一、前言

上一篇对于journalctl的扫盲篇幅过于冗长，对每个参数基本都细致入微讲解剖析，更像是一本厚重的“工具书”。生产上很少用到其中的大部分参数，所以也被催更对journalctl的一些常见用法和使用场景进行汇总，承蒙呼声之高，权当对前文的延续和回应。无论你是初学者还是经验丰富的技术专家，本文都将为你提供有价值的信息和见解。

二、用法汇总

1.基于boot引导筛选日志

首先可通过`--list-boots`参数列出当前系统所有的boot引导：

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

```
 ⊙ 11:51:50  ~ 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 2651b144594544958 95d5c6801721648 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 Fri 2023-06-09 11:53:33 CST
 ⊙ 11:54:13  ~
```

最左边的IDX列为序列号，0表示当前BOOT引导ID，-1表示上一次引导ID，以此类推，右边两列则为每个BOOT引导的时间范围，每次重启系统后都会生成一个新的BOOT ID。

明白此概念后，比如想筛选上一次引导所产生的错误日志，可以是：

```
journalctl -b -1  -g "fail|error"
```

```
⊙ 12:05:05 ~ journalctl -b -1 -g "fail|error" -n 30
May 25 14:59:41 gentoo systemd[1]: run-credentials-systemd\x2dtmpfiles\x2dsetup.service.mount: Failed with result 'exit-code'.
May 25 14:59:38 gentoo systemd[1]: chunzhen.service: Failed with result 'signal'.
May 25 14:56:40 gentoo systemd[1]: systemd-userdbd.service: Failed with result 'watchdog'.
May 23 08:24:19 gentoo kernel: cfg80211: failed to load regulatory.db
May 23 08:24:19 gentoo kernel: platform regulatory.0: Direct firmware load for regulatory.db failed with error -2
May 23 08:24:06 gentoo systemd-vconsole-setup[604]: Failed to import credentials, ignoring: No such file or directory
May 23 08:24:06 gentoo kernel: RAS: Correctable Errors collector initialized.
May 23 08:24:06 gentoo kernel: pci 0000:00:15.3: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:15.4: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:15.5: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:15.6: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:15.7: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:16.3: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:16.4: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:16.5: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:16.6: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:16.7: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:17.3: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:17.4: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:17.5: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:17.6: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:17.7: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:18.2: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:18.3: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:18.4: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:18.5: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:18.6: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:18.7: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:18.7: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:18.6: BAR 13: failed to assign [io  size 0x1000]
⊙ 12:05:09 ~
```

-n 30 显示最近30条日志，不加则显示全部。

或者通过日志定义的优先级来筛选上一次启动期间0-3级(emerge、alert、crit、error)的错误日志：

```
journalctl -b -1 -p 0..3
```

```
⊙ 12:04:03 ~ journalctl -b -1 -p 0..3
May 23 08:24:19 gentoo kernel: piix4_smbus 0000:00:07.3: SMBus Host Controller not enabled!
May 25 14:55:09 gentoo systemd[1]: systemd-userdbd.service: Watchdog timeout (limit 3min)!
May 25 14:57:02 gentoo systemd-coredump[12335]: elfutils disabled, parsing ELF objects not supported
May 25 14:57:02 gentoo systemd-coredump[12336]: elfutils disabled, parsing ELF objects not supported
May 25 14:57:02 gentoo systemd-coredump[12334]: elfutils disabled, parsing ELF objects not supported
May 25 14:57:02 gentoo systemd-coredump[12334]: [↗] Process 11265 (systemd-userwor) of user 0 dumped core.
May 25 14:57:04 gentoo kernel: Out of memory: Killed process 11262 (geekbench_x86_6) total-vm:16247240kB, anon-rss:13191296kB, file-rss:52kB, shmem-rss:0kB, UID:0 pgtables:30280kB
May 25 14:57:02 gentoo systemd-coredump[12336]: [↗] Process 11266 (systemd-userwor) of user 0 dumped core.
May 25 14:57:02 gentoo systemd-coredump[12335]: [↗] Process 868 (systemd-userdbd) of user 0 dumped core.
May 25 14:57:02 gentoo systemd-coredump[12333]: elfutils disabled, parsing ELF objects not supported
May 25 14:57:02 gentoo systemd-coredump[12333]: [↗] Process 11267 (systemd-userwor) of user 0 dumped core.
⊙ 12:04:08 ~ ✔ 3s 3488 0.02 09.06.23
```

2.指定时间范围筛选特定服务的日志

筛选特定systemd服务，使用-u参数，比如-u "sshd"筛选sshd服务的日志。

指定时间范围，则通过--since(-S)和--until(-U)参数实现。

时间格式为标准的年月日时分秒(YYYY-MM-DD HH:MM:SS)：“2023-06-09 18:00:00”，当然你也可以简写为"yesterday"、"today"、"tomorrow"。

筛选从昨天开始到现在的sshd服务日志：

```
journalctl -u sshd --since yesterday
```

```
⊙ 12:02:22 ~ journalctl -u sshd --since yesterday
Jun 08 00:26:27 arch sshd[64513]: Accepted password for root from 192.168.1.3 port 29481 ssh2
Jun 08 00:26:27 arch sshd[64513]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
Jun 09 11:50:19 arch sshd[69576]: Accepted password for root from 192.168.1.3 port 11107 ssh2
Jun 09 11:50:19 arch sshd[69576]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
⊙ 12:13:08 ~
```

当缺省"--unitl(-U)"时，则表示当前时间。

筛选2023年5月1号到5月31号的sshd服务日志并显示最近20条记录：

```
journalctl -u sshd --since "2023-05-01" --until "2023-5-31" -n 20
```

```
 ⊙ 12:18:03  ~ journalctl -u sshd --since "2023-05-01" --until "2023-5-31" -n 20
May 01 14:53:37 arch sshd[2258]: Accepted password for root from [redacted].179.137 port 39125 ssh2
May 01 14:53:37 arch sshd[2258]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 01 19:36:53 arch sshd[2504]: Accepted password for root from [redacted]179.137 port 39126 ssh2
May 01 19:36:53 arch sshd[2504]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 01 21:38:02 arch sshd[3217]: Accepted password for root from [redacted].179.137 port 44236 ssh2
May 01 21:38:02 arch sshd[3217]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 02 13:18:25 arch sshd[3743]: Accepted password for root from [redacted].63.184 port 35023 ssh2
May 02 13:18:25 arch sshd[3743]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 02 13:18:58 arch sshd[3832]: Accepted password for root from [redacted].31.243 port 61041 ssh2
May 02 13:18:58 arch sshd[3832]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 02 15:32:31 arch sshd[3962]: Accepted password for root from [redacted].179.137 port 18488 ssh2
May 02 15:32:31 arch sshd[3962]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 02 20:11:40 arch sshd[4303]: Accepted password for root from [redacted]179.137 port 1850 ssh2
May 02 20:11:40 arch sshd[4303]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 04 21:43:07 arch sshd[5455]: Accepted password for root from [redacted].1.3 port 53429 ssh2
May 04 21:43:07 arch sshd[5455]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
May 04 23:03:19 arch sshd[667]: Received signal 15; terminating.
May 04 23:03:19 arch systemd[1]: Stopping OpenSSH Daemon...
May 04 23:03:20 arch systemd[1]: sshd.service: Deactivated successfully.
May 04 23:03:20 arch systemd[1]: Stopped OpenSSH Daemon.
 ⊙ 12:18:16  ~
```

3.使用正则表达式过滤特定日志

当日志数量庞大时，我们只想过滤特定符合条件内容的日志，此时--grep(-g)参数就派上了用场；journalctl的日志由很多字段组成，其中日志信息内容会记录到**MESSAGE**字段，-g参数的作用域也是这个字段，并支持Perl正则写法，具体语法可以通过**man pcre2pattern**来查看。

