalid characters
May 28 16:39:25 arch sshd[20039]: banner exchange: Connection from 89.248.165.103 port 34220: invalid format
May 28 18:36:01 arch sshd[20082]: Accepted password for root from 192.168.1.3 port 21902 ssh2
May 28 18:36:01 arch sshd[20082]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
 19:32:08  ~
```

默认输出格式为short，所以这个参数值是生效的。

假设使用json格式来输出：

```
journalctl -u sshd --no-hostname -n -o json-pretty
```

```
 19:34:53  ~  journalctl -u sshd --no-hostname -n 1 -o json-pretty
{
        "_CAP_EFFECTIVE" : "1ffffffffff",
        "_SOURCE_REALTIME_TIMESTAMP" : "1685270161166856",
        "_HOSTNAME" : "arch",
        "_CMDLINE" : "\"sshd: root [priv]\"",
        "_RUNTIME_SCOPE" : "system",
        "_UID" : "0",
        "__REALTIME_TIMESTAMP" : "1685270161166991",
        "PRIORITY" : "6",
        "_SYSTEMD_UNIT" : "sshd.service",
        "_COMM" : "sshd",
        "_PID" : "20082",
        "__MONOTONIC_TIMESTAMP" : "468750355696",
        "MESSAGE" : "pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)",
        "_SYSTEMD_SLICE" : "system.slice",
        "_EXE" : "/usr/bin/sshd",
        "SYSLOG_TIMESTAMP" : "May 28 18:36:01 ",
        "_GID" : "0",
        "SYSLOG_FACILITY" : "10",
        "_MACHINE_ID" : "b8fd7a061aca4427b8f4f27e474d4989",
        "_SYSTEMD_CGROUP" : "/system.slice/sshd.service",
        "__CURSOR" : "s=9de5c4069e3b4a5182f62465ddd61d5b;i=194e5a;b=91b890e06b654b49a54257184891e744;m=6d23b2ccf0;t=5fcbe894e928f;x=33c5ee374687c533",
        "_TRANSPORT" : "syslog",
        "_SYSTEMD_INVOCATION_ID" : "21174b37b37c4cd5b4d20c828cbffaa3",
        "SYSLOG_IDENTIFIER" : "sshd",
        "_BOOT_ID" : "91b890e06b654b49a54257184891e744",
        "SYSLOG_PID" : "20082"
}
 19:34:59  ~
```

依然会输出主机名字段。

12.截断输出(--no-full)

此参数会禁止完整显示长日志消息的内容。当日志消息非常长时，会被截断为摘要形式，以保持输出的简洁性：

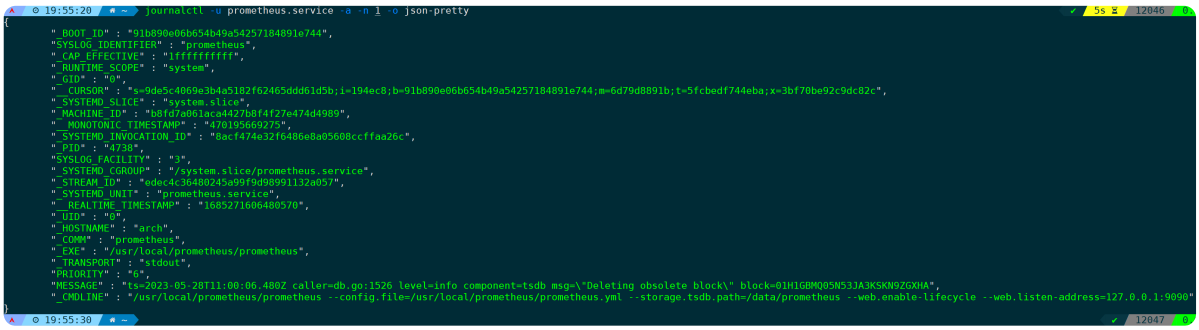
```
journalctl --no-full
```

