Archiwa i logi rotowane: zcat, zless, zgrep, journalctl

Poziom: średnio **Dotyczy: RHEL/Rocky 9–10, Debian/Ubuntu, SLES** **Czas: ~12 min**

Jak przeszukiwać logi, które logrotate spakował i przeniósł, bez rozpakowywania ich na dysku — i jak robić to samo w journaldzie, gdzie logów w ogóle nie ma w postaci plików tekstowych.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
OOM	Out Of Memory: brak pamięci; OOM killer to mechanizm jądra zabijający proces, gdy jej zabraknie
RAM	Random Access Memory: pamięć operacyjna
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL

Fiszka

TYLKO ODCZYT

Podgląd bez rozpakowywania na dysk: zcat

```
zcat /var/log/messages-20260916.gz | tail -n 50
```

Rozpakowujemy gzipowany plik w locie i wypisujemy wynik na standardowe wyjście — na dysku nie powstaje żaden nowy plik. Działa jak `cat`, tylko na skompresowanym źródle.

Przykład:

```
Sep 16 23:58:41 srv01 sshd: Accepted publickey for adm_user
Sep 16 23:59:02 srv01 kernel: [98765.432] link becomes ready
```

zcat = CLI: rozpakuj i wypisz zawartość pliku skompresowanego gzipem na standardowe wyjście, tu do przekazania dalej do psql przy imporcie schematu bazy

TYLKO ODCZYT

Przewijanie skompresowanego pliku jak zwykłego: zless

```
zless /var/log/messages-20260916.gz
```

Mamy pełną nawigację `less` (`/wzorzec` szuka w przód, `G` skacze na koniec, `q` wychodzi) na pliku spakowanym gzipem, dekompresując go w locie w miarę przewijania.

TYLKO ODCZYT

Przeszukanie skompresowanego pliku: zgrep

```
zgrep 'CRITICAL' /var/log/messages-20260916.gz
```

Rozpakowujemy plik w locie i przepuszczamy wynik przez `grep` — przyjmujemy te same flagi co `grep` (`-i`, `-c`, `-v`, `-E`, `-A`, `-B`). Bez tego `grep` widziałby w skompresowanym pliku tylko losowe bajty.

Przykład:

```
/var/log/messages-20260916.gz:Sep 16 14:02:11 srv01 app: CRITICAL disk full
```

TYLKO ODCZYT

Przeszukanie wszystkich rotacji jednym wzorcem: zgrep -h

```
sudo zgrep -h -i 'connection refused' /var/log/messages*
```

Przeszukujemy jednym przebiegiem cały katalog rotacji — skompresowane i nieskompresowane pliki naraz — bez rozpakowywania czegokolwiek na dysk. `-h` wyłącza dopisywanie nazwy pliku przed każdą linią, więc wynik wygląda jak z jednego ciągłego logu.

Przykład:

```
Sep 15 03:41:09 srv01 app: connection refused from 203.0.113.14
Sep 17 08:12:55 srv01 app: connection refused from 203.0.113.22
```

TYLKO ODCZYT

Obsługa innych formatów kompresji: bzcat / xzcat / zstdcat

```
bzcat plik.bz2 | less
xzcat plik.xz | less
zstdcat plik.zst | less
```

Ten sam mechanizm co `zcat`, tylko dla innych algorytmów kompresji: `bzip2` (rzadko dziś spotykany), `xz` (najlepsza kompresja, wolne pakowanie) i `zstd` (bardzo szybkie w obie strony). Każdy ma też swój wariant `grep` (`bzgrep`, `xzgrep`, `zstdgrep`).

TYLKO ODCZYT

Wypisanie spisu zawartości archiwum bez rozpakowywania: tar -tzf

```
tar -tzf backup.tar.gz | grep nginx.conf
```

Wypisujemy listę plików w archiwum `.tar.gz`, nie zapisując niczego na dysk. `-t` to *list*, `-z` mówi `tar`-owi, żeby przepuścił dane przez gzip, `-f` poprzedza nazwę pliku archiwum.