同时需要注意，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。

筛选sshd服务登录失败的日志并显示最近30条：

```
journalctl -u sshd --grep "Failed" -n 30
```

```
12:28:24  ~ journalctl -u sshd --grep "Failed" -n 30
May 28 20:04:13 arch sshd[20971]: pam_systemd(sshd:session): Failed to create session: Transport endpoint is not connected
May 28 20:03:59 arch sshd[21068]: Failed password for root from 192.168.1.228 port 45526 ssh2
May 28 20:03:54 arch sshd[21068]: Failed password for root from 192.168.1.228 port 45526 ssh2
May 28 20:03:11 arch sshd[20971]: Failed password for root from 192.168.1.228 port 57644 ssh2
May 28 20:03:07 arch sshd[20971]: Failed password for root from 192.168.1.228 port 57644 ssh2
May 28 20:02:52 arch sshd[20830]: Failed password for root from 192.168.1.228 port 42946 ssh2
May 28 20:01:21 arch sshd[20830]: Failed password for root from 192.168.1.228 port 42946 ssh2
May 28 20:00:26 arch sshd[20639]: Failed password for root from 192.168.1.228 port 37370 ssh2
May 28 20:00:17 arch sshd[20639]: Failed password for root from 192.168.1.228 port 37370 ssh2
May 28 01:47:15 arch sshd[7150]: Failed password for root from 192.168.1.228 port 47812 ssh2
-- Boot 1ab5a0b32c7249ad9eace156dfa47211 --
-- Boot 06dd1eebfbbe4970bb59559cd3fcca51 --
-- Boot 282b653f8aff45989654220e708a9001 --
-- Boot 7b34bd6a8ccc4b7886606477d3fa880b --
-- Boot dd7a8c34b9a84d598a90658faf9767bd --
-- Boot dc9b2e2a8a9e4166a699a63c748e9322 --
-- Boot 056213455b5e4f6f9dea07c233c808ec --
-- Boot f99a93283b104b16967783e160f2e014 --
Apr 12 12:36:34 arch sshd[53160]: Failed password for root from 64.226.64.165 port 45820 ssh2
Apr 12 12:36:33 arch sshd[53158]: Failed password for root from 64.226.64.165 port 45806 ssh2
Apr 12 12:36:33 arch sshd[53150]: Failed password for root from 64.226.64.165 port 45750 ssh2
Apr 12 12:36:33 arch sshd[53156]: Failed password for root from 64.226.64.165 port 45792 ssh2
Apr 12 12:36:33 arch sshd[53154]: Failed password for root from 64.226.64.165 port 45776 ssh2
Apr 12 12:36:33 arch sshd[53152]: Failed password for root from 64.226.64.165 port 45764 ssh2
Apr 12 12:36:32 arch sshd[53148]: Failed password for root from 64.226.64.165 port 45734 ssh2
Apr 12 12:36:32 arch sshd[53146]: Failed password for root from 64.226.64.165 port 45722 ssh2
Apr 12 12:36:32 arch sshd[53144]: Failed password for invalid user roo from 64.226.64.165 port 45718 ssh2
Apr 12 12:36:31 arch sshd[53142]: Failed password for invalid user roo from 64.226.64.165 port 45714 ssh2
-- Boot b8f41f5a560c4a5bb851368faae9e898 --
Feb 14 03:17:05 arch sshd[40743]: Failed password for root from 192.168.1.1 port 37806 ssh2
Feb 14 03:16:51 arch sshd[40741]: Failed password for root from 192.168.1.1 port 56026 ssh2
Feb 14 03:16:45 arch sshd[40741]: Failed password for root from 192.168.1.1 port 56026 ssh2
Feb 14 03:16:35 arch sshd[40741]: Failed password for root from 192.168.1.1 port 56026 ssh2
Feb 14 03:15:44 arch sshd[40731]: Failed password for root from 192.168.1.1 port 41336 ssh2
Feb 14 03:15:38 arch sshd[40731]: Failed password for root from 192.168.1.1 port 41336 ssh2
Feb 14 03:15:26 arch sshd[40731]: Failed password for root from 192.168.1.1 port 41336 ssh2
Feb 14 03:15:17 arch sshd[40715]: Failed password for root from 192.168.1.1 port 45370 ssh2
Feb 14 03:15:09 arch sshd[40715]: Failed password for root from 192.168.1.1 port 45370 ssh2
Feb 14 03:15:02 arch sshd[40715]: Failed password for root from 192.168.1.1 port 45370 ssh2
12:28:27  ~
```

从上图可以观察到，如果日志跨了BOOT ID，会把对应的BOOT ID也展示出来。

筛选prometheus服务的master节点最近五行的错误日志，且不允许分页：

```
journalctl -u prometheus -g '(?i)Web master node.*error' -n 5 --no-pager
```

```
12:32:26  ~ journalctl -u prometheus -g '(?i)Web master node.*error' -n 5 --no-pager      ✓ 12329 0.00 09.06.23
-- Boot 1ab5a0b32c7249ad9eace156dfa47211 --
-- Boot 06dd1eebfbbe4970bb59559cd3fcca51 --
-- Boot 282b653f8aff45989654220e708a9001 --
-- Boot 7b34bd6a8ccc4b7886606477d3fa880b --
-- Boot dd7a8c34b9a84d598a90658faf9767bd --
-- Boot dc9b2e2a8a9e4166a699a63c748e9322 --
-- Boot 056213455b5e4f6f9dea07c233c808ec --
-- Boot f99a93283b104b16967783e160f2e014 --
-- Boot b8f41f5a560c4a5bb851368faae9e898 --
-- Boot 81e669da42904059b2272b4e75c3e84c --
-- Boot 132652d2d3bd4a0ab732205025562c30 --
-- Boot 4a197eb813a44cac9a75464640e0b1b0 --
-- Boot 709bc21614a64157b6d1eebe36c0cf3f --
-- Boot 821e72ec38154ff3b7c34d0db99f37b1 --
-- Boot a0b5baf657a148aaaf2b64feefbee21c --
Nov 19 21:42:48 arch prometheus[555]: ts=2022-11-19T13:42:48.510Z caller=scrape.go:1655 level=warn component="scrape manager" scrape_pool="Web master node" target=http://192.168.1.8
1:9100/metrics msg="Error on ingesting samples that are too old or are too far into the future" num_dropped=1045
Nov 19 21:41:48 arch prometheus[555]: ts=2022-11-19T13:41:48.501Z caller=scrape.go:1655 level=warn component="scrape manager" scrape_pool="Web master node" target=http://192.168.1.8
1:9100/metrics msg="Error on ingesting samples that are too old or are too far into the future" num_dropped=1045
Nov 19 21:40:48 arch prometheus[555]: ts=2022-11-19T13:40:48.502Z caller=scrape.go:1655 level=warn component="scrape manager" scrape_pool="Web master node" target=http://192.168.1.8
1:9100/metrics msg="Error on ingesting samples that are too old or are too far into the future" num_dropped=1045
-- Boot 2651b14459454495895d5c6801721648 --
-- Boot fdbb0dfab2cd453aa7b473eb7e7a153d --
-- Boot 2651b14459454495895d5c6801721648 --
Nov 11 05:48:48 arch prometheus[552]: ts=2022-11-10T21:48:48.500Z caller=scrape.go:1655 level=warn component="scrape manager" scrape_pool="Web master node" target=http://192.168.1.8
1:9100/metrics msg="Error on ingesting samples that are too old or are too far into the future" num_dropped=1045
Nov 11 05:47:48 arch prometheus[552]: ts=2022-11-10T21:47:48.501Z caller=scrape.go:1655 level=warn component="scrape manager" scrape_pool="Web master node" target=http://192.168.1.8
1:9100/metrics msg="Error on ingesting samples that are too old or are too far into the future" num_dropped=1045
12:32:37  ~      ✓ 12330 0.00 09.06.23
```