```
 19:47:10  ~  journalctl -u prometheus -n 10 -r                                                                                ✔  12033  0.03  28.05.23
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.480Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1GBMQ05N53JA3KSKN9ZGXHA
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.476Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1H07WN79S4YMCJKPMYA4NZ6
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.472Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1FQ1H4BBZJQ4CG0BEYBH0GT
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.463Z caller=compact.go:460 level=info component=tsdb msg="compact blocks" count=3 mint=1685188800662 maxt=1685253600000 ulid=01H1H07WW9A9XV36GV9TQR2M8K source>
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.151Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1GBMPQAHA4Q2FAFRB7F36CE
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.146Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1G4RZFB3GQCRN9PQQ7EXJV6
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.142Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1GJGDZ9KW5FXQ7VT4RW39NA
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.135Z caller=compact.go:460 level=info component=tsdb msg="compact blocks" count=3 mint=1685232000662 maxt=1685253600000 ulid=01H1H07WN79S4YMCJKPMYA4NZ6 source>
May 28 19:00:05 arch prometheus[4738]: ts=2023-05-28T11:00:05.926Z caller=head.go:1192 level=info component=tsdb msg="Head GC completed" caller=truncateMemory duration=7.310154ms
May 28 19:00:05 arch prometheus[4738]: ts=2023-05-28T11:00:05.915Z caller=compact.go:519 level=info component=tsdb msg="write block" mint=1685260800662 maxt=1685268000000 ulid=01H1H07WFBDE21QCV8XYA2Z07R duration=175.632ms
 19:47:17  ~  journalctl -u prometheus -n 10 -r --no-full                                                                     ✔  12034  0.03  28.05.23
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.480Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1GBMQ05N53JA3KSKN9ZGXHA
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.476Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1H07WN79S4YMCJKPMYA4NZ6
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.472Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1FQ1H4BBZJQ4CG0BEYBH0GT
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.463Z caller=compact.go:460 level=info component=tsdb msg="compact blocks" count=3 mint=1685188800662 maxt=1685253600000 ulid=01H1H07WW9A9XV3…ration=310.30646ms
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.151Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1GBMPQAHA4Q2FAFRB7F36CE
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.146Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1G4RZFB3GQCRN9PQQ7EXJV6
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.142Z caller=db.go:1526 level=info component=tsdb msg="Deleting obsolete block" block=01H1GJGDZ9KW5FXQ7VT4RW39NA
May 28 19:00:06 arch prometheus[4738]: ts=2023-05-28T11:00:06.135Z caller=compact.go:460 level=info component=tsdb msg="compact blocks" count=3 mint=1685232000662 maxt=1685253600000 ulid=01H1H07WN79S4YM…ation=208.414104ms
May 28 19:00:05 arch prometheus[4738]: ts=2023-05-28T11:00:05.926Z caller=head.go:1192 level=info component=tsdb msg="Head GC completed" caller=truncateMemory duration=7.310154ms
May 28 19:00:05 arch prometheus[4738]: ts=2023-05-28T11:00:05.915Z caller=compact.go:519 level=info component=tsdb msg="write block" mint=1685260800662 maxt=1685268000000 ulid=01H1H07WFBDE21QCV8XYA2Z07R duration=175.632ms
 19:47:27  ~                                                                                                                    ✔  12035  0.06  28.05.23
```

13.完整输出所有字段(-a, --all)

完整地显示所有字段，即使它们包括不可打印的字符或非常长：

```
journalctl -u prometheus.service -a -n 1 -o json-pretty
```

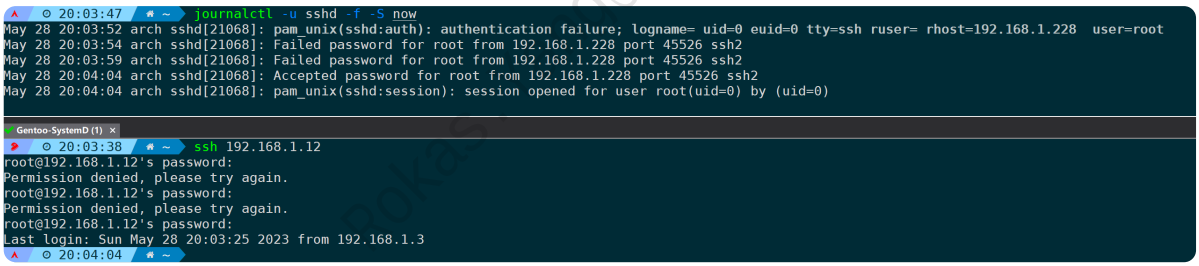


14.跟踪日志条目(-f, --follow)

作用和tail命令的-f参数类似，一直跟踪日志的输出。

比如跟踪sshd服务日志输出，时间点从现在开始：

```
journalctl -u sshd -f -S now
```



测试两次失败登录一次成功登录都实时显示在了日志上。

15.显示所有已存储的输出行(--no-tail)

使用-f参数时，默认只显示后10行日志，然后继续追踪日志，而--no-tail会将所有行显示出来，再继续追踪日志：

