journalctl: logi systemd

Poziom: średnio

Dotyczy: dystrybucje z systemd (RHEL/Rocky 9–10 domyślnie, Debian/Ubuntu, SLES)

Czas: ~20 min

journalctl czyta binarny dziennik systemd — logi wszystkich usług, jądra i samego rozruchu w jednym miejscu, z polami do filtrowania zamiast grepowania po tekście. Jeden dziennik zamiast tuzina plików w `/var/log`*.*

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
HTTP	Hypertext Transfer Protocol: protokół stron WWW
JSON	JavaScript Object Notation: tekstowy format danych z klamrami i cudzysłowami
PID	Process ID: numer procesu w systemie
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL
WMS	Warehouse Management System: system zarządzania magazynem

Fiszka

TYLKO ODCZYT

Wyświetlenie całego dziennika: `journalctl`

```
journalctl
```

Pokazuje wszystko od najstarszego zachowanego wpisu, otwarte w pagerze. Bez filtra jednostki obejmuje jądro, wszystkie usługi i cały system — rzadko to, czego naprawdę szukamy (patrz „Częste błędy").

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Przejście od razu na koniec dziennika: `-e`

```
journalctl -e
```

Otwiera pager już przewinięty na koniec dziennika, jak `tail` dla zwykłego pliku — bez ręcznego `Shift+G` w `less`.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Śledzenie dziennika na żywo: `-f`

```
journalctl -f
```

Odpowiednik `tail -f` dla journalda. Nowe wpisy pojawiają się na ekranie w miarę, jak trafiają do dziennika. `Ctrl+C` kończy śledzenie.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Ograniczenie do ostatnich N wpisów: `-n`

```
journalctl -u nginx -n 5
```

Ogranicza wynik do podanej liczby najnowszych wpisów — tu dodatkowo zawężonych do jednej jednostki flagą `-u`, opisaną niżej.

Przykład:

```
Sep 17 08:02:11 srv01 systemd[1]: Starting The nginx HTTP and reverse proxy server...
Sep 17 08:02:11 srv01 nginx[1842]: nginx: configuration file /etc/nginx/nginx.conf test
is successful
Sep 17 08:02:11 srv01 systemd[1]: Started The nginx HTTP and reverse proxy server.
Sep 17 08:14:02 srv01 nginx[1844]: 2026/09/17 08:14:02 [error] 1844#0: *12 connect()
failed (111: Connection refused)
Sep 17 11:00:00 srv01 nginx[1842]: 2026/09/17 11:00:00 [notice] 1842#0: signal 1
(SIGHUP) received, reconfiguring
```

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Odwrócenie kolejności wpisów: `-r`

```
journalctl -r
```

Najnowsze wpisy na górze zamiast na dole — wygodne, gdy interesuje Cię tylko to, co się właśnie stało, i nie chce Ci się przewijać do końca.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Zawężenie do jednej jednostki (albo kilku naraz): `-u`

```
journalctl -u nginx
journalctl -u nginx -u postgresql
```

Zawęża dziennik do jednej jednostki systemd albo — powtórzone — do kilku naraz. To pierwsza flaga, po którą sięgamy niemal zawsze; bez niej dostajemy cały system naraz.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Zawężenie do jednostki `--user`: `--user-unit`

```
journalctl --user-unit demo.service
```

Odpowiednik `-u`, ale dla jednostek uruchamianych w trybie `--user` (strona o systemd). Zwykłe `-u` szuka wśród jednostek systemowych i tam demo.service by nie znalazło.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Zawężenie do priorytetu i pilniejszych: `-p err`

```
journalctl -p err
```

Pokazuje wyłącznie wpisy o priorytecie `err` lub wyższym (pilniejszym) — tabela priorytetów jest na końcu Fiszki. Rzadko używane samodzielnie, prawie zawsze razem z `-u` i `--since`.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Wybór bieżącego albo poprzedniego rozruchu: `-b` / `-b -1`

```
journalctl -b
journalctl -b -1
```