4.实时跟踪特定服务的日志

journalctl提供了-f(--follow)参数来追踪日志，可实时输出所产生的日志。

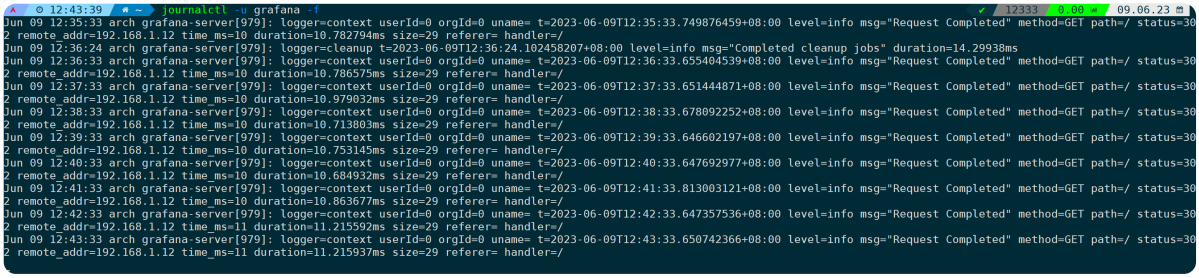
跟踪sshd服务从现在开始所产生的日志：

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

```
12:32:37  ~ journalctl -u sshd -S now -f
Jun 09 12:38:12 arch sshd[69891]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
Jun 09 12:38:14 arch sshd[69891]: Failed password for root from 192.168.1.228 port 39256 ssh2
Jun 09 12:38:25 arch sshd[69891]: Accepted password for root from 192.168.1.228 port 39256 ssh2
Jun 09 12:38:25 arch sshd[69891]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
```

跟踪grafana服务产生的日志，并显示最近10行已经存储的日志：

```
journalctl -u grafana -f
```



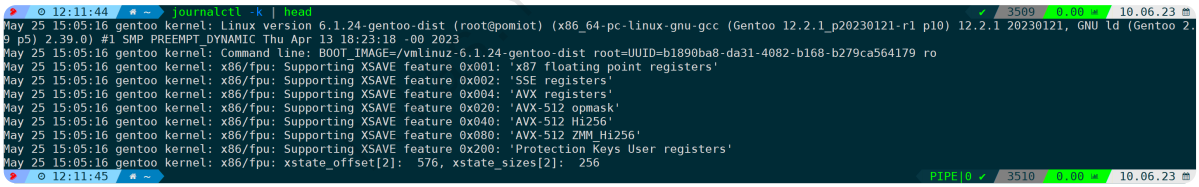
当不加-S now参数时，默认会把最近已产生的10行日志输出出来再进行实时跟踪，如果你想打印已经存储的所有行日志再进行追踪，加上**--no-tail**参数即可：

```
journalctl -u grafana -f --no-tail
```

5.筛选与内核相关的错误日志

--demgs或者**-k**参数j将会打印内核相关日志：

```
journalctl -k
```



而只想看到有没有错误日志，可以通过-p来指定日志优先级，比如**输出错误级别为0-3级(emerge、alert、crit、error)的日志**：

```
jouornalctl -k -p 0..3
```

```
⊙ 12:16:51  ~  journalctl -k -p 0..3
May 23 08:23:54 arch kernel: piix4_smbus 0000:00:07.3: SMBus Host Controller not enabled!
May 31 21:11:11 arch kernel: watchdog: BUG: soft lockup - CPU#61 stuck for 27s! [ksoftirqd/61:388]
May 31 21:11:12 arch kernel: watchdog: BUG: soft lockup - CPU#35 stuck for 23s! [swapper/35:0]
May 31 21:11:12 arch kernel: watchdog: BUG: soft lockup - CPU#50 stuck for 27s! [migration/50:321]
May 31 21:11:29 arch kernel: watchdog: BUG: soft lockup - CPU#48 stuck for 22s! [fail2ban-server:992]
May 31 21:11:30 arch kernel: watchdog: BUG: soft lockup - CPU#37 stuck for 23s! [f2b/a.sshd:1219]
May 31 21:11:30 arch kernel: watchdog: BUG: soft lockup - CPU#45 stuck for 32s! [prometheus:4743]
May 31 21:11:30 arch kernel: watchdog: BUG: soft lockup - CPU#44 stuck for 24s! [ksoftirqd/44:285]
May 31 21:11:30 arch kernel: watchdog: BUG: soft lockup - CPU#33 stuck for 24s! [migration/33:218]
May 31 21:11:30 arch kernel: watchdog: BUG: soft lockup - CPU#47 stuck for 21s! [ksoftirqd/47:303]
May 31 21:11:31 arch kernel: watchdog: BUG: soft lockup - CPU#54 stuck for 30s! [systemd-resolve:924]
May 31 21:11:31 arch kernel: watchdog: BUG: soft lockup - CPU#49 stuck for 25s! [ksoftirqd/49:316]
May 31 21:11:31 arch kernel: watchdog: BUG: soft lockup - CPU#42 stuck for 23s! [migration/42:272]
May 31 21:11:32 arch kernel: watchdog: BUG: soft lockup - CPU#29 stuck for 25s! [ksoftirqd/29:194]
May 31 21:11:32 arch kernel: watchdog: BUG: soft lockup - CPU#25 stuck for 42s! [ntpd:999]
May 31 21:11:32 arch kernel: watchdog: BUG: soft lockup - CPU#51 stuck for 21s! [swapper/51:0]
May 31 21:11:34 arch kernel: watchdog: BUG: soft lockup - CPU#53 stuck for 21s! [kworker/53:2:19010]
May 31 21:11:34 arch kernel: watchdog: BUG: soft lockup - CPU#17 stuck for 29s! [node_exporter:1008]
May 31 21:11:35 arch kernel: watchdog: BUG: soft lockup - CPU#56 stuck for 28s! [dockerd:1228]
May 31 21:11:35 arch kernel: rcu: INFO: rcu_preempt detected stalls on CPUs/tasks:
May 31 21:11:35 arch kernel: rcu:         29-....: (14 ticks this GP) idle=f404/1/0x4000000000000000 softirq=413401/413401 fqs=18
May 31 21:11:35 arch kernel: rcu:         35-...0: (50 ticks this GP) idle=e4cc/1/0x4000000000000004 softirq=350833/350833 fqs=18
May 31 21:11:35 arch kernel: rcu:         45-...0: (18 GPs behind) idle=539c/1/0x4000000000000000 softirq=466081/466081 fqs=18
May 31 21:11:35 arch kernel: rcu:         49-...0: (5 GPs behind) idle=6f44/1/0x4000000000000000 softirq=779803/779804 fqs=18
May 31 21:11:35 arch kernel: rcu:         50-...0: (1 GPs behind) idle=3c9c/1/0x4000000000000000 softirq=653495/653495 fqs=18
May 31 21:11:35 arch kernel: rcu:         61-...0: (1 GPs behind) idle=7bcc/1/0x4000000000000000 softirq=337303/337303 fqs=18
May 31 21:11:35 arch kernel: watchdog: BUG: soft lockup - CPU#41 stuck for 33s! [dockerd:1229]
⊙ 12:17:04  ~
```

默认只会显示本次系统从启动到现在所产生的内核日志，如果想看到上一次的内核日志，则可以加上**-b -1**参数，比如**想要查看上一次从开机到关机**(boot为-1)**所产生的内核日志，且级别为**0-4级(emerge、alert、crit、error、warning)：

```
journalctl -k -p 0..4 -b -1
```

```
⊙ 12:22:12  ~  journalctl -k -p 0..4 -b -1
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_defrag_ipv6 nf_defrag_ipv4
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 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 1f 44 00 00 <65> ff 0d
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
May 10 20:17:27 arch kernel:  __run_timers+0x222/0x2c0
May 10 20:17:27 arch kernel:  __do_softirq+0xd1/0x2c8
May 10 20:17:27 arch kernel:  ? sched_clock_cpu+0xd/0xb0
May 10 20:17:27 arch kernel:  __irq_exit_rcu+0xb7/0xe0
May 10 20:17:27 arch kernel:  sysvec_apic_timer_interrupt+0x72/0x90
```