```
journalctl -u sshd -f --no-tail
```

```
⊙ 20:07:22  ~  journalctl -u sshd -f
May 28 20:03:11 arch sshd[20971]: Failed password for root from 192.168.1.228 port 57644 ssh2
May 28 20:03:52 arch sshd[21068]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
May 28 20:03:54 arch sshd[21068]: Failed password for root from 192.168.1.228 port 45526 ssh2
May 28 20:03:59 arch sshd[21068]: Failed password for root from 192.168.1.228 port 45526 ssh2
May 28 20:04:04 arch sshd[21068]: Accepted password for root from 192.168.1.228 port 45526 ssh2
May 28 20:04:04 arch sshd[21068]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 20:04:13 arch sshd[20971]: Accepted password for root from 192.168.1.228 port 57644 ssh2
May 28 20:04:13 arch sshd[20971]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 20:04:13 arch sshd[20971]: pam_systemd(sshd:session): Failed to create session: Transport endpoint is not connected
May 28 20:04:13 arch sshd[20971]: pam_unix(sshd:session): session closed for user root
^C
⊙ 20:07:28  ~  journalctl -u sshd -f --no-tail
May 23 08:25:16 arch systemd[1]: Started OpenSSH Daemon.
May 23 08:25:20 arch sshd[990]: Server listening on 0.0.0.0 port 22.
May 23 08:25:20 arch sshd[990]: Server listening on :: port 22.
May 23 12:45:47 arch sshd[1539]: Accepted password for root from 192.168.1.3 port 9580 ssh2
May 23 12:45:47 arch sshd[1539]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 23 17:52:49 arch sshd[1880]: Accepted password for root from 121.5.179.137 port 49117 ssh2
May 23 17:52:49 arch sshd[1880]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 23 22:49:39 arch sshd[2252]: Accepted password for root from 27.46.67.32 port 2135 ssh2
May 23 22:49:39 arch sshd[2252]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 24 17:13:00 arch sshd[2741]: Accepted password for root from 121.5.179.137 port 19756 ssh2
May 24 17:13:00 arch sshd[2741]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 25 17:02:57 arch sshd[3338]: Accepted password for root from 121.5.179.137 port 56503 ssh2
May 25 17:02:57 arch sshd[3338]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 25 18:32:26 arch sshd[3512]: Accepted password for root from 121.5.179.137 port 45545 ssh2
May 25 18:32:26 arch sshd[3512]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 26 15:14:43 arch sshd[4169]: Accepted password for root from 121.5.27.141 port 4438 ssh2
May 26 15:14:43 arch sshd[4169]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 26 15:16:04 arch sshd[4272]: Accepted password for root from 121.5.27.141 port 49739 ssh2
May 26 15:16:04 arch sshd[4272]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 00:18:42 arch sshd[4608]: Accepted password for root from 192.168.1.3 port 8798 ssh2
May 27 00:18:42 arch sshd[4608]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 11:43:58 arch sshd[5041]: Accepted password for root from 192.168.1.3 port 7228 ssh2
May 27 11:43:58 arch sshd[5041]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 12:38:52 arch sshd[5153]: Accepted password for root from 192.168.1.3 port 13610 ssh2
May 27 12:38:52 arch sshd[5153]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 27 22:47:54 arch sshd[5465]: Accepted password for root from 192.168.1.3 port 58238 ssh2
May 27 22:47:54 arch sshd[5465]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 01:46:29 arch sshd[7052]: Accepted password for root from 192.168.1.3 port 56322 ssh2
May 28 01:46:29 arch sshd[7052]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 01:47:13 arch sshd[7150]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
May 28 01:47:15 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
May 28 01:47:19 arch sshd[7150]: Connection closed by authenticating user root 192.168.1.228 port 47812 [preauth]
May 28 14:01:03 arch sshd[19425]: Accepted password for root from 192.168.1.3 port 1232 ssh2
May 28 14:01:03 arch sshd[19425]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 16:39:25 arch sshd[20039]: error: kex_exchange_identification: banner line contains invalid characters
May 28 16:39:25 arch sshd[20039]: banner exchange: Connection from 89.248.165.103 port 34220: invalid format
May 28 18:36:01 arch sshd[20082]: Accepted password for root from 192.168.1.3 port 21902 ssh2
May 28 18:36:01 arch sshd[20082]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 28 20:00:15 arch sshd[20639]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
```

16.静默模式下输出(-q,--quiet)

只显示关键的日志信息，过滤掉一些额外的提示和冗余内容。

静默模式输出上一次引导时的最新10行日志：

```
journalctl -b -1 -n -q
```

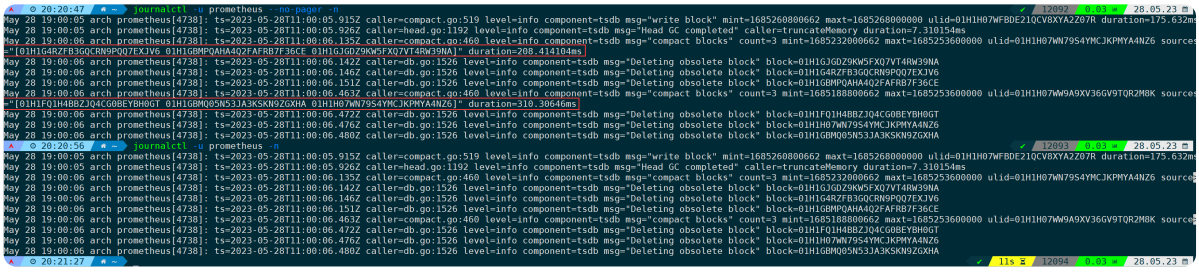

四、分页控制选项详解

1.分页输出(--no-pager)

禁止把输出内容输出到一个页面中，当日志内容冗长容易被截断的时候非常有用。

分页输出最近10行prometheus的日志：

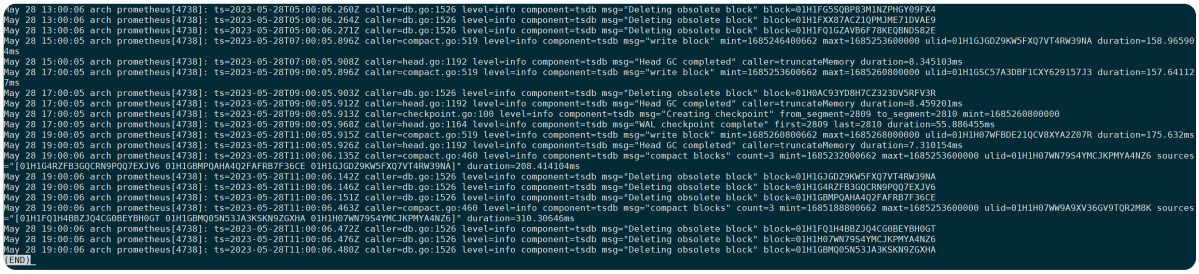
```
journalctl -u prometheus --no-pager -n
```



可以看到不加--no-pager和加此参数是有明显区别的，不加的情况下一行输出不下，直接截断不输出后面的内容，加的情况下，后面的内容会另起一行输出。

同时也可以管道给less命令来条条查看输出：