Przykład:

```
etc/nginx/nginx.conf
```

tar = tape archive: archiwum plików (kiedyś na taśmę)

TYLKO ODCZYT

Wyciągnięcie jednego pliku z archiwum: tar -xzf

```
tar -xzf backup.tar.gz etc/nginx/nginx.conf
```

Rozpakowujemy z całego archiwum tylko wskazaną ścieżkę, odtwarzając ją w bieżącym katalogu razem ze strukturą katalogów (`etc/nginx/nginx.conf`). Ścieżka musi być identyczna z tą, którą pokazuje `-tzf` — bez wiodącego ukośnika.

tar = tape archive: archiwum plików (kiedyś na taśmę)

TYLKO ODCZYT

Podgląd pliku z archiwum bez zapisu: tar -xzOf | less

```
tar -xzOf backup.tar.gz etc/nginx/nginx.conf | less
```

`-O` wypisuje wybrany plik na standardowe wyjście zamiast zapisywać go na dysk — do obejrzenia zawartości bez odtwarzania struktury katalogów i bez zostawiania śladu w systemie plików.

tar = tape archive: archiwum plików (kiedyś na taśmę)

TYLKO ODCZYT

Wypisanie logów jednej usługi: journalctl -u

```
journalctl -u NazwaUslugi -n 50
```

Filtrujemy dziennik do wpisów pochodzących z jednej jednostki systemd. `-n 50` ogranicza wynik do ostatnich pięćdziesięciu wpisów, tak jak `tail`.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Ograniczenie do zakresu czasu: journalctl --since

```
journalctl -u NazwaUslugi --since '2026-09-17 08:00' --until '2026-09-17 09:00'
```

Zawężamy wynik do przedziału czasu. `--since` / `--until` rozumieją też formy potoczne: `'2 hours ago'`, `yesterday`, samą godzinę `'09:00'` (domyślnie dzisiaj).

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Filtrowanie po priorytecie: journalctl -p err

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

Pokazujemy tylko wpisy o priorytecie `err` i wyższym (`err`, `crit`, `alert`, `emerg`) — skala priorytetów sysloga, od `0 emerg` do `7 debug`. `-b` ogranicza wynik do bieżącego rozruchu.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Wypisanie logów z poprzedniego rozruchu: journalctl -b -1

```
journalctl -b -1
```

Pokazujemy logi z rozruchu poprzedzającego obecny — `-b -2` cofa o kolejny, `-b 0` (domyślne) to bieżący rozruch. Pierwsze miejsce do sprawdzenia po nieoczekiwanym restarcie.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Wypisanie komunikatów jądra: journalctl -k

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

Odpowiednik `dmesg`, tylko przez `journalctl` i z dostępem do pełnej historii, nie tylko bieżącego rozruchu. Przydaje się przy podejrzeniu OOM killera albo paniki jądra.

OOM = Out Of Memory: brak pamięci; OOM killer to mechanizm jądra zabijający proces, gdy jej zabraknie · **journalctl** = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Sprawdzenie rozmiaru dziennika: journalctl --disk-usage

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

Pokazujemy, ile miejsca na dysku zajmują pliki dziennika łącznie.

Przykład:

```
Archived and active journals take up 3.8G in the file system.
```

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Jednorazowe sprzątanie dziennika: journalctl --vacuum-time

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

Usuwamy wpisy starsze niż podany czas, jednorazowo. Trwałe ograniczenie rozmiaru ustawia się przez `SystemMaxUse=` w `/etc/systemd/journald.conf`, aktywne po restarcie `systemd-journald`.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Wszystkie formaty

Format	Narzędzia	Kompresja	Szybkość
`.gz`	`zcat` `zgrep` `zless` `zdiff`	średnia	szybka; domyślna dla logrotate
`.bz2`	`bzcat` `bzgrep` `bzless`	lepsza od gz	wolna, dziś rzadka
`.xz`	`xzcat` `xzgrep` `xzless`	najlepsza	wolno pakuje, szybko rozpakowuje
`.zst`	`zstdcat` `zstdgrep`	od gz do xz	bardzo szybka w obie strony

Uważaj

- `grep` na katalogu z rotacjami nie przeszuka plików `.gz` — powie `Binary file matches` albo nic. Do tego jest `zgrep`.
- `zgrep` na wielu plikach dokleja nazwę pliku przed każdą linią; `-h` to wyłącza.
- `journalctl` bez katalogu `/var/log/journal` trzyma logi w `/run` i traci je przy restarcie systemu.