或者我想通过MESSAGE字段的内容来匹配满足我需要的内核日志：

```
journalctl -k -g "fail|error|bug|out of memory" -b -1 -n -r
```

```
⊙ 12:24:56  ~  journalctl -k -g "fail|error|bug|out of memory" -b -1 -n -r                                                                                    ✔ 3516
May 25 14:57:04 gentoo kernel: Out of memory: Killed process 11262 (geekbench_x86_6) total-vm:16247240kB, anon-rss:13191296kB, file-rss:52kB, shmem-rss:0kB, UID:0 pgtables:30280kB oom_score_adj:0
May 23 08:24:19 gentoo kernel: cfg80211: failed to load regulatory.db
May 23 08:24:19 gentoo kernel: platform regulatory.0: Direct firmware load for regulatory.db failed with error -2
May 23 08:24:17 gentoo systemd[1]: Mounted sys-kernel-debug.mount.
May 23 08:24:17 gentoo systemd[1]: Mounting sys-kernel-debug.mount...
May 23 08:24:06 gentoo kernel: RAS: Correctable Errors collector initialized.
May 23 08:24:06 gentoo kernel: pci 0000:00:15.3: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:15.4: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:15.5: BAR 13: failed to assign [io  size 0x1000]
May 23 08:24:06 gentoo kernel: pci 0000:00:15.6: BAR 13: failed to assign [io  size 0x1000]
⊙ 12:25:21  ~                                                                                                                                               ✔ 3517
```

-n(--lines)不指定数字默认只显示最近10行，**-r(--reverse)**反向显示，从近到远。

6.筛选与身份验证相关的日志

systemd-journald会按照不同设备对日志分门别类，最常见的设备有：

- **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。

然后通过--facility参数来指定设备模块所产生的日志，**比如想筛选与身份验证相关的日志最近30行**：

```
journalctl --facility=auth -n 30
```

```
⊙ 12:47:58  ~  journalctl --facility=auth -n 30
Jun 09 13:10:27 arch systemd-logind[932]: Session 48 logged out. Waiting for processes to exit.
Jun 09 20:56:11 arch sshd[71140]: Accepted password for root from 1.116.5.179 port 35460 ssh2
Jun 09 20:56:11 arch systemd-logind[932]: New session 50 of user root.
Jun 09 21:25:26 arch systemd-logind[932]: Session 50 logged out. Waiting for processes to exit.
Jun 09 23:50:21 arch sshd[71829]: error: kex_exchange_identification: Connection closed by remote host
Jun 09 23:50:21 arch sshd[71829]: Connection closed by 143.42.56.115 port 60000
Jun 10 00:12:57 arch sshd[71889]: Invalid user CoCo from 143.42.56.115 port 43060
Jun 10 00:12:58 arch sshd[71887]: Invalid user 111 from 143.42.56.115 port 43046
Jun 10 00:12:58 arch sshd[71891]: Invalid user CoCo from 143.42.56.115 port 43078
Jun 10 00:12:58 arch sshd[71893]: Invalid user DaisyZhang from 143.42.56.115 port 43104
Jun 10 00:12:59 arch sshd[71895]: Invalid user CoCo from 143.42.56.115 port 43064
Jun 10 00:12:59 arch sshd[990]: error: beginning MaxStartups throttling
Jun 10 00:12:59 arch sshd[990]: drop connection #10 from [143.42.56.115]:39412 on [192.168.1.12]:22 past MaxStartups
Jun 10 00:13:00 arch sshd[71889]: Failed password for invalid user CoCo from 143.42.56.115 port 43060 ssh2
Jun 10 00:13:00 arch sshd[71887]: Failed password for invalid user 111 from 143.42.56.115 port 43046 ssh2
Jun 10 00:13:00 arch sshd[71891]: Failed password for invalid user CoCo from 143.42.56.115 port 43078 ssh2
Jun 10 00:13:00 arch sshd[71893]: Failed password for invalid user DaisyZhang from 143.42.56.115 port 43104 ssh2
Jun 10 00:14:56 arch sshd[71887]: fatal: Timeout before authentication for 143.42.56.115 port 43046
Jun 10 00:14:56 arch sshd[71889]: fatal: Timeout before authentication for 143.42.56.115 port 43060
Jun 10 00:14:57 arch sshd[71891]: fatal: Timeout before authentication for 143.42.56.115 port 43078
Jun 10 00:14:57 arch sshd[71893]: fatal: Timeout before authentication for 143.42.56.115 port 43104
Jun 10 00:14:58 arch sshd[71895]: fatal: Timeout before authentication for 143.42.56.115 port 43064
Jun 10 00:14:58 arch sshd[71897]: fatal: Timeout before authentication for 143.42.56.115 port 43118
Jun 10 00:14:58 arch sshd[71899]: fatal: Timeout before authentication for 143.42.56.115 port 43094
Jun 10 00:14:58 arch sshd[71901]: fatal: Timeout before authentication for 143.42.56.115 port 43138
Jun 10 00:14:58 arch sshd[71903]: fatal: Timeout before authentication for 143.42.56.115 port 39390
Jun 10 00:14:59 arch sshd[71905]: fatal: Timeout before authentication for 143.42.56.115 port 39398
Jun 10 12:10:32 arch sshd[990]: exited MaxStartups throttling after 11:57:33, 1 connections dropped
Jun 10 12:10:32 arch sshd[73586]: Accepted password for root from 192.168.1.3 port 5082 ssh2
Jun 10 12:10:32 arch systemd-logind[932]: New session 51 of user root.
⊙ 12:48:05  ~
```

筛选与身份验证和授权的私有信息相关的日志最近20行：

```
journalctl --facility=authpriv -n 20
```

```
 ⊙ 12:48:05  ~  journalctl --facility=authpriv -n 30
Jun 07 10:23:16 arch sshd[59483]: pam_unix(sshd:session): session closed for user root
Jun 08 00:26:27 arch sshd[64513]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
Jun 08 01:48:07 arch sshd[64513]: pam_unix(sshd:session): session closed for user root
Jun 09 11:50:19 arch sshd[69576]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
Jun 09 12:38:12 arch sshd[69891]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.228  user=root
Jun 09 12:38:25 arch sshd[69891]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
Jun 09 13:10:27 arch sshd[69576]: pam_unix(sshd:session): session closed for user root
Jun 09 13:10:27 arch sshd[69891]: pam_unix(sshd:session): session closed for user root
Jun 09 20:56:11 arch sshd[71140]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
Jun 09 21:25:26 arch sshd[71140]: pam_unix(sshd:session): session closed for user root
Jun 10 00:12:58 arch sshd[71889]: pam_faillock(sshd:auth): User unknown
Jun 10 00:12:58 arch sshd[71889]: pam_systemd_home(sshd:auth): systemd-homed is not available: Unit dbus-org.freedesktop.home1.service not found.
Jun 10 00:12:58 arch sshd[71889]: pam_unix(sshd:auth): check pass; user unknown
Jun 10 00:12:58 arch sshd[71889]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=143.42.56.115
Jun 10 00:12:58 arch sshd[71889]: pam_faillock(sshd:auth): User unknown
Jun 10 00:12:58 arch sshd[71887]: pam_faillock(sshd:auth): User unknown
Jun 10 00:12:58 arch sshd[71887]: pam_unix(sshd:auth): check pass; user unknown
Jun 10 00:12:58 arch sshd[71887]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=143.42.56.115
Jun 10 00:12:58 arch sshd[71887]: pam_faillock(sshd:auth): User unknown
Jun 10 00:12:58 arch sshd[71891]: pam_faillock(sshd:auth): User unknown
Jun 10 00:12:58 arch sshd[71891]: pam_systemd_home(sshd:auth): systemd-homed is not available: Unit dbus-org.freedesktop.home1.service not found.
Jun 10 00:12:58 arch sshd[71891]: pam_unix(sshd:auth): check pass; user unknown
Jun 10 00:12:58 arch sshd[71891]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=143.42.56.115
Jun 10 00:12:58 arch sshd[71891]: pam_faillock(sshd:auth): User unknown
Jun 10 00:12:59 arch sshd[71893]: pam_faillock(sshd:auth): User unknown
Jun 10 00:12:59 arch sshd[71893]: pam_systemd_home(sshd:auth): systemd-homed is not available: Unit dbus-org.freedesktop.home1.service not found.
Jun 10 00:12:59 arch sshd[71893]: pam_unix(sshd:auth): check pass; user unknown
Jun 10 00:12:59 arch sshd[71893]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=143.42.56.115
Jun 10 00:12:59 arch sshd[71893]: pam_faillock(sshd:auth): User unknown
Jun 10 12:10:32 arch sshd[73586]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
 ⊙ 12:49:04  ~
```