`-b` bez argumentu to bieżący rozruch systemu. `-b -1` to rozruch poprzedni — przydaje się, gdy szukamy przyczyny nieoczekiwanego restartu i chcemy zobaczyć, co działo się tuż przed nim, a nie po nim.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Wylistowanie wszystkich zapisanych rozruchów: `--list-boots`

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

Przykład:

```
-1 3f9a2e1c... Wed 2026-09-16 22:01:03 UTC—Wed 2026-09-16 23:58:41 UTC
 0 8b71d4a0... Thu 2026-09-17 23:59:10 UTC—Thu 2026-09-17 15:22:00 UTC
```

Pierwsza kolumna to numer rozruchu liczony względem bieżącego (`0` to teraz, `-1` to poprzedni) — dokładnie ten numer podajemy do `-b`.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Ograniczenie do komunikatów jądra: `-k`

```
journalctl -k
```

Odpowiednik `dmesg`, ale z historią rozruchów i pełnym filtrowaniem journalctla — w tym możliwością cofnięcia się do jądra z poprzedniego rozruchu przez `-k -b -1`.

> **Pojęcie: dziennik strukturalny.** Wpis w journaldzie to nie jedna linia tekstu, tylko zestaw nazwanych pól: kto go wysłał (`_PID`, `_UID`, `_COMM`), do jakiej jednostki należał (`_SYSTEMD_UNIT`), z którego rozruchu pochodzi (`_BOOT_ID`) i sama treść komunikatu (`MESSAGE`). Dzięki temu journalctl filtruje po konkretnym polu zamiast przeszukiwać tekst jak `grep` — pytanie idzie wprost o wartość pola, a nie o fragment linii, w której akurat mogłaby się znaleźć.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Filtrowanie po konkretnym procesie: `_PID=`

```
journalctl _PID=1842
```

Pokazuje wyłącznie wpisy wysłane przez proces o tym PID-zie — niezależnie od tego, do jakiej jednostki należał. Analogicznie `_UID=1000` filtruje po identyfikatorze użytkownika, który wysłał wpis.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Filtrowanie po nazwie binarki: `_COMM=`

```
journalctl _COMM=sshd
```

Filtruje po nazwie uruchomionego programu, niezależnie od PID-u czy jednostki — wygodne, gdy proces restartował się wielokrotnie i każdy restart miał inny PID.

PID = Process ID: numer procesu w systemie · **journalctl** = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Ograniczenie do zakresu czasu: `--since` / `--until`

```
journalctl --since "10 min ago"
journalctl --since yesterday --until "1 hour ago"
journalctl --since "2026-09-01 08:00" --until "2026-09-01 09:00"
```

Przyjmują zarówno wyrażenia względne (`"10 min ago"`, `yesterday`), jak i sztywne znaczniki daty i godziny. Różnica między tymi dwoma stylami ma znaczenie — patrz „Częste błędy".

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Szukanie wzorca w treści: `-g`

```
journalctl -g "timeout|refused"
journalctl -g "ERROR" --case-sensitive
```

Przeszukuje treść komunikatów wyrażeniem regularnym, tak jak `grep` — w odróżnieniu od filtrów po polu (`_PID=`, `-u`), to jest przeszukiwanie tekstu, więc jest wolniejsze na dużym dzienniku.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Pokazanie mikrosekund w znaczniku czasu: `-o short-precise`

```
journalctl -u nginx -o short-precise
```

Domyślny format czasu ma rozdzielczość sekundy — za mało, gdy porównujemy kolejność zdarzeń z dwóch usług w ciągu tej samej sekundy. `-o verbose` idzie dalej i pokazuje wszystkie pola surowe, nie tylko czas.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Pokazanie pełnych pól w czytelnym JSON-ie: `-o json-pretty`

```
journalctl -u nginx -o json-pretty -n 1
```

Ten sam wpis, co w zwykłym formacie, ale rozbity na wszystkie pola naraz — dokładnie te, o których mowa w ramce „dziennik strukturalny" wyżej.