Z życia wzięte

Z ŻYCIA WZIĘTE

Poniedziałek, ostatni dzień miesiąca, 22:40. Księgowość na `sql01` zgłasza błąd sprzed trzech dni, a Tomek musi przeszukać dwanaście rotacji logu naraz, część spakowanych gzipem, część nie. Zamiast rozpakowywać każdy plik osobno, odpala `zgrep -h wzorzec /var/log/aplikacja/*`, bo `zgrep` radzi sobie z obiema postaciami jednym poleceniem.

> Znajduje wpis w `messages-20260915.gz` i zamyka zgłoszenie przed północą.

Po co to istnieje

Logi rosną bez końca, a partycja `/var` ma skończony rozmiar. Gdyby `/var/log/messages` po prostu rósł, po kilku miesiącach zapchałby dysk i zatrzymał usługi, które nie mają już gdzie pisać.

> **Pojęcie: rotacja i kompresja logów.** Rotacja to okresowa wymiana pliku logu na nowy, pusty — raz dziennie albo po przekroczeniu rozmiaru stary plik dostaje nazwę z datą, powstaje nowy pusty plik, a najstarsza kopia jest kasowana po przekroczeniu ustalonego limitu liczby kopii. Kompresja to dodatkowy krok wykonywany na już zrotowanych kopiach: tekstowy log ściska się zwykle dziesięciokrotnie, więc trzymanie starszych rotacji w postaci `.gz` pozwala trzymać więcej historii na tej samej powierzchni dysku. Oba mechanizmy razem sprawiają, że katalog `/var/log` ma ograniczony, przewidywalny rozmiar, mimo że usługi piszą do niego bez przerwy. Robi to `logrotate` z timera systemd albo z `cron.daily`.

Katalog wygląda potem tak:

```
ls -l /var/log/messages*
```

```
-rw------- 1 root root  2410984 Sep 17 11:42 /var/log/messages
-rw------- 1 root root 18334721 Sep 17 03:14 /var/log/messages-20260917
-rw------- 1 root root  1204331 Sep 16 03:12 /var/log/messages-20260916.gz
```

Incydent z zeszłego wtorku leży w pliku `.gz`. Rozpakowanie zabiera miejsce, którego może już nie być, i zmienia stan systemu w trakcie diagnozy. Stąd rodzina `z*` — narzędzia, które rozpakowują strumień w locie i nie zostawiają śladu na dysku.

Analogia

To jest **archiwum papierowe firmy**.

Na biurku leżą segregatory z bieżącego miesiąca — otwierasz je i czytasz od ręki. Dokumenty sprzed pół roku są w kartonach w piwnicy, zaklejonych taśmą i opisanych datą, i nie da się ich czytać, dopóki się ich nie otworzy.

Ktoś, kto szuka jednej faktury, znosi karton na górę, rozcina taśmę i wysypuje zawartość na podłogę. Ktoś inny robi w taśmie szparę, przegląda zawartość latarką i odkłada karton na miejsce — piwnica wygląda tak samo jak przed jego wizytą. `zgrep` i `zcat` to ta druga metoda: dane

przelatują przez pamięć procesu, na dysku nie powstaje ani jeden nowy plik.

Analogia niesie ograniczenie, które tłumaczy realną cechę tych narzędzi. Przez szparę nie zajrzysz od razu na stronę 400 — trzeba przewertować wszystko, co jest przed nią. Gzip jest strumieniowy: żeby dotrzeć do końca pliku, trzeba zdekompresować całość od początku. Dlatego `zcat wielki.gz | tail -n 50` nie jest natychmiastowe tak jak `tail -n 50 wielki`, choć komendy wyglądają bliźniaczo. Z tego samego powodu opłaca się przeszukać wszystkie rotacje jednym przebiegiem, zamiast otwierać je po kolei.

Jak to działa

Rozwiń: jak to działa pod spodem

Narzędzia `z*` nie są osobnymi programami do przeszukiwania — to skrypty powłoki (`head -n 20 /usr/bin/zgrep`). `zgrep` sprawdza rozszerzenie pliku, uruchamia odpowiedni dekompresor i podaje jego wyjście do zwykłego `grep`. Konsekwencje są trzy: przyjmuje wszystkie flagi `grep` (`-i`, `-c`, `-v`, `-E`, `-A`, `-B`); radzi sobie także z plikami **nieskompresowanymi**, więc jeden wzorzec obejmuje cały katalog rotacji; i — skoro pod spodem jest `grep` — przy wielu plikach nazwa pliku trafia przed każdą linię, a przy pojedynczym nie. Dlatego ten sam potok raz działa, a raz rozwala `awk '{print $1}'` po drugiej stronie.