这里能详细打印出pam模块的验证过程。

筛选auth模块并且指定sshd标识符，且关键词为fail|invalid|error|timeout的日志最近30行：

```
journalctl --facility=auth -t sshd -g "fail|invalid|error|timeout" -n 30
```

```
 ⊙ 12:57:44  ~  journalctl --facility=auth -t sshd -g "fail|invalid|error|timeout" -n 30
Jun 10 00:14:59 arch sshd[71905]: fatal: Timeout before authentication for 143.42.56.115 port 39398
Jun 10 00:14:58 arch sshd[71903]: fatal: Timeout before authentication for 143.42.56.115 port 39390
Jun 10 00:14:58 arch sshd[71901]: fatal: Timeout before authentication for 143.42.56.115 port 43138
Jun 10 00:14:58 arch sshd[71899]: fatal: Timeout before authentication for 143.42.56.115 port 43094
Jun 10 00:14:58 arch sshd[71897]: fatal: Timeout before authentication for 143.42.56.115 port 43118
Jun 10 00:14:58 arch sshd[71895]: fatal: Timeout before authentication for 143.42.56.115 port 43064
Jun 10 00:14:57 arch sshd[71893]: fatal: Timeout before authentication for 143.42.56.115 port 43104
Jun 10 00:14:57 arch sshd[71891]: fatal: Timeout before authentication for 143.42.56.115 port 43078
Jun 10 00:14:56 arch sshd[71889]: fatal: Timeout before authentication for 143.42.56.115 port 43060
Jun 10 00:14:56 arch sshd[71887]: fatal: Timeout before authentication for 143.42.56.115 port 43046
Jun 10 00:13:00 arch sshd[71893]: Failed password for invalid user DaisyZhang from 143.42.56.115 port 43104 ssh2
Jun 10 00:13:00 arch sshd[71891]: Failed password for invalid user CoCo from 143.42.56.115 port 43078 ssh2
Jun 10 00:13:00 arch sshd[71887]: Failed password for invalid user 111 from 143.42.56.115 port 43046 ssh2
Jun 10 00:13:00 arch sshd[71889]: Failed password for invalid user CoCo from 143.42.56.115 port 43060 ssh2
Jun 10 00:12:59 arch sshd[990]: error: beginning MaxStartups throttling
Jun 10 00:12:59 arch sshd[71895]: Invalid user CoCo from 143.42.56.115 port 43064
Jun 10 00:12:58 arch sshd[71893]: Invalid user DaisyZhang from 143.42.56.115 port 43104
Jun 10 00:12:58 arch sshd[71891]: Invalid user CoCo from 143.42.56.115 port 43078
Jun 10 00:12:58 arch sshd[71887]: Invalid user 111 from 143.42.56.115 port 43046
Jun 10 00:12:57 arch sshd[71889]: Invalid user CoCo from 143.42.56.115 port 43060
Jun 09 23:50:21 arch sshd[71829]: error: kex_exchange_identification: Connection closed by remote host
Jun 09 12:38:14 arch sshd[69891]: Failed password for root from 192.168.1.228 port 39256 ssh2
Jun 03 22:03:32 arch sshd[50128]: banner exchange: Connection from 92.63.196.148 port 65390: invalid format
Jun 03 22:03:32 arch sshd[50128]: error: kex_exchange_identification: banner line contains invalid characters
Jun 03 21:49:29 arch sshd[50094]: banner exchange: Connection from 92.63.196.149 port 64935: invalid format
Jun 03 21:49:29 arch sshd[50094]: error: kex_exchange_identification: banner line contains invalid characters
May 31 04:25:13 arch sshd[23381]: banner exchange: Connection from 89.248.165.7 port 64735: invalid format
May 31 04:25:13 arch sshd[23381]: error: kex_exchange_identification: banner line contains invalid characters
May 30 18:05:51 arch sshd[22795]: error: kex_exchange_identification: read: Connection reset by peer
May 28 20:03:59 arch sshd[21068]: Failed password for root from 192.168.1.228 port 45526 ssh2
 ⊙ 12:57:49  ~
```