Przykład:


```
{
        "_BOOT_ID" : "8b71d4a0f2c34e1a9b6d7e5c3a2f1908",
        "_COMM" : "nginx",
        "_PID" : "1842",
        "_UID" : "0",
        "_SYSTEMD_UNIT" : "nginx.service",
        "MESSAGE" : "nginx: configuration file /etc/nginx/nginx.conf test is
successful",
        "PRIORITY" : "6",
        "SYSLOG_IDENTIFIER" : "nginx",
        "__REALTIME_TIMESTAMP" : "1758096131000000"
}
```


Zwykłe `-o json` (bez `-pretty`) daje to samo, ale po jednym obiekcie na linię (JSON Lines) — nieczytelne na ekranie, ale wygodne do przetworzenia przez `jq`.

JSON = JavaScript Object Notation: tekstowy format danych z klamrami i cudzysłowami · **journalctl** = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Pokazanie samego komunikatu bez metadanych: `-o cat`

```
journalctl -u nginx -o cat
```

Odwrotność `json-pretty` — tylko treść `MESSAGE`, bez znacznika czasu, nazwy hosta czy PID-u. Przydaje się, gdy komunikat sam w sobie jest już ustrukturyzowany (np. JSON logowany przez aplikację) i metadane journalda tylko przeszkadzają.

JSON = JavaScript Object Notation: tekstowy format danych z klamrami i cudzysłowami · **journalctl** = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Wyłączenie pagera w skryptach: `--no-pager`

```
journalctl -u nginx --no-pager -n 20
```

Bez tego journalctl może próbować otworzyć `less` w środowisku bez terminala (crontab, inny skrypt) i zawiesić proces w oczekiwaniu na wejście, które nigdy nie nadejdzie. Nawyk w każdym skrypcie: zawsze `--no-pager` albo przekierowanie wyjścia.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Sprawdzenie miejsca zajmowanego przez dziennik: `--disk-usage`

```
journalctl --disk-usage
```

Przykład:

```
Archived and active journals take up 2.1G in the file system.
```

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Przycięcie dziennika do rozmiaru: `--vacuum-size`

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

Przykład:

```
Vacuuming done, freed 1.6G of archived journals from /var/log/journal.
```

Usuwa najstarsze zarchiwizowane pliki dziennika, dopóki łączny rozmiar nie zejdzie poniżej podanej wartości. Nie rusza bieżącego, aktywnego pliku.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Przycięcie dziennika do wieku: `--vacuum-time`

```
sudo journalctl --vacuum-time=30d
```

To samo co `--vacuum-size`, ale kryterium jest wiek wpisów, nie łączny rozmiar — usuwa wszystko starsze niż podany okres.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Sprawdzenie integralności plików dziennika: `--verify`

```
journalctl --verify
```

Sprawdzamy sumy kontrolne plików dziennika na dysku. Sięgamy po to głównie po podejrzeniu uszkodzenia — nagłym zaniku prądu w trakcie zapisu albo błędach systemu plików.

> **Pojęcie: priorytet (sysloga).** Liczba od 0 do 7 dołączana do każdego wpisu, mówiąca, jak pilny jest komunikat — od `emerg` (system nienadający się do użycia) po `debug` (szczegóły techniczne, nieistotne na co dzień). journalctl przyjmuje priorytet jako nazwę albo numer i pokazuje wpisy o **danym poziomie i pilniejszych**, nie tylko dokładnie dopasowane — `-p err` pokaże więc też `crit`, `alert` i `emerg`.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Priorytety sysloga

Numer	Nazwa	Znaczenie
0	emerg	system nie nadaje się do użycia
1	alert	wymaga natychmiastowej reakcji
2	crit	stan krytyczny
3	err	błąd
4	warning	ostrzeżenie
5	notice	istotne, ale normalne zdarzenie
6	info	informacyjne
7	debug	szczegóły debugowania

Uważaj

- `journalctl` bez `-u` wypisuje **wszystko** — jądro, wszystkie usługi, cały system od pierwszego zapisanego rozruchu. Zawsze filtruj.
- `tail -f` nie zadziała na logach journald — to nie są zwykłe pliki tekstowe. Właściwe narzędzie to `journalctl -f`.
- Bez trwałego przechowywania (`Storage=persistent`) dziennik z poprzedniego rozruchu może w ogóle nie istnieć po restarcie.