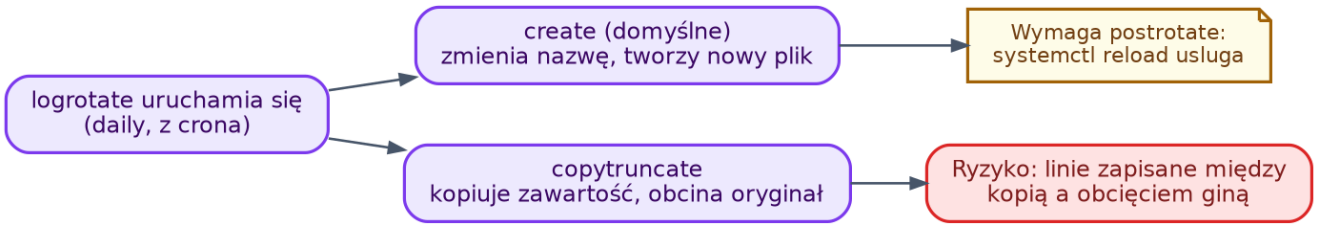
Myląca bywa też kolejność: powłoka rozwija `messages*` alfabetycznie, więc dostajesz `messages`, potem `messages-20260915.gz`, `messages-20260916.gz` i dopiero `messages-20260917`.

logrotate konfiguruje się w `/etc/logrotate.conf` (ustawienia globalne) i w `/etc/logrotate.d/` (po jednym pliku na usługę). Minimalny zestaw dyrektyw wygląda tak:

```
/var/log/aplikacja/*.log {
    daily
    rotate 14
    compress
    delaycompress
    missingok
    notifempty
    create 0640 app app
}
```

`rotate 14` trzyma czternaście kopii, `compress` pakuje gzipem, `create` tworzy nowy pusty plik z podanymi uprawnieniami, a `delaycompress` odkłada kompresję o jeden cykl, żeby proces wciąż trzymający stary deskryptor zdążył dopisać ostatnie linie. Stąd nieskompresowany `messages-`

`20260917` w listingu wyżej.



Dwie strategie logrotate: create wymaga przeładowania usługi, copytruncate ryzykuje utratą linii (kliknięcie otwiera pełny rozmiar)

Najważniejszy wybór to `create` kontra `copytruncate`. Przy `create` (domyślne) logrotate zmienia nazwę pliku i tworzy nowy, a proces pisze dalej do starego i-węzła, dopóki nie dostanie sygnału do ponownego otwarcia logu — stąd `postrotate ... systemctl reload usluga ... endscript`. Przy `copytruncate` logrotate kopiuje zawartość do pliku z datą i obcina oryginał do zera: nic nie trzeba przeładowywać, ale między kopiowaniem a obcięciem mija ułamek sekundy, w którym aplikacja może coś zapisać, i te linie przepadają.

journald działa inaczej. Nie ma tam plików tekstowych — są binarne pliki dziennika w `/var/log/journal/` (albo `/run/log/journal/`, jeśli logi są ulotne), z indeksem, polami strukturalnymi i sumami kontrolnymi. `grep` na nich jest bezużyteczny, jedynym interfejsem jest `journalctl`, a rotacja i limity rozmiaru są wbudowane — logrotate ich nie dotyczy.

Krok po kroku

1. Znajdź wpis we wszystkich rotacjach naraz.

```
sudo zgrep -h -i 'connection refused' /var/log/messages* | tail -n 20
```

Jeden przebieg przez cały katalog, bez rozpakowywania czegokolwiek na dysk. `-h` jest istotne: bez niego przed każdą linią stanie nazwa pliku.

2. Policz wystąpienia w podziale na dni.

```
sudo zgrep -c 'OOM' /var/log/messages*
```

```
/var/log/messages:0
/var/log/messages-20260916.gz:41
/var/log/messages-20260917:3
```

Tu nazwy plików są potrzebne, więc `-h` celowo nie ma — widać od razu, że problem zaczął się 16 września.