从日志可以看出，攻击者IP：143.42.56.115 在尝试多次登录服务器并用户名密码错误后，00:14:56开始的所有登录请求全部超时，因为00:12:59这个时间点已经被fail2ban给屏蔽了：

```
 ⊙ 12:57:49  ~  systemctl status fail2ban.service
● fail2ban.service - Fail2Ban Service
     Loaded: loaded (/usr/lib/systemd/system/fail2ban.service; enabled; preset: disabled)
     Active: active (running) since Tue 2023-05-23 08:25:16 CST; 2 weeks 4 days ago
       Docs: man:fail2ban(1)
   Main PID: 992 (fail2ban-server)
      Tasks: 7 (limit: 38481)
     Memory: 146.2M
        CPU: 39min 9.025s
     CGroup: /system.slice/fail2ban.service
             └─992 /usr/bin/python /usr/bin/fail2ban-server -xf start

May 23 08:25:16 arch systemd[1]: Starting Fail2Ban Service...
May 23 08:25:16 arch systemd[1]: Started Fail2Ban Service.
May 23 08:25:33 arch fail2ban-server[992]: 2023-05-23 08:25:33,522 fail2ban.configreader   [992]: WARNING 'allowipv6' not defined in 'D
May 23 08:25:49 arch fail2ban-server[992]: Server ready
 ⊙ 13:01:58  ~  grep 143.42.56.115 /var/log/fail2ban.log
2023-06-10 00:12:58,146 fail2ban.filter         [992]: INFO    [sshd] Found 143.42.56.115 - 2023-06-10 00:12:57
2023-06-10 00:12:58,496 fail2ban.filter         [992]: INFO    [sshd] Found 143.42.56.115 - 2023-06-10 00:12:58
2023-06-10 00:12:58,899 fail2ban.filter         [992]: INFO    [sshd] Found 143.42.56.115 - 2023-06-10 00:12:58
2023-06-10 00:12:58,900 fail2ban.filter         [992]: INFO    [sshd] Found 143.42.56.115 - 2023-06-10 00:12:58
2023-06-10 00:12:59,175 fail2ban.filter         [992]: INFO    [sshd] Found 143.42.56.115 - 2023-06-10 00:12:59
2023-06-10 00:12:59,252 fail2ban.actions        [992]: NOTICE  [sshd] Ban 143.42.56.115
2023-06-10 00:13:00,497 fail2ban.filter         [992]: INFO    [sshd] Found 143.42.56.115 - 2023-06-10 00:13:00
2023-06-10 00:13:00,928 fail2ban.filter         [992]: INFO    [sshd] Found 143.42.56.115 - 2023-06-10 00:13:00
2023-06-10 00:13:00,929 fail2ban.filter         [992]: INFO    [sshd] Found 143.42.56.115 - 2023-06-10 00:13:00
2023-06-10 00:13:01,246 fail2ban.filter         [992]: INFO    [sshd] Found 143.42.56.115 - 2023-06-10 00:13:00
 ⊙ 13:02:13  ~  journalctl --facility=auth -t sshd -g "fail|invalid|error|timeout" -n 20 -o short-iso
2023-06-10T00:14:59+0800 arch sshd[71905]: fatal: Timeout before authentication for 143.42.56.115 port 39398
2023-06-10T00:14:58+0800 arch sshd[71903]: fatal: Timeout before authentication for 143.42.56.115 port 39390
2023-06-10T00:14:58+0800 arch sshd[71901]: fatal: Timeout before authentication for 143.42.56.115 port 43138
2023-06-10T00:14:58+0800 arch sshd[71899]: fatal: Timeout before authentication for 143.42.56.115 port 43094
2023-06-10T00:14:58+0800 arch sshd[71897]: fatal: Timeout before authentication for 143.42.56.115 port 43118
2023-06-10T00:14:58+0800 arch sshd[71895]: fatal: Timeout before authentication for 143.42.56.115 port 43064
2023-06-10T00:14:57+0800 arch sshd[71893]: fatal: Timeout before authentication for 143.42.56.115 port 43104
2023-06-10T00:14:57+0800 arch sshd[71891]: fatal: Timeout before authentication for 143.42.56.115 port 43078
2023-06-10T00:14:56+0800 arch sshd[71889]: fatal: Timeout before authentication for 143.42.56.115 port 43060
2023-06-10T00:14:56+0800 arch sshd[71887]: fatal: Timeout before authentication for 143.42.56.115 port 43046
2023-06-10T00:13:00+0800 arch sshd[71893]: Failed password for invalid user DaisyZhang from 143.42.56.115 port 43104 ssh2
2023-06-10T00:13:00+0800 arch sshd[71891]: Failed password for invalid user CoCo from 143.42.56.115 port 43078 ssh2
2023-06-10T00:13:00+0800 arch sshd[71887]: Failed password for invalid user 111 from 143.42.56.115 port 43046 ssh2
2023-06-10T00:13:00+0800 arch sshd[71889]: Failed password for invalid user CoCo from 143.42.56.115 port 43060 ssh2
2023-06-10T00:12:59+0800 arch sshd[990]: error: beginning MaxStartups throttling
2023-06-10T00:12:59+0800 arch sshd[71895]: Invalid user CoCo from 143.42.56.115 port 43064
2023-06-10T00:12:58+0800 arch sshd[71893]: Invalid user DaisyZhang from 143.42.56.115 port 43104
2023-06-10T00:12:58+0800 arch sshd[71891]: Invalid user CoCo from 143.42.56.115 port 43078
```

如果你对fail2ban比较感兴趣，可以参照我写的**这篇**。

7.以json格式输出日志

journalctl提供了格式化输出选项，除了json还支持short、verbose等，如果你想了解更多的输出格式，可以参照我**这篇**。

通过-o(--output)参数来指定输出格式，还是拿上一条举例，**筛选auth模块并且指定sshd标识符，且关键词为fail|invalid|error|timeout的日志最近20行，但是以json格式输出**：

```
journalctl --facility=auth -t sshd -g "fail|invalid|error|timeout" -n 20 -o json
```

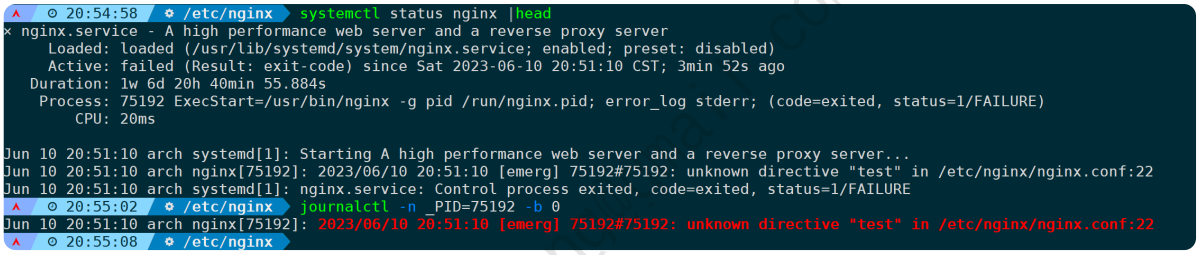
这么看不太直观，通过**json-pretty**参数可有优化输出：

```
journalctl --facility=auth -t sshd -g "fail|invalid|error|timeout" -n 1 -o json-pretty
```

8.根据进程PID过滤日志

当某个服务报错时，我们想通过进程PID来过滤出相关错误日志：

```
journalctl -n _PID=<PID> -b 0
```



-n只显示最近10行，**-b 0**只显示本次系统启动到目前为止的日志。

通过systemctl查看服务状态，可以看到失败时的PID，即使这个服务并没有运行成功，**systemd-journald**也会将记录存储到**_PID**字段，因此当服务并没有正常运行时，你通过lsof、netstat、pidof、ps诸如此类的命令是查不到PID的，不要觉得奇怪，因为它们只能查当前正在运行的进程PID，而systemd会记录进程的PID，不管服务是否正常。

我们不妨通过verbose或json格式来输出，这样你应该能清晰认识到为什么能够通过**_PID**字段来过滤日志：

```
journalctl -n _PID=75192 -b 0 -o verbose
journalctl -n _PID=75192 -b 0 -o json-pretty
```

```
 ⚙ 21:03:17  /etc/nginx  journalctl -n _PID=75192 -b 0 -o verbose
Sat 2023-06-10 20:51:10.538974 CST [s=9de5c4069e3b4a5182f62465ddd61d5b;i=1aea1b;b=91b890e06b654b49a54257184891e744;m=1748afddf40;t=5fdc5f099a4de;x=872b6f1637225a4e]
    _BOOT_ID=91b890e06b654b49a54257184891e744
    _MACHINE_ID=b8fd7a061aca4427b8f4f27e474d4989
    _HOSTNAME=arch
    _RUNTIME_SCOPE=system
    SYSLOG_FACILITY=3
    _UID=0
    _GID=0
    _SYSTEMD_SLICE=system.slice
    PRIORITY=3
    _TRANSPORT=stdout
    _STREAM_ID=e202068cd85749a7bbf17995f8aa4795
    SYSLOG_IDENTIFIER=nginx
    MESSAGE=2023/06/10 20:51:10 [emerg] 75192#75192: unknown directive "test" in /etc/nginx/nginx.conf:22
    _PID=75192
    _COMM=nginx
    _CAP_EFFECTIVE=1fff7fdfffff
    _SYSTEMD_CGROUP=/system.slice/nginx.service
    _SYSTEMD_UNIT=nginx.service
    _SYSTEMD_INVOCATION_ID=ae6e3614e57e4724a210015060ae864e

 ⚙ 21:06:11  /etc/nginx  journalctl -n _PID=75192 -b 0 -o json-pretty
{
        "PRIORITY" : "3",
        "_RUNTIME_SCOPE" : "system",
        "_SYSTEMD_UNIT" : "nginx.service",
        "_UID" : "0",
        "SYSLOG_FACILITY" : "3",
        "_TRANSPORT" : "stdout",
        "MESSAGE" : "2023/06/10 20:51:10 [emerg] 75192#75192: unknown directive \"test\" in /etc/nginx/nginx.conf:22",
        "_MACHINE_ID" : "b8fd7a061aca4427b8f4f27e474d4989",
        "SYSLOG_IDENTIFIER" : "nginx",
        "__REALTIME_TIMESTAMP" : "1686401470538974",
        "__CURSOR" : "s=9de5c4069e3b4a5182f62465ddd61d5b;i=1aea1b;b=91b890e06b654b49a54257184891e744;m=1748afddf40;t=5fdc5f099a4de;x=872b6f1637225a4e",
        "_COMM" : "nginx",
        "_SYSTEMD_INVOCATION_ID" : "ae6e3614e57e4724a210015060ae864e",
        "__MONOTONIC_TIMESTAMP" : "1600059727680",
        "_SYSTEMD_CGROUP" : "/system.slice/nginx.service",
        "_SYSTEMD_SLICE" : "system.slice",
        "_BOOT_ID" : "91b890e06b654b49a54257184891e744",
        "_PID" : "75192",
        "_GID" : "0",
        "_HOSTNAME" : "arch",
        "_STREAM_ID" : "e202068cd85749a7bbf17995f8aa4795",
        "_CAP_EFFECTIVE" : "1fff7fdfffff"
}
 ⚙ 21:06:17  /etc/nginx
```