Z życia wzięte

Z ŻYCIA WZIĘTE

O 6:10 Marek zgłasza, że WMS w magazynie Swarzędz wyrzuca błędy tuż po starcie zmiany. Tomek na ora01 odpala `journalctl -u wms -p err --since today` i od razu widzi tylko wpisy o błędach z dzisiejszego dnia, bez przekopywania się przez cały dziennik. Okazuje się, że przyczyną był restart sieci o 6:02, widoczny w tym samym oknie czasu.

Po co to istnieje

Przed journald logi leżały porozrzucane po dziesiątkach plików: `/var/log/messages`, `/var/log/secure`, `/var/log/maillog`, plus ścieżki własne każdej aplikacji. Każdy plik miał swój format znacznika czasu, żadnych wspólnych pól do filtrowania, a korelacja „co się działo o 8:03 jednocześnie w nginksie i w bazie" oznaczała otwieranie kilku plików naraz i ręczne dopasowywanie linijek po oku.

journald centralizuje to inaczej: przechwytuje stdout/stderr każdej jednostki systemd, komunikaty jądra i wszystko, co aplikacje wyślą przez `syslog()`, i zapisuje to razem z metadanymi — nazwą jednostki, PID-em, UID-em, identyfikatorem rozruchu, ścieżką do binarki — w jednym, indeksowanym dzienniku. Nie musisz już wiedzieć, w którym pliku szukać; mówisz journalctlowi, jaki fragment czasu i jakie pole Cię interesuje, a on odpowiada, zamiast Ciebie grepować dziesięć plików.

Analogia

Pomyśl o **elektronicznej książce wejść na portierni** zamiast papierowego zeszytu.

W papierowym zeszycie każdy wpis to ręcznie zapisana linijka: godzina, nazwisko, cel wizyty. Żeby odpowiedzieć na pytanie „kto wchodził we wtorek między 8 a 9 i miał przepustkę gościnną", musisz kartkować zeszyt strona po stronie, licząc na to, że portier pisał czytelnie i po kolei. Elektroniczna książka wejść odpowiada na to samo pytanie od razu — bo każdy wpis ma pola (godzina, osoba, typ przepustki), a nie tylko wolny tekst.