3. Przewijaj skompresowany log jak zwykły plik.

`sudo zless /var/log/messages-20260916.gz` daje pełną nawigację `less`: `/wzorzec` szuka w przód, `G` skacze na koniec, `q` wychodzi. Na dużym pliku `G` chwilę trwa, bo dekompresja musi dobiec do końca.

4. Zajrzyj do archiwum tar bez rozpakowywania.

```
tar -tzf /backup/srv01-etc.tar.gz | grep 'nginx.conf'
tar -xzOf /backup/srv01-etc.tar.gz etc/nginx/nginx.conf | less
```

```
etc/nginx/nginx.conf
```

Ścieżka w archiwum jest **względna** (`etc/...`, nie `/etc/...`), bo `tar` przy pakowaniu usuwa wiodący ukośnik — podanie jej z ukośnikiem kończy się `Not found in archive`. `-O` wypisuje plik na standardowe wyjście; bez `-O` wyląduje on w bieżącym katalogu, odtwarzając strukturę `etc/nginx/`.

5. Sprawdź logi usługi w journaldzie.

```
journalctl -u nginx --since '2026-09-17 08:00' --until '2026-09-17 09:00' -p err
```

`-p err` ogranicza wynik do priorytetu `err` i wyższych (`err`, `crit`, `alert`, `emerg`) — skala jest ta sama co w sysloga, od `0 emerg` do `7 debug`. `--since` rozumie też formy potoczne: `'2 hours ago'`, `yesterday`, `'09:00'`.

6. Zajrzyj do poprzedniego rozruchu po nagłym restarcie.

```
journalctl --list-boots | tail -n 3
journalctl -b -1 -p err
journalctl -k -b -1
```

`-b -1` to poprzedni rozruch, `-k` zawęża do komunikatów jądra. Jeśli serwer padł przez panikę jądra albo OOM killera, ślad jest właśnie tutaj.

7. Ogranicz rozmiar dziennika.

```
journalctl --disk-usage          # "journals take up 3.8G in the file system"
sudo journalctl --vacuum-time=30d
```

`--vacuum-*` sprząta jednorazowo; trwałe limity to `SystemMaxUse=` i `MaxRetentionSec=` w `/etc/systemd/journald.conf`, aktywne po restarcie `systemd-journald`.

Częste błędy

`grep` na katalogu z rotacjami i fałszywy wniosek, że błędu nie było.

```
sudo grep -i 'segfault' /var/log/messages*
```

```
/var/log/messages-20260916.gz: binary file matches
```

Albo, częściej, zupełna cisza. `grep` widzi skompresowany plik jako ciąg losowych bajtów i wzorca w nim nie znajdzie, więc raport „w logach nic nie ma" jest nieprawdziwy. Na katalogu logów zawsze `zgrep` — poradzi sobie i z `.gz`, i ze zwykłymi plikami.

`journalctl` nie pokazuje niczego sprzed ostatniego restartu.

```
journalctl -b -1
```

```
Specifying boot ID or boot offset has no effect, no persistent journal was found.
```

Domyślne `Storage=auto` w `/etc/systemd/journald.conf` znaczy: pisz na dysk tylko wtedy, gdy istnieje katalog `/var/log/journal`. Bez niego dziennik siedzi w `/run/log/journal`, czyli w tmpfs, i znika przy każdym rozruchu — a to właśnie logi sprzed restartu są zwykle potrzebne. Włączenie trwałego dziennika:

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

Na Debianie/Ubuntu katalog zwykle już istnieje; na minimalnych obrazach RHEL/Rocky i w kontenerach — często nie.

Zaufanie `copytruncate` przy logu o dużym ruchu.

Aplikacja nie umie przeładować logu, więc ktoś dopisał `copytruncate` i sprawa wydaje się zamknięta. Objaw wychodzi dopiero przy analizie: w archiwum brakuje kilkudziesięciu linii dokładnie wokół godziny 03:12 każdego dnia, czyli w momencie rotacji. To zapisy, które trafiły do

pliku między skopiowaniem a obcięciem. Jeśli log jest podstawą rozliczeń albo audytu, `copytruncate` odpada — trzeba `create` plus `postrotate` z sygnałem do ponownego otwarcia pliku.