```
journalctl -u prometheus -n 20 --no-pager|less
```

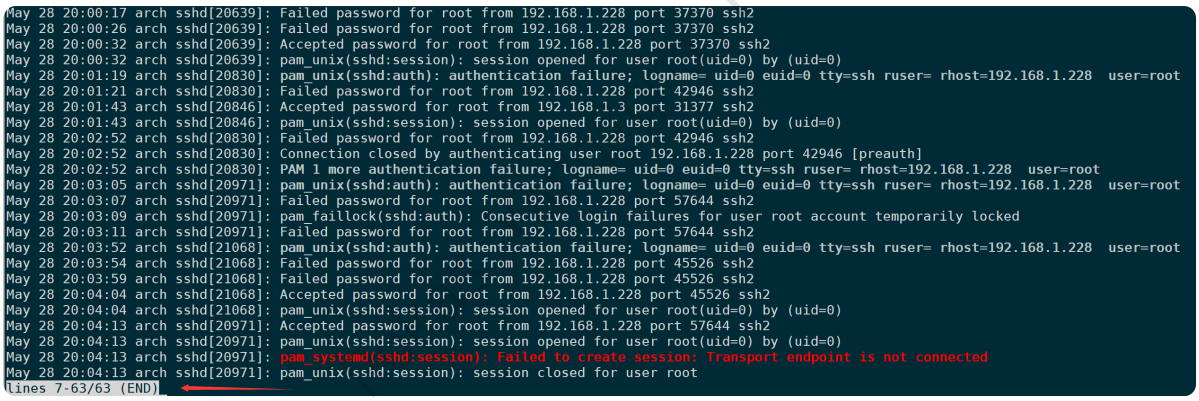


2.定位到日志末尾行(-e, --pager-end)

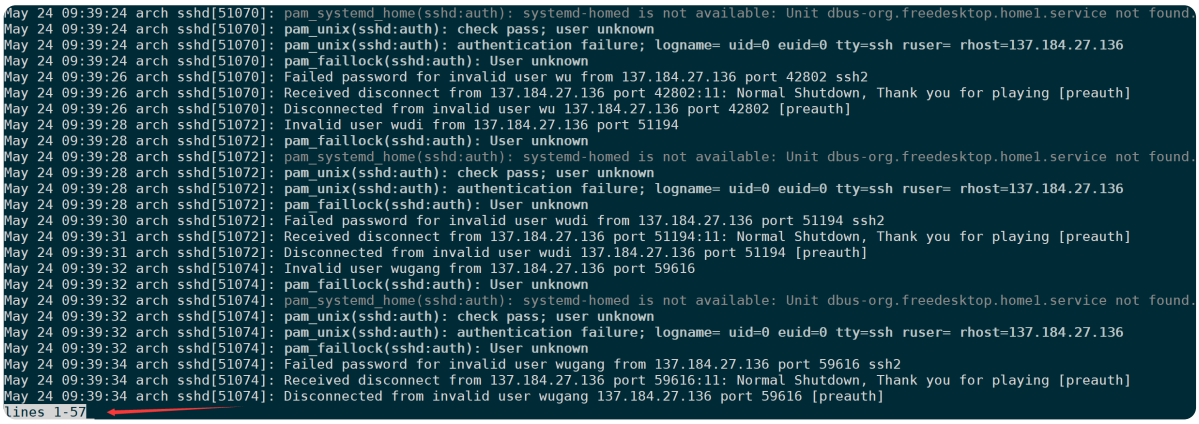
这个选项在查看最新的日志内容时特别有用，这样可以不用手动滚动到日志末尾。

定位sshd服务末尾行日志：

```
journalctl -u sshd -e
```



不加此参数的效果，从最开始一页页翻阅：



五、命令选项

下面这些命令选项更像是一些统计汇总，和对日志集的操作查看等，默认不会展示日志记录，只统计输出。

1.输出所有字段名(-N, --fields)

打印当前在journal所有条目中使用的所有字段名：

```
journalctl -N
```

```
 20:45:43  ~ journalctl -N |head -n 20
AUDIT_FIELD_TERMINAL
SYSLOG_IDENTIFIER
PRIORITY
_SYSTEMD_SLICE
_AUDIT_TYPE
_GID
MESSAGE
AUDIT_FIELD_GRANTORS
_SYSTEMD_INVOCATION_ID
_COMM
JOB_ID
CODE_LINE
_SYSTEMD_CGROUP
_HOSTNAME
_SYSTEMD_UNIT
_AUDIT_LOGINUID
TID
AUDIT_FIELD_ACCT
JOB_TYPE
CODE_FUNC
 20:45:46  ~
```

2.统计日志中指定字段的所有取值(-F, --field)

比如打印journal记录的日志中_BOOT_ID、 _HOSTNAME、 PRIORITY字段的所有取值：