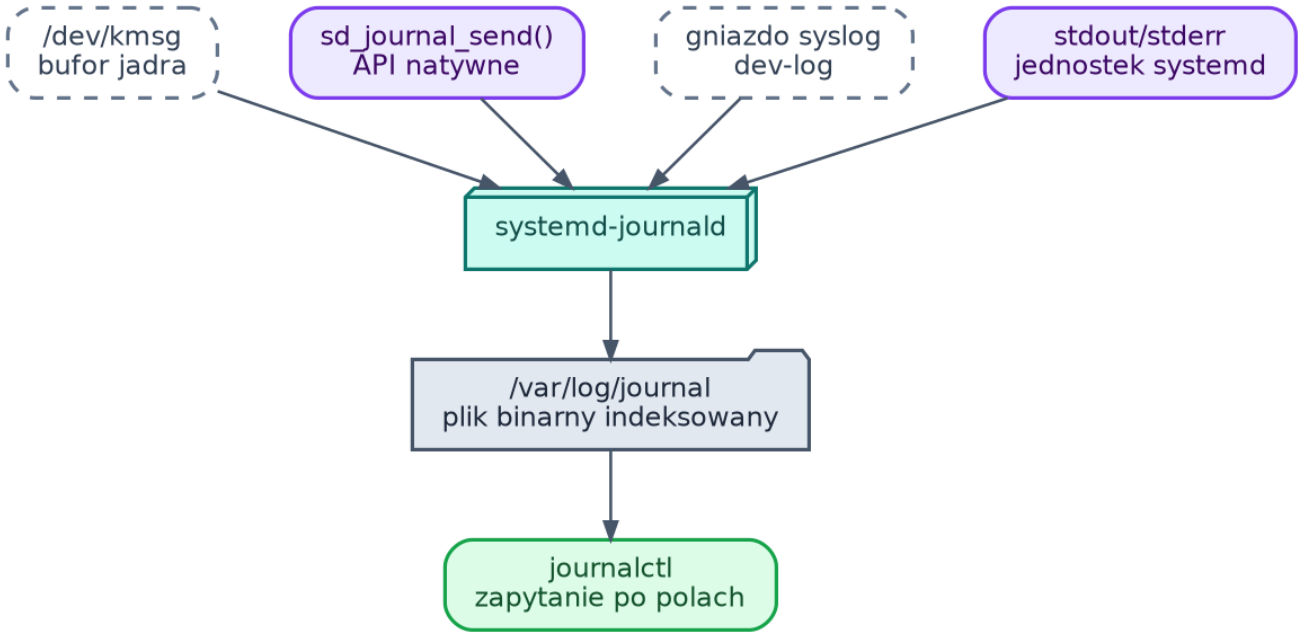
journalctl to dokładnie ta elektroniczna książka. `journalctl -u nginx --since "08:00" --until "09:00" -p err` to właśnie pytanie „kto wchodził we wtorek między 8 a 9 z przepustką gościnną" — odpowiedź przychodzi natychmiast, bo pola są indeksowane, a nie wymagają czytania kartka po kartce.

Ograniczenie tej analogii jest realne: elektroniczna książka potrzebuje prądu i miejsca na dysku, żeby cokolwiek pamiętać. Papierowy zeszyt, raz zapisany, leży na półce archiwum praktycznie za darmo, dopóki ktoś fizycznie nie wyrzuci pudła. Dziennik journald bez skonfigurowanej trwałości (`Storage=persistent`) jest bardziej jak tablica korkowa, którą ktoś czyści przy każdym restarcie — i wymaga świadomej polityki retencji (`vacuum`), żeby nie urosnąć bez końca, czego papierowy zeszyt na półce po prostu nie robi.

Jak to działa

Rozwiń: jak to działa pod spodem

journald zbiera wpisy z czterech źródeł: bufora kołowego jądra (`/dev/kmsg`, odpowiednik `dmesg`, ale z korelacją do konkretnego rozruchu), natywnego API strukturalnego (`sd_journal_send()`, używanego m.in. przez `systemd-cat`), gniazda kompatybilności syslogowej (`/run/systemd/journal/dev-log`, wciąż przyjmującego klasyczne wywołania `syslog(3)`) oraz stdout/stderr każdej jednostki uruchomionej przez systemd, przechwytywanych automatycznie przez gniazdo aktywowane na jednostkę.



journald zbiera wpisy z jądra, API strukturalnego, gniazda syslogowego i stdout jednostek, zapisuje je w indeksowanym dzienniku, a journalctl odpytuje go po polach (kliknięcie otwiera pełny rozmiar)

Przechowywanie. Wpisy trafiają do binarnych, indeksowanych plików per-rozruch. Domyślnie (`Storage=auto` w `/etc/systemd/journald.conf`) journald pisze do `/var/log/journal/<machine-id>/`, jeśli ten katalog istnieje (dziennik trwały, przeżywa restart), a jeśli nie istnieje — do `/run/log/journal/<machine-id>/` na `tmpfs` (dziennik ulotny, znika przy restarcie). Wiele minimalnych obrazów RHEL/Rocky nie tworzy tego katalogu domyślnie, więc logi z poprzednich rozruchów po prostu nie istnieją. Włączenie trwałego logowania:

```
sudo mkdir -p /var/log/journal
sudo systemd-tmpfiles --create --prefix /var/log/journal
sudo systemctl restart systemd-journald
```