很明显，完整的日志会存储到不同的字段，每个字段分工明确。不用怀疑，上面的字段你都可以拿来单独作为过滤条件或者组合使用都是没问题的。

比如指定_PID的情况下同时指定_SYSTEMD_UNIT字段：

```
journalctl -n _PID=75192 _SYSTEMD_UNIT=nginx.service -o verbose
journalctl -n _PID=75192 _SYSTEMD_UNIT=nginx.service -o json-pretty
```

```
 ⚙ 21:15:26  /etc/nginx  journalctl -n _PID=75192 _SYSTEMD_UNIT=nginx.service -o verbose
Sat 2023-06-10 20:51:10.538974 CST [s=9de5c4069e3b4a5182f62465ddd61d5b;i=1aea1b;b=91b890e06b654b49a54257184891e744;m=1748afddf40;t=5fdc5f099a4de;x=872b6f1637225a4e]
    _BOOT_ID=91b890e06b654b49a54257184891e744
    _MACHINE_ID=b8fd7a061aca4427b8f4f27e474d4989
    _HOSTNAME=arch
    _RUNTIME_SCOPE=system
    SYSLOG_FACILITY=3
    _UID=0
    _GID=0
    _SYSTEMD_SLICE=system.slice
    PRIORITY=3
    _TRANSPORT=stdout
    _STREAM_ID=e202068cd85749a7bbf17995f8aa4795
    SYSLOG_IDENTIFIER=nginx
    MESSAGE=2023/06/10 20:51:10 [emerg] 75192#75192: unknown directive "test" in /etc/nginx/nginx.conf:22
    _PID=75192
    _COMM=nginx
    _CAP_EFFECTIVE=1fff7fdfffff
    _SYSTEMD_CGROUP=/system.slice/nginx.service
    _SYSTEMD_UNIT=nginx.service
    _SYSTEMD_INVOCATION_ID=ae6e3614e57e4724a210015060ae864e

 ⚙ 21:15:30  /etc/nginx  journalctl -n _PID=75192 _SYSTEMD_UNIT=nginx.service -o json-pretty
{
        "_SYSTEMD_SLICE" : "system.slice",
        "__CURSOR" : "s=9de5c4069e3b4a5182f62465ddd61d5b;i=1aea1b;b=91b890e06b654b49a54257184891e744;m=1748afddf40;t=5fdc5f099a4de;x=872b6f1637225a4e",
        "__REALTIME_TIMESTAMP" : "1686401470538974",
        "SYSLOG_FACILITY" : "3",
        "_SYSTEMD_CGROUP" : "/system.slice/nginx.service",
        "_SYSTEMD_UNIT" : "nginx.service",
        "_RUNTIME_SCOPE" : "system",
        "_TRANSPORT" : "stdout",
        "_BOOT_ID" : "91b890e06b654b49a54257184891e744",
        "_HOSTNAME" : "arch",
        "_STREAM_ID" : "e202068cd85749a7bbf17995f8aa4795",
        "_COMM" : "nginx",
        "_SYSTEMD_INVOCATION_ID" : "ae6e3614e57e4724a210015060ae864e",
        "_CAP_EFFECTIVE" : "1fff7fdfffff",
        "PRIORITY" : "3",
        "__MONOTONIC_TIMESTAMP" : "1600059727680",
        "_PID" : "75192",
        "MESSAGE" : "2023/06/10 20:51:10 [emerg] 75192#75192: unknown directive \"test\" in /etc/nginx/nginx.conf:22",
        "_UID" : "0",
        "SYSLOG_IDENTIFIER" : "nginx",
        "_MACHINE_ID" : "b8fd7a061aca4427b8f4f27e474d4989",
        "_GID" : "0"
}
 ⚙ 21:15:34  /etc/nginx
```

过滤条件越多日志越精准越具备唯一性。

当然你也可以通过-u参数来跟踪特定服务的日志：

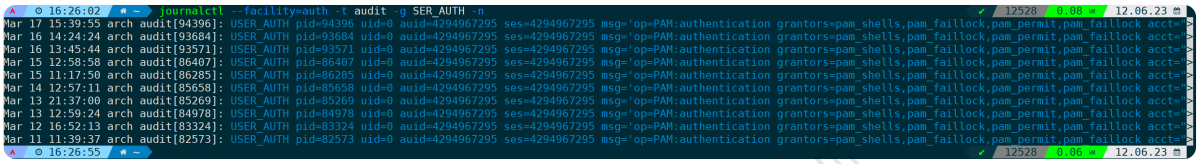
```
journalctl -u nginx.service -n
```

```
⊙ 20:56:17 /etc/nginx journalctl -u nginx.service -n
May 28 00:10:14 arch systemd[1]: Started A high performance web server and a reverse proxy server.
Jun 10 20:51:10 arch systemd[1]: Stopping A high performance web server and a reverse proxy server...
Jun 10 20:51:10 arch systemd[1]: nginx.service: Deactivated successfully.
Jun 10 20:51:10 arch systemd[1]: Stopped A high performance web server and a reverse proxy server.
Jun 10 20:51:10 arch systemd[1]: nginx.service: Consumed 1min 31.177s CPU time.
Jun 10 20:51:10 arch systemd[1]: Starting A high performance web server and a reverse proxy server...
Jun 10 20:51:10 arch nginx[75192]: 2023/06/10 20:51:10 [emerg] 75192#75192: unknown directive "test" in /etc/nginx/nginx.conf:22
Jun 10 20:51:10 arch systemd[1]: nginx.service: Control process exited, code=exited, status=1/FAILURE
Jun 10 20:51:10 arch systemd[1]: nginx.service: Failed with result 'exit-code'.
Jun 10 20:51:10 arch systemd[1]: Failed to start A high performance web server and a reverse proxy server.
⊙ 21:03:17 /etc/nginx
```

9.禁止截断输出和截断输出

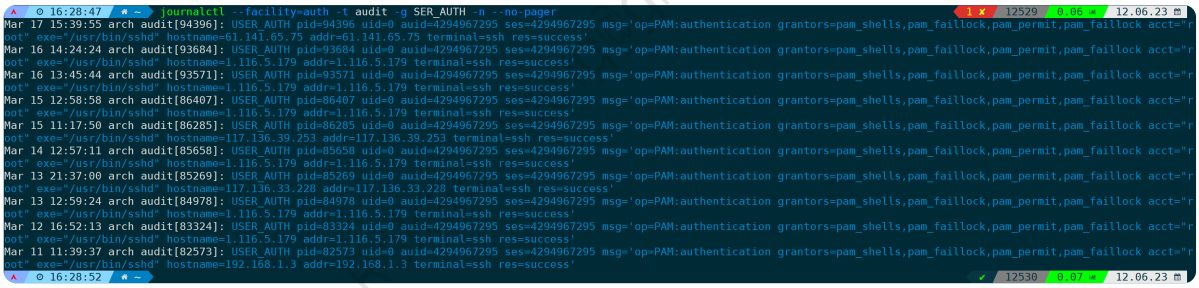
默认情况下，当日志过长，journalctl会截断输出，比如以下这条命令，显示身份验证模块的日志最近10行：

```
journalctl --facility=auth -t audit -g SER_AUTH -n
```