□□ Ćwiczenia

> Wszystko poniżej wykonujesz na własnej maszynie, w katalogu `/tmp`. Nic nie wymaga uprawnień administratora.

Ćwiczenie 1 — przeszukaj mieszankę plików skompresowanych i nie

```
mkdir -p /tmp/logi && cd /tmp/logi
printf 'Sep 15 10:00:02 srv01 app: ERROR timeout\n' > app-20260915
printf 'Sep 16 10:00:01 srv01 app: ERROR refused\n' > app-20260916
gzip app-20260915
printf 'Sep 17 10:00:01 srv01 app: OK\n' > app.log
grep -c ERROR app*; zgrep -c ERROR app*; zgrep -h ERROR app*
```

Dlaczego `grep` i `zgrep` dają różne wyniki?

Odpowiedź

`grep` znajduje wpis tylko w `app-20260916`, a `app-20260915.gz` pomija albo raportuje jako binarny — wzorzec `ERROR` fizycznie w nim nie występuje, bo zawartość jest skompresowana. `zgrep` rozpakowuje strumień w locie i znajduje obie linie; `-h` usuwa prefiks z nazwą pliku.

Ćwiczenie 2 — sprawdź stan swojego dziennika

```
ls -ld /var/log/journal 2>/dev/null || echo "brak katalogu — dziennik jest ulotny"
grep -E '^#?Storage' /etc/systemd/journald.conf
journalctl --disk-usage; journalctl -p err -b --no-pager | tail -n 20
```

Jeśli `/var/log/journal` nie istnieje, a `Storage` jest zakomentowane (czyli `auto`), Twoje logi giną przy każdym restarcie. `--no-pager` przydaje się w skryptach: bez niego `journalctl` uruchamia `less` i czeka na klawisz.

Pytania kontrolne

1. Kolega mówi, że przeszukał `grep`-em cały katalog `/var/log` i nie znalazł śladu awarii sprzed tygodnia. Co mu odpowiadasz?

Odpowiedź

Że przeszukał tylko pliki nieskompresowane, czyli mniej więcej ostatnie dwa dni — starsze rotacje są spakowane gzipem i `grep` nie widzi w nich tekstu. Powtórka przez `zgrep -h wzorzec /var/log/messages*` obejmie i `.gz`, i zwykłe pliki. Jeśli usługa loguje przez journald, w plikach tekstowych jej wpisów nie ma w ogóle — wtedy `journalctl -u usluga --since '8 days ago'`.

2. `copytruncate` czy `create` dla aplikacji, której log jest podstawą rozliczeń z klientem?

Odpowiedź

`create` plus sekcja `postrotate` wysyłająca aplikacji sygnał do ponownego otwarcia logu. `copytruncate` ma wbudowany wyścig: między skopiowaniem pliku a obcięciem oryginału aplikacja może zapisać linie, które przepadają, a przy logu rozliczeniowym utrata choćby jednego wpisu dziennie jest nieakceptowalna. `copytruncate` zostaje jako ostateczność dla programów, które nie umieją zareagować na sygnał.

3. `journalctl -b -1` odpowiada `no persistent journal was found`. Gdzie są logi i jak to naprawić?

Odpowiedź

W `/run/log/journal`, czyli w tmpfs — pamięci RAM czyszczonej przy każdym rozruchu. Przyczyną jest `Storage=auto` bez katalogu `/var/log/journal`. Naprawa: `mkdir -p /var/log/journal`, `systemd-tmpfiles --create --prefix /var/log/journal`, `systemctl restart systemd-journald`. Przy okazji ustaw `SystemMaxUse=`, żeby dziennik nie zjadł partycji `/var`.

Powiązane: head i tail: okno na początek i koniec pliku · Pliki, które nie są tekstem: file, strings, hexdump, od, xxd · logrotate — konfiguracja i diagnostyka · journalctl w praktyce

Źródła: `man 1 zgrep` · `man 1 zcat` · `man 8 logrotate` · `man 1 journalctl` · `man 5 journald.conf` · `man 1 tar`

Revision #19
Created 2026-09-17 11:56:36 UTC by Tomasz Konieczny
Updated 2026-10-03 19:07:47 UTC by Tomasz Konieczny