Dla pewności, niezależnie od tego, czy katalog już istniał, można to wymusić jawnie w `/etc/systemd/journald.conf`:

```
[Journal]
Storage=persistent
```

Indeksowanie. Pola takie jak `_PID`, `_UID`, `_COMM`, `_SYSTEMD_UNIT`, `MESSAGE`, `PRIORITY` są zapisywane przy każdym wpisie i indeksowane osobno — dlatego filtrowanie po polu (`journalctl _PID=1842`) jest szybkie, a nie liniowym przeszukiwaniem tekstu jak `grep`. To samo dotyczy identyfikatora rozruchu (`_BOOT_ID`) — każdy start systemu dostaje unikalny

identyfikator, i na tym opiera się `-b`, `-b -1` oraz `--list-boots`.

Relacja z rsyslogiem. Na czystym RHEL 9/10 `rsyslog` nie jest już instalowany domyślnie, więc `/var/log/messages` i `/var/log/secure` — tradycyjnie zapisywane właśnie przez rsysloga czytającego z gniazda syslogowego — mogą po prostu nie istnieć. Wszystko, co kiedyś tam trafiało, dziś leży wyłącznie w dzienniku journald. Jeśli `rsyslog` jest zainstalowany (Debian/Ubuntu i SLES częściej mają go domyślnie), działa obok journalda i zwykle konsumuje z niego wpisy (moduł `imjournal`), odtwarzając klasyczne pliki tekstowe dla starych narzędzi. Rotację i przeszukiwanie tych plików opisuje strona *Archiwa i logi rotowane: zcat, zless, zgrep, journalctl*.

Przesyłanie do centralnego loggera. journald może przekazywać wpisy dalej przez `ForwardToSyslog=yes` do lokalnego rsysloga/syslog-ng, który wyśle je do zdalnego kolektora (`@@storage01.ad.zielona.example:514`), albo przez natywne narzędzia `systemd-journal-upload`/`systemd-journal-remote`, przesyłające dziennik do dziennika po HTTP(S) między maszynami, z zachowaniem struktury pól. To osobny, większy temat — tu wystarczy wiedzieć, że opcja istnieje i nie wymaga rezygnacji z journalda na rzecz syslogu.

Krok po kroku

1. Zobacz, co się działo w ostatnich 10 minutach.

```
journalctl --since "10 min ago"
```

Bez filtra jednostki zobaczysz wszystko — jądro, usługi systemowe, sesje logowania. To bywa przytłaczające, ale jest dobrym pierwszym krokiem, kiedy nie wiesz jeszcze, czego szukasz.

2. Zawęź do jednej usługi i tylko błędów.

```
journalctl -u nginx -p err --since today
```

```
Sep 17 08:14:02 srv01 nginx[1842]: 2026/09/17 08:14:02 [error] 1844#0: *12 connect() failed
(111: Connection refused)
```

3. Sprawdź, co się stało w poprzednim rozruchu — np. po nieoczekiwanym restarcie.

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

```
 -1 3f9a2e1c... Wed 2026-09-16 22:01:03 UTC—Wed 2026-09-16 23:58:41 UTC
  0 8b71d4a0... Wed 2026-09-16 23:59:10 UTC—Thu 2026-09-17 15:22:00 UTC
```

```
journalctl -b -1 -p err
```

Jeśli poprzedni rozruch skończył się nagle (awaria zasilania, `kill -9` na całym systemie, panika jądra), ostatnie linijki `journalctl -b -1` tuż przed końcem często zawierają przyczynę.