```
journalctl -F _BOOT_ID
journalctl -F _HOSTNAME
journalctl -F PRIORITY
```

```
 20:52:19  ~ journalctl -F _BOOT_ID
dc0b8d563e324ecba747f4c196aaa8d4
f6fced3a28424c52ac609e1baf6bb805
6990345cb14549d88daa83bcfed9169b
91b890e06b654b49a54257184891e744
a0b5baf657a148aaaf2b64feefbee21c
6bcbe7f703074b03a4c806e3ffd60e80
bd872ff980c14071be90e2d6e43e46e2
056213455b5e4f6f9dea07c233c808ec
dd7a8c34b9a84d598a90658faf9767bd
dc9b2e2a8a9e4166a699a63c748e9322
7b34bd6a8ccc4b7886606477d3fa880b
fdbb0dfab2cd453aa7b473eb7e7a153d
97f0cb53180345c2ad476fd3376904ef
282b653f8aff45989654220e708a9001
f99a93283b104b16967783e160f2e014
b2178c2e30904178908b94da546fb412
acf18b3add8a40909287e9d16e2c7a2c
7510cd8e2e7b47619926f46093be4162
81e669da42904059b2272b4e75c3e84c
008dfccde22947449bd0384505376742
2651b14459454495895d5c6801721648
4a197eb813a44cac9a75464640e0b1b0
709bc21614a64157b6d1eebe36c0cf3f
821e72ec38154ff3b7c34d0db99f37b1
1ab5a0b32c7249ad9eace156dfa47211
06dd1eebfbbe4970bb59559cd3fcca51
b8f41f5a560c4a5bb851368faae9e898
f4ece52b515e44a9ad10308008d46e7e
132652d2d3bd4a0ab732205025562c30
42646ca3d25c4c509c8c73e3c9c93ff3
874906559b7f4583b6150a5616f3bfe6
53a9a1c3f27246fd87e37b0fb1e4850a
 20:52:33  ~ journalctl -F _HOSTNAME
arch
 20:52:43  ~ journalctl -F PRIORITY
7
6
5
2
3
4
0
 20:52:54  ~
```

3.列出当前统计的所有BOOT ID(--list-boots)

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

```
 ⊙ 20:53:26  ~  journalctl --list-boots
IDX BOOT ID                          FIRST ENTRY                 LAST ENTRY
-29 bd872ff980c14071be90e2d6e43e46e2 Tue 2022-05-24 09:05:48 CST Sat 2022-05-28 13:49:32 CST
-28 6990345cb14549d88daa83bcfed9169b Sun 2022-05-29 11:44:16 CST Tue 2022-06-07 02:04:57 CST
-27 dc0b8d563e324ecba747f4c196aaa8d4 Tue 2022-06-07 02:46:32 CST Fri 2022-06-24 05:17:03 CST
-26 7510cd8e2e7b47619926f46093be4162 Fri 2022-06-24 17:26:08 CST Fri 2022-06-24 18:28:47 CST
-25 acf18b3add8a40909287e9d16e2c7a2c Fri 2022-06-24 18:28:55 CST Fri 2022-06-24 19:08:13 CST
-24 b2178c2e30904178908b94da546fb412 Fri 2022-06-24 19:09:57 CST Thu 2022-09-22 02:42:56 CST
-23 f6fced3a28424c52ac609e1baf6bb805 Thu 2022-09-22 02:47:00 CST Sat 2022-10-01 21:27:47 CST
-22 53a9a1c3f27246fd87e37b0fb1e4850a Sat 2022-10-01 21:45:59 CST Mon 2022-10-03 13:07:07 CST
-21 f4ece52b515e44a9ad10308008d46e7e Mon 2022-10-03 13:22:37 CST Thu 2022-10-06 21:58:46 CST
-20 6bcbe7f703074b03a4c806e3ffd60e80 Thu 2022-10-06 22:00:43 CST Sat 2022-10-08 05:54:49 CST
-19 874906559b7f4583b6150a5616f3bfe6 Sat 2022-10-08 05:57:35 CST Fri 2022-10-14 22:07:18 CST
-18 008dfccde22947449bd0384505376742 Fri 2022-10-14 22:09:16 CST Wed 2022-11-02 21:33:31 CST
-17 fdbb0dfab2cd453aa7b473eb7e7a153d Wed 2022-11-02 22:05:48 CST Fri 2022-11-11 13:40:31 CST
-16 2651b14459454495895d5c6801721648 Fri 2022-11-11 13:49:36 CST Sun 2022-11-20 05:38:07 CST
-15 a0b5baf657a148aaaf2b64feefbee21c Sat 2022-11-19 21:38:20 CST Tue 2023-01-17 20:37:20 CST
-14 821e72ec38154ff3b7c34d0db99f37b1 Tue 2023-01-17 20:37:38 CST Wed 2023-01-18 18:29:45 CST
-13 709bc21614a64157b6d1eebe36c0cf3f Wed 2023-01-18 18:32:19 CST Wed 2023-01-25 11:44:22 CST
-12 4a197eb813a44cac9a75464640e0b1b0 Wed 2023-01-25 20:55:47 CST Wed 2023-01-25 20:59:22 CST
-11 132652d2d3bd4a0ab732205025562c30 Wed 2023-01-25 21:09:42 CST Wed 2023-01-25 21:18:22 CST
-10 81e669da42904059b2272b4e75c3e84c Wed 2023-01-25 21:38:55 CST Thu 2023-01-26 18:01:22 CST
 -9 b8f41f5a560c4a5bb851368faae9e898 Thu 2023-01-26 18:25:22 CST Fri 2023-03-17 15:51:26 CST
 -8 f99a93283b104b16967783e160f2e014 Fri 2023-03-17 15:51:45 CST Sat 2023-04-15 10:11:57 CST
 -7 056213455b5e4f6f9dea07c233c808ec Sat 2023-04-15 10:12:18 CST Sun 2023-04-16 12:09:19 CST
 -6 dc9b2e2a8a9e4166a699a63c748e9322 Sun 2023-04-16 12:47:46 CST Tue 2023-04-18 20:34:45 CST
 -5 dd7a8c34b9a84d598a90658faf9767bd Tue 2023-04-18 20:36:13 CST Thu 2023-04-20 00:22:33 CST
 -4 7b34bd6a8ccc4b7886606477d3fa880b Thu 2023-04-20 15:57:58 CST Sat 2023-04-29 12:01:33 CST
 -3 282b653f8aff45989654220e708a9001 Sat 2023-04-29 12:48:31 CST Sat 2023-04-29 12:52:33 CST
 -2 06dd1eebfbbe4970bb59559cd3fcca51 Sat 2023-04-29 13:06:27 CST Thu 2023-05-04 23:03:22 CST
 -1 1ab5a0b32c7249ad9eace156dfa47211 Thu 2023-05-04 23:04:20 CST Tue 2023-05-23 07:39:22 CST
  0 91b890e06b654b49a54257184891e744 Tue 2023-05-23 08:23:46 CST Sun 2023-05-28 20:56:33 CST
 ⊙ 20:56:36  ~
```

4.展示硬盘使用情况(--disk-usage)

显示所有日志文件的当前磁盘使用率，这显示了所有存档和活动日志文件的磁盘使用量之和：

