Retencja i czyszczenie dzienników

◫◫ Poziom: średnio

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

▯ Czas: ~18 min

Dziennik systemowy i logi w `/var/log` rosną, dopóki ktoś ich nie ograniczy. Ta strona to o tym, jak bezpiecznie zwolnić miejsce, zamiast ręcznie kasować pliki, których nie da się potem odzyskać — ani, co gorsza, których wciąż potrzebuje jakiś proces.

◫◫ Skróty użyte na tej stronie (3)

Skrót	Co znaczy
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

Fiszka

◫◫ TYLKO ODCZYT

Ile miejsca zajmuje journal

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

Pierwsza komenda przed jakąkolwiek próbą czyszczenia — pokazuje, ile faktycznie zajmuje binarny dziennik `systemd-journald`, zanim zaczniemy decydować, co z tym zrobić.

Przykład:

```
Archived and active journals take up 3.2G in the file system.
```

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Ograniczenie journala do rozmiaru

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

Usuwa najstarsze wpisy dziennika, dopóki jego łączny rozmiar nie zmieści się w podanym limicie. Jednorazowa operacja — nie ustawia trwałego limitu, tylko przycina to, co już jest zapisane, w tej chwili.

Przykład:

```
Vacuuming done, freed 2.7G of archived journals from /var/log/journal.
```

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Ograniczenie journala do wieku wpisów

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

Usuwa wpisy starsze niż podany czas, niezależnie od tego, ile miejsca zajmują. Z produkcji zdarzał się też krótszy wariant, na maszynie z bardzo ograniczonym miejscem na dysku:

```
sudo journalctl --vacuum-time=2days
```

Obie jednostki czasu (`d` / `days`, `h`, `m`, `w`, `month`, `year`) są równoważne w składni `systemd`; wybór zależy tylko od tego, jak krótki okres retencji jest akceptowalny.

Przykład:

```
Vacuuming done, freed 1.1G of archived journals from /var/log/journal.
```

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Ograniczenie liczby plików dziennika

```
sudo journalctl --vacuum-files=5
```

Zamiast rozmiaru albo wieku, ogranicza liczbę pozostawionych plików archiwalnych journala — przydatne, gdy chcemy zachować przewidywalną, małą liczbę plików niezależnie od tego, ile danych każdy z nich zawiera.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Wymuszenie rotacji przed czyszczeniem

```
sudo journalctl --rotate
```

Zamyka bieżący, aktywny plik dziennika i zaczyna nowy — bez tego `--vacuum-*` może nie zwolnić tyle miejsca, ile się spodziewamy, bo aktywny plik (wciąż otwarty do zapisu) nie jest brany pod uwagę przy czyszczeniu. Typowa kolejność w praktyce: `--rotate` najpierw, potem `--vacuum-size` albo `--vacuum-time`.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

`SystemMaxUse` — trwały limit rozmiaru w konfiguracji

```
sudo vim /etc/systemd/journald.conf
```

```
[Journal]
SystemMaxUse=1G
SystemKeepFree=2G
MaxRetentionSec=30day
MaxFileSec=1week
```

Cztery parametry, które razem definiują trwałą politykę retencji, zamiast jednorazowego sprzątania: `SystemMaxUse` to twardy sufit rozmiaru dziennika, `SystemKeepFree` to ilość miejsca na dysku, które journal ma zawsze zostawić wolne niezależnie od własnego limitu, `MaxRetentionSec` to maksymalny wiek pojedynczego wpisu, a `MaxFileSec` to maksymalny czas życia jednego pliku dziennika przed wymuszoną rotacją.

vim = Vi IMproved: ulepszony edytor vi

□□ ZMIENIA SYSTEM

Zastosowanie zmian w `journald.conf`

```
sudo systemctl restart systemd-journald
```

Zmiany w `journald.conf` wymagają restartu samej usługi dziennika, żeby zaczęły obowiązywać — edycja pliku sama w sobie niczego nie zmienia w działającym procesie.

□□ **systemctl** = system control: sterowanie usługami systemd

□□ TYLKO ODCZYT

Ile miejsca zajmuje cały `/var/log`

```
du -sh /var/log
```

W przeciwieństwie do `journalctl --disk-usage`, które liczy wyłącznie binarny dziennik systemd, to polecenie liczy wszystko w `/var/log` — łącznie z logami aplikacji piszących do zwykłych plików tekstowych, obsługiwanych przez `logrotate`, nie przez journal.

Przykład:

```
4.8G□/var/log
```

□□ **du** = disk usage: ile miejsca zajmuje katalog

□□ TYLKO ODCZYT

Które podkatalogi zajmują najwięcej

```
du -sh /var/log/* | sort -h
```

Rozbija poprzedni wynik na poszczególne podkatalogi i pliki, posortowane rosnąco — ostatnia linia wskazuje największego konsumenta miejsca, zwykle pierwszego kandydata do interwencji.

Przykład:

```
4.0K□/var/log/btmp
16K□/var/log/wtmp
124M□/var/log/audit
```

```
890M	/var/log/journal
3.7G	/var/log/httpd
```

du = disk usage: ile miejsca zajmuje katalog

TYLKO ODCZYT

Usunięcie starych, skompresowanych archiwów

```
find /var/log -name '*.gz' -mtime +90 -delete
```

Usuwa skompresowane archiwa logów (typowy efekt działania `logrotate`) starsze niż 90 dni. **Zanim uruchomimy to na produkcji** — sprawdzamy najpierw bez `-delete`, samym `find ... -mtime +90` wypisującym listę, i upewniamy się, że żaden z tych plików nie jest objęty wymogiem dowodowym albo polityką retencji dłuższą niż 90 dni. Skasowany w ten sposób plik nie trafia do żadnego kosza — znika od razu i bezpowrotnie.

find = przeszukaj drzewo katalogów według nazwy, typu, daty albo rozmiaru

TYLKO ODCZYT

Wymuszenie rotacji poza harmonogramem

```
sudo logrotate -f /etc/logrotate.conf
```

`-f` (force) wymusza rotację wszystkich logów objętych konfiguracją `logrotate`, niezależnie od tego, czy według harmonogramu i tak by do niej doszło. Przydatne po ręcznej zmianie `/etc/logrotate.d/`, żeby sprawdzić efekt od razu, bez czekania na najbliższe uruchomienie z crona albo timera systemd.

OSTROŻNIE · NIEODWRACALNE

`truncate -s 0` kontra `rm` na aktywnym logu

```
sudo truncate -s 0 SciezkaDoPliku
```

> (gdzie `SciezkaDoPliku