4. Znajdź komunikaty jądra o konkretnym urządzeniu.

```
journalctl -k -g "sda|nvme0"
```

5. Wyeksportuj wpisy do JSON i przetwórz je dalej.

```
journalctl -u nginx --since "1 hour ago" -o json | jq -r '.MESSAGE'
```

`-o json` daje po jednym obiekcie JSON na linię (JSON Lines) — wygodne do `jq`, ale nieczytelne dla człowieka wprost na ekranie; do czytania służy `json-pretty`, pokazany w Fiszce.

6. Sprawdź, ile miejsca zajmuje dziennik, i przytnij go.

```
journalctl --disk-usage
sudo journalctl --vacuum-size=500M
```

Przykładowe wyniki obu komend są w Fiszce, przy odpowiednich kartach. Trwałe ustawienie limitu (zamiast ręcznego `vacuum` co jakiś czas) to `SystemMaxUse=500M` w `/etc/systemd/journald.conf`, zastosowane po `sudo systemctl restart systemd-journald`.

Częste błędy

`journalctl` bez `-u` — spodziewałeś się logów jednej usługi, dostałeś cały system.

```
$ journalctl | wc -l
284917
```

Bez filtra jednostki journalctl pokazuje dosłownie wszystko od pierwszego zachowanego wpisu — jądro, każdą usługę, każde logowanie. Zawsze zaczynaj od `-u nazwa-usługi`, a dopiero potem dokładaj `-p`, `--since` czy `-g`, żeby zawęzić dalej.

Próba `tail -f` na logu, którego na czystym RHEL 9/10 po prostu nie ma.

```
$ tail -f /var/log/messages
tail: cannot open '/var/log/messages' for reading: No such file or directory
```

Bez zainstalowanego `rsyslog` ten plik nigdy nie powstaje — wszystko ląduje wyłącznie w journaldzie. Właściwe narzędzie do śledzenia na żywo to `journalctl -f`, ewentualnie `journalctl -u nazwa-usługi -f` dla jednej usługi.

Brak trwałego przechowywania — `journalctl -b -1` po restarcie jest puste.

```
$ journalctl -b -1
No journal files were found.
```

Na wielu minimalnych obrazach `/var/log/journal/` nie istnieje, więc journald pisze do ulotnego `/run/log/journal/`, kasowanego przy każdym restarcie. Po awarii, którą akurat chcesz zdiagnozować, logów z niej po prostu nie ma. Włącz trwałość *przed* awarią: `sudo mkdir -p /var/log/journal && sudo systemd-tmpfiles --create --prefix /var/log/journal`.

Zapomniane `--no-pager` w skrypcie — skrypt wisi bez żadnego komunikatu.

```
#!/bin/bash
journalctl -u nginx -n 100 | grep error
```

Uruchomiony interaktywnie działa. Uruchomiony z crona albo z innego skryptu bez terminala może zawiesić się w oczekiwaniu na pager (`less`), zwłaszcza gdy coś w łańcuchu potoków sprawia, że journalctl myśli, że ma do czynienia z terminalem. Nawyk w skryptach: zawsze `--no-pager`, albo przekierowanie, które i tak wymusza brak pagera.

Mylenie granicy `--since "yesterday"` z „24 godziny temu".

```
journalctl --since yesterday
```

`yesterday` to północ **poprzedniej doby kalendarzowej**, nie „24 godziny wstecz od teraz". O 23:50 różnica między tymi dwoma interpretacjami to prawie doba danych więcej albo mniej. Jeśli chodzi Ci dokładnie o „ostatnie 24 godziny", pisz `--since "24 hours ago"` wprost, zamiast polegać na nazwanych granicach dnia.

□□ Ćwiczenia

> Wszystko poniżej wykonujesz na własnej maszynie, w katalogu `/tmp`. Nic nie wymaga uprawnień administratora — działa przez `logger`, który zapisuje wpisy do journalda jako zwykły użytkownik.