```
$ journalctl --disk-usage
Archived and active journals take up 4.0G in the file system.
$
```

归档和活动日志共占用4G大小。

5.校验日志文件内部的一致性(--verify)

此参数会对系统日志文件进行检查，并报告任何可能的损坏或错误。它会检查日志文件的完整性、有效性以及与相关索引文件的一致性：

```
journalctl --verify
```

```
⚙ 20:57:59 ~ journalctl --verify
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@9e76047525b84fa5b1de14c01229eb28-000000000000e57bc-0005e1e4ebd85599.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@40e586edd81a4acaafd2dc5ef88396b1-0000000000000001-0005e92d9c4d7efe.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@0005e0c50f914b6d-32641e6938a7ccc4.journal~
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@9e76047525b84fa5b1de14c01229eb28-00000000000a5e6a-0005e1d7f2c5c53e.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@9de5c4069e3b4a5182f62465ddd61d5b-0000000000013273-0005edd2ef825ef0.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@0005ea7227f1c766-4eb1db95694364fc.journal~
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@b61fea5e5dc5470b9405f29f665d976b-0000000000000001-0005e01e5a9238be.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/user-1001@f779249a98494387a1a5ae2075a8aaf3-0000000000019f95-0005e073c5baed6e.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@9e76047525b84fa5b1de14c01229eb28-000000000007d61a-0005e1ce585b615d.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@0005e223cf6ffd6c-cc089d401f17af2e.journal~
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@b61fea5e5dc5470b9405f29f665d976b-000000000001e4fe-0005e075197d8edf.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@f6276f769caa4cd5853e5cec90312298-0000000000003b065-0005dfb805f44b71.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@9e76047525b84fa5b1de14c01229eb28-00000000000fa9c7-0005e1e92a499e16.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/user-1001@ce32af30bd82412894b1d83d64369531-000000000059e81-0005dfc417cd5420.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@0005f96cc698c657-4dec3197eea5ba4f.journal~
45f580: Unused data (entry_offset==0)
45f5e8: Unused data (entry_offset==0)
45f6d8: Unused data (entry_offset==0)
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@9de5c4069e3b4a5182f62465ddd61d5b-0000000000146440-0005f99b682a5743.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@0005fa724d3ad7dd-5bc8520247c20ace.journal~
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/user-1001@5176e21dbcbf4e71b457827d9ebadaf1-0000000000027c01-0005eeaa7443ffdc.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/user-1001@5176e21dbcbf4e71b457827d9ebadaf1-000000000013a8b-0005ed2b1761b7e8.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/user-1001@ce32af30bd82412894b1d83d64369531-000000000000397a8-0005dfb78ea82cf7.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@0005e00851710001f-35c778fb1f628dc8.journal~
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@0005e22e290ffb8b-1db674a56c0f8a00.journal~
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@0005fa728d508bf8-168636808f863ad5.journal~
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@f6276f769caa4cd5853e5cec90312298-0000000000061469-0005dfc71f684ea7.journal
22b8f98: Unused data (entry_offset==0)
22b9000: Unused data (entry_offset==0)
22b9060: Unused data (entry_offset==0)
22b90c0: Unused data (entry_offset==0)
22b91b0: Unused data (entry_offset==0)
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@9de5c4069e3b4a5182f62465ddd61d5b-0000000000130cf6-0005f8d023b45ba0.journal
PASS: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@fc0489ccaa0f41108340cd3e2777dd04-0000000000000001-0005e22e290f42c6.journal
                                                                                                                                                                92%
```

6.只显示头部信息(--header)

不显示日志内容，只显示日志的头部信息：