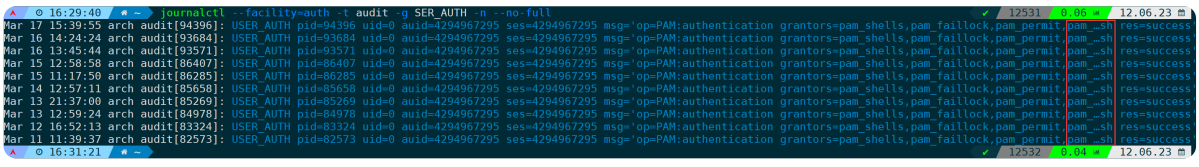


可以看到每一行末尾都用">"字符截断了，看不到完整内容，此时我们加上**--no-pager**参数即可：

```
journalctl --facility=auth -t audit -g SER_AUTH -n --no-pager
```



当然，如果你想省略中间内容，不需要完全显示，可以使用**--no-full**参数，journalctl会用"..."省略这部分内容：



--no-full的意思是如果日志信息太长，需要另起一行，则不显示所有，截断输出中间部分。

10.汇总统计日志字段出现次数

当使用一些组合命令时，可以轻松统计我们想要的字段出现的次数。

比如统计今天的错误日志(**MESSAGE**字段包含：fail|error|fatal的日志)，汇总输出为对应的二进制命令的次数：

```
journalctl --no-pager --since today -g 'fail|error|fatal' -o json | jq '._EXE' | sort | uniq -c | sort -nr -k 1
```

把时间限定去掉：

```
journalctl --no-pager -g 'fail|error|fatal' -o json | jq '._EXE' | sort | uniq -c | sort -nr -k 1
```

```
⊙ 16:59:50 ~ journalctl --no-pager --since today -g 'fail|error|fatal' -o json | jq '._EXE' | sort | uniq -c | sort -nr -k 1
   2048 "/usr/local/bin/mikrotik-exporter"
      2 "/usr/lib/systemd/systemd"
⊙ 17:03:53 ~ journalctl --no-pager -g 'fail|error|fatal' -o json | jq '._EXE' | sort | uniq -c | sort -nr -k 1
690872 "/usr/bin/sshd"
415240 null
400833 "/usr/bin/nginx"
391946 "/usr/local/bin/mikrotik-exporter"
161325 "/usr/bin/dbus-daemon"
  1715 "/usr/local/prometheus/prometheus"
   987 "/usr/lib/systemd/systemd"
   335 "/usr/bin/grafana-server"
   251 "/usr/bin/containerd"
   156 "/usr/lib/systemd/systemd-resolved"
   150 "/usr/bin/udevadm"
   102 "/usr/bin/gpg"
    80 "/usr/bin/grafana"
    68 "/usr/bin/dockerd"
    49 "/usr/bin/ntpd"
    24 "/usr/bin/python3.10"
    12 "/usr/bin/rz"
     6 "/usr/lib/systemd/systemd-logind"
     4 "/usr/bin/login"
     4 "/usr/bin/dhcpcd"
     3 "/opt/javtubeserver/javtube-server"
     1 "/usr/local/node_exporter/node_exporter"
⊙ 17:07:03 ~
```

可以看到其中第二行为null，情况有以下几种：

- 系统日志记录的事件没有与特定的可执行文件关联；
- 可执行文件的信息不可用（比如该文件已被删除或信息丢失）；
- 内核本身产生的日志，不需要与任何可执行文件关联。

上面是根据_EXE字段来统计汇总，那么其它字段也是同理，**比如统计错误日志出现次数，并按照服务字段(_SYSTEMD_UNIT)汇总次数，取TOP 25行**：

```
journalctl --no-pager -g 'fail|error|fatal' -o json | jq '._SYSTEMD_UNIT' | sort | uniq -c | sort -nr -k 1 |head -n 25
```

```
⊙ 17:26:09 ~ journalctl --no-pager -g 'fail|error|fatal' -o json | jq '._SYSTEMD_UNIT' | sort | uniq -c | sort -nr -k 1 |head -n 25
691767 "sshd.service"
413653 null
400833 "nginx.service"
391996 "mikrotik_exporter.service"
161325 "dbus.service"
  1724 "prometheus.service"
   987 "init.scope"
   702 "archlinux-keyring-wkd-sync.service"
   415 "grafana.service"
   229 "docker.service"
   156 "systemd-resolved.service"
   150 "systemd-udevd.service"
    91 "containerd.service"
    49 "ntpd.service"
    24 "fail2ban.service"
    23 "javtube-server.service"
    15 "dnsmasq.service"
     6 "systemd-logind.service"
     5 "process_exporter.service"
     5 "node_exporter.service"
     4 "getty@tty1.service"
     4 "dhcpcd.service"
     4 "blackbox_exporter.service"
     3 "session-70.scope"
     3 "session-20.scope"
⊙ 17:28:42 ~
```

各个字段含义可通过 man 7 systemd.journal-fields 获取。

又或者，**按照系统日志消息存储的类别(_TRANSPORT)来统计**：

```
journalctl --no-pager -o json | jq '._TRANSPORT' | sort | uniq -c | sort -nr -k 1 # 统计范围为所有。
journalctl --no-pager -g 'fail|error|fatal' -o json | jq '._TRANSPORT' | sort | uniq -c | sort -nr -k 1 # 统计范围为匹配fail|error|fatal的日志。
```

- `stdout`：表示消息通过标准输出（stdout）传输，通常是由应用程序直接打印到控制台或输出到文件中。
- `journal`：表示消息通过本地套接字传输，由 `systemd-journald` 接收和处理。这是默认的传输方式，系统日志消息被写入到系统日志文件（通常位于 `/var/log/journal/` 目录下）。
- `syslog`：表示消息通过 `syslog` 协议传输，由 `rsyslog` 或其他类似的日志服务接收和处理。这种传输方式允许将日志消息发送到远程服务器或进行其他配置和处理。
- `journal+console`：表示消息同时通过本地套接字和标准输出传输，消息会同时被写入到系统日志和控制台。
- `journal+syslog`：表示消息同时通过本地套接字和 `syslog` 协议传输，消息会同时被写入到系统日志和 `syslog`。

三、总结

通过本文，我们详细介绍了journalctl工具的用法和应用场景。作为一名系统管理员或开发人员，掌握journalctl的使用技巧对于有效管理和分析系统日志至关重要。

从前面的示例不难看出，journalctl是一个功能强大且灵活的命令行工具，它提供了多种过滤和排序选项，能够快速定位和解决问题。我们学习了如何按时间、进程和日志级别等方式过滤日志，并提到了使用正则表达式过滤日志、相关身份验证模块的介绍和使用、以及汇总统计日志字段的用法，我们还介绍了journalctl的实用功能，包括实时监视、高级搜索和过滤。通过使用日志标识符和元数据，我们能够更精确地定位特定事件，并深入分析应用程序的活动，每个场景拿来用于生产环境都是没有任何问题的。

掌握journalctl带来了许多价值和效益。它不仅能够帮助我们快速诊断故障和进行性能优化，还能用于安全审计和系统稳定性的维护。通过减少故障排除时间，提高工作效率，我们可以提升整个系统的可靠性和可用性。

同时也强烈建议深入学习journalctl的更多功能和用法。可参阅官方文档或者我的**另一篇文章**，探索更多高级特性和实践案例。