Ćwiczenie 1 — własne wpisy i filtrowanie po priorytecie

```
logger -t cwiczenie_dziennika "wpis testowy numer 1"
logger -t cwiczenie_dziennika -p user.err "wpis testowy o priorytecie err"
logger -t cwiczenie_dziennika "wpis testowy numer 2"
```

`-t` ustawia pole `SYSLOG_IDENTIFIER` — filtrujesz po nim tak samo jak po jednostce:

```
journalctl -t cwiczenie_dziennika
journalctl -t cwiczenie_dziennika -p err
```

Czego się spodziewać

Pierwsza komenda pokaże wszystkie trzy wpisy w kolejności czasu. Druga — tylko ten z `-p user.err`, bo pozostałe dwa mają domyślny priorytet `notice`, niższy pilnością niż `err` w tabeli priorytetów.

Jeśli robiłeś ćwiczenie ze strony o systemd i masz jeszcze jednostkę `demo.service` (`--user`), przefiltruj jej log tym samym narzędziem:

```
journalctl --user -u demo.service --since "-10min"
```

Pytania kontrolne

1. Uruchamiasz `journalctl` bez żadnych flag na serwerze produkcyjnym i dostajesz setki tysięcy linii. Co zrobiłeś źle i jak to naprawić?

Odpowiedź

Nie podałeś filtra — bez `-u` (albo innego zawężenia, np. `--since`) journalctl pokazuje cały dziennik od pierwszego zachowanego wpisu, wszystkie jednostki i jądro razem. Naprawa: zacznij od `-u nazwa-usługi`, dołóż `--since`/`-p` w zależności od tego, czego szukasz.

2. `tail -f /var/log/messages` nie działa na świeżo postawionym RHEL 9. Dlaczego, i jaka jest właściwa komenda?

Odpowiedź

Plik prawdopodobnie w ogóle nie istnieje, bo `rsyslog` (który tradycyjnie go zapisywał) nie jest domyślnie instalowany — wszystko trafia wyłącznie do journalda. Właściwa komenda to `journalctl -f`, ewentualnie `journalctl -u nazwa-usługi -f` dla konkretnej usługi.

3. Serwer padł w nocy i restartował się sam. Chcesz zobaczyć logi z rozruchu SPRZED restartu, ale `journalctl -b -1` zwraca `No journal files were found`. Co najprawdopodobniej jest przyczyną?

Odpowiedź

Dziennik nie jest skonfigurowany jako trwały (`Storage=persistent` albo istniejący `/var/log/journal/`) — journald pisał do ulotnego `/run/log/journal/` na `tmpfs`, które zniknęło razem z restartem. Logi z poprzedniego rozruchu przepadły bezpowrotnie; jedyne, co można zrobić, to włączyć trwałość na przyszłość.

4. Czym różni się `--since "yesterday"` od `--since "24 hours ago"`, i kiedy ta różnica ma znaczenie?

Odpowiedź

`yesterday` to sztywna granica: północ poprzedniej doby kalendarzowej do teraz (albo do `--until`). `24 hours ago` to ruchoma granica: dokładnie doba wstecz od bieżącego momentu. Różnica ma znaczenie w każdej godzinie poza samą północą — im później w ciągu dnia sprawdzasz, tym większa rozbieżność między „od wczorajszej północy" a „z ostatnich 24 godzin".

Powiązane: Archiwa i logi rotowane: zcat, zless, zgrep, journalctl · systemd i systemctl: jednostki, zależności, usługi · Obserwacja plików na żywo: tail -f, less +F, watch · head i tail: okno na początek i koniec pliku

Źródła: `man 1 journalctl` · `man 5 journald.conf` · `man 7 systemd.journal-fields` · freedesktop.org — journalctl · systemd: NEWS (Fossies) — `-W`/`--no-hostname` dodane w systemd 259, Varlink `GetEntries()` w systemd 260

Revision #19
Created 2026-09-17 15:07:26 UTC by Tomasz Konieczny
Updated 2026-10-03 19:08:24 UTC by Tomasz Konieczny