```
journalctl --header
```

默认会显示所有的journal日志文件的头部信息，可以使用**--file**参数指定：

```
journalctl --file=</path/xxx.journal> --header
```

```
⚙ 21:21:00 ~ journalctl --file=/var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@9de5c4069e3b4a5182f62465ddd61d5b-000000000015c1a6-0005fa728d492b89.journal --header
File path: /var/log/journal/b8fd7a061aca4427b8f4f27e474d4989/system@9de5c4069e3b4a5182f62465ddd61d5b-000000000015c1a6-0005fa728d492b89.journal
File ID: aaa7e535d1704d87aa2a291552d1986d
Machine ID: b8fd7a061aca4427b8f4f27e474d4989
Boot ID: 1ab5a0b32c7249ad9eace156dfa47211
Sequential number ID: 9de5c4069e3b4a5182f62465ddd61d5b
State: ARCHIVED
Compatible flags:
Incompatible flags: COMPRESSED-ZSTD KEYED-HASH COMPACT
Header size: 264
Arena size: 29828928
Data hash table size: 233016
Field hash table size: 333
Rotate suggested: no
Head sequential number: 1425830 (15c1a6)
Tail sequential number: 1469255 (166b47)
Head realtime timestamp: Sat 2023-04-29 13:06:27 CST (5fa728d492b89)
Tail realtime timestamp: Thu 2023-05-04 23:03:22 CST (5fadf7944b488)
Tail monotonic timestamp: 5d 9h 57min 5.833s (6cec97b468)
Objects: 92515
Entry objects: 43426
Data objects: 46146
Data hash table fill: 19.8%
Field objects: 77
Field hash table fill: 23.1%
Tag objects: 0
Entry array objects: 2864
Deepest field hash chain: 2
Deepest data hash chain: 2
Disk usage: 28.4M
⚙ 21:21:52 ~
```

7.列出可用的日志分类目录(--list-catalog)

将日志目录的消息ID以表格形式列出来，并加上其简短的描述字符串：

```
journalctl --list-catalog
```

```
⊙ 21:25:08  ~  journalctl --list-catalog
0027229ca0644181a76c4e92458afa2e systemd: One or more messages could not be forwarded to syslog
0e4284a0caca4bfc81c0bb6786972673 systemd: Unit skipped
1675d7f172174098b1108bf8c7dc8f5d systemd: DNSSEC validation failed
1b3bb94037f04bbf81028e135a12d293 systemd: Failed to generate valid unit name from path '@MOUNT_POINT@'.
1c0454c1bd2241e0ac6fefb4bc631433 systemd: systemd-udev-settle.service is deprecated.
1dee0369c7fc4736b7099b38ecb46ee7 systemd: Mount point is not empty
24d8d4452573402496068381a6312df2 systemd: A virtual machine or container has been started
3354939424b4456d9802ca8333ed424a systemd: Session @SESSION_ID@ has been terminated
36db2dfa5a9045e1bd4af5f93e1cf057 systemd: DNSSEC mode has been turned off, as server doesn't support it
39f53479d3a045ac8e11786248231fbf systemd: A start job for unit @UNIT@ has finished successfully
3f7d5ef3e54f4302b4f0b143bb270cab systemd: TPM PCR Extended
45f82f4aef7a4bbf942ce861d1f20990 systemd: Time zone change to @TIMEZONE@
4d4408cfd0d144859184d1e65d7c8a65 systemd: A DNSSEC trust anchor has been revoked
50876a9db00f4c40bde1a2ad381c3a1b systemd: The system is configured in a way that might cause problems
58432bd3bace477cb514b56381b8a758 systemd: A virtual machine or container has been terminated
5aadd8e954dc4b1a8c954d63fd9e1137 systemd: Core file was truncated to @SIZE_LIMIT@ bytes.
5eb03494b6584870a536b337290809b3 systemd: Automatic restarting of a unit has been scheduled
641257651c1b4ec9a8624d7a40a9e1e7 systemd: Process @EXECUTABLE@ could not be executed
6bbd95ee977941e497c48be27c254128 systemd: System sleep state @SLEEP@ entered
7ad2d189f7e94e70a38c781354912448 systemd: Unit succeeded
7b05ebc668384222baa8881179cfda54 systemd: A reload job for unit @UNIT@ has finished
7c8a41f37b764941a0e1780b1be2f037 systemd: Initial clock synchronization
7d4958e842da4a758f6c1cdc7b36dcc5 systemd: A start job for unit @UNIT@ has begun execution
7db73c8af0d94eeb822ae04323fe6ab6 systemd: Initial clock bump
8811e6df2a8e40f58a94cea26f8ebf14 systemd: System sleep state @SLEEP@ left
8d45620c1a4348dbb17410da57c60c66 systemd: A new session @SESSION_ID@ has been created for user @USER_ID@
98268866d1d54a499c4e98921d93bc40 systemd: System shutdown initiated
98e322203f7a4ed290d09fe03c09fe15 systemd: Unit process exited
9d1aaa27d60140bd96365438aad20286 systemd: A stop job for unit @UNIT@ has finished
a596d6fe7bfa4994828e72309e95d61e systemd: Messages from a service have been suppressed
ae8f7b866b0347b9af31fe1c80b127c0 systemd: Resources consumed by unit runtime
b07a249cd024414a82dd00cd181378ff systemd: System start-up is now complete
b480325f9c394a7b802c231e51a2752c systemd: Special user @OFFENDING_USER@ configured, this is not safe!
b61fdac612e94b9182285b998843061f systemd: Accepting user/group name @USER_GROUP_NAME@, which does not match strict user/group name rules.
be02cf6855d2428ba40df7e9d022f03d systemd: A start job for unit @UNIT@ has failed
c14aaf76ec284a5fa1f105f88dfb061c systemd: System factory reset initiated
c7a787079b354eaaa9e77b371893cd27 systemd: Time change
d34d037fff1847e6ae669a370e694725 systemd: A reload job for unit @UNIT@ has begun execution
d93fb3c9c24d451a97cea615ce59c00b systemd: The journal has been stopped
d9b373ed55a64feb8242e02dbe79a49c systemd: Unit failed
de5b426a63be47a7b6ac3eaac82e2f6f systemd: A stop job for unit @UNIT@ has begun execution
e7852bfe46784ed0accde04bc864c2d5 systemd: Seat @SEAT_ID@ has now been removed
e9bf28e6e834481bb6f48f548ad13606 systemd: Journal messages have been missed
ec387f577b844b8fa948f33cad9a75e6 systemd: Disk space used by the journal
eed00a68ffd84e31882105fd973abdd1 systemd: User manager start-up is now complete
f77379a8490b408bbe5f6940505a777b systemd: The journal has been started
fc2e22bc6ee647b6b90729ab34a250b1 systemd: Process @COREDUMP_PID@ (@COREDUMP_COMM@) dumped core
fcbefc5da23d428093f97c82a9290f7b systemd: A new seat @SEAT_ID@ is now available
fe6faa94e7774663a0da52717891d8ef systemd: A process of @UNIT@ unit has been killed by the OOM killer.
⊙ 21:25:18  ~
```

每个目录都具有一个唯一的标识符和一个描述，用于识别和描述该目录的用途。

8.显示catalog的内容(--dump-catalog)

显示消息目录的内容，每个条目由两个破折号和ID组成的行隔开（格式与.catalog文件相同）。

```
journalctl --dump-catalog
```

```
 21:31:38  ~ journalctl --dump-catalog
-- 0027229ca0644181a76c4e92458afa2e
Subject: One or more messages could not be forwarded to syslog
Defined-By: systemd
Support: https://lists.freedesktop.org/mailman/listinfo/systemd-devel

One or more messages could not be forwarded to the syslog service
running side-by-side with journald. This usually indicates that the
syslog implementation has not been able to keep up with the speed of
messages queued.

-- 0e4284a0caca4bfc81c0bb6786972673
Subject: Unit skipped
Defined-By: systemd
Support: https://lists.freedesktop.org/mailman/listinfo/systemd-devel

The unit @UNIT@ was skipped due to an ExecCondition= command failure, and has
entered the 'dead' state with result '@UNIT_RESULT@'.

-- 1675d7f172174098b1108bf8c7dc8f5d
Subject: DNSSEC validation failed
Defined-By: systemd
Support: https://lists.freedesktop.org/mailman/listinfo/systemd-devel
Documentation: man:systemd-resolved.service(8)

A DNS query or resource record set failed DNSSEC validation. This is usually
indication that the communication channel used was tampered with.

-- 1b3bb94037f04bbf81028e135a12d293
Subject: Failed to generate valid unit name from path '@MOUNT_POINT@'.
Defined-By: systemd
Support: https://lists.freedesktop.org/mailman/listinfo/systemd-devel

The following mount point path could not be converted into a valid .mount
unit name:

    @MOUNT_POINT@

Typically this means that the path to the mount point is longer than allowed
for valid unit names.
```


当然也可以指定某个catalog：

```
journalctl --dump-catalog <catalog id>
```

```
 21:35:00  ~ journalctl --dump-catalog 0027229ca0644181a76c4e92458afa2e
-- 0027229ca0644181a76c4e92458afa2e
Subject: One or more messages could not be forwarded to syslog
Defined-By: systemd
Support: https://lists.freedesktop.org/mailman/listinfo/systemd-devel

One or more messages could not be forwarded to the syslog service
running side-by-side with journald. This usually indicates that the
syslog implementation has not been able to keep up with the speed of
messages queued.

 21:35:05  ~
```

9.更新catalog(--update-catalog)

更新消息目录索引。每次安装、删除或更新新的目录文件时，都需要执行这个命令，以重建二进制目录索引。

```
journalctl --update-catalog
```

10.同步未写入的日志(--sync)

要求journal的daemon进程将所有尚未写入的日志数据写入备份文件系统并同步所有日志。

```
journalctl --sync
```

六、总结

通过以上示例，**journalctl**可谓是非常强悍的日志查看和分析工具，不仅能对各类系统日志分门别类还支持各种格式化输出。同时还具有与其他工具的集成能力，比如与 **ELK**（Elasticsearch、Logstash 和 Kibana）等日志聚合和分析平台的集成，进一步扩展了日志分析的能力。

上面的参数熟练掌握后，能精准筛选出任何想要的日志，方便快速及时定位系统及服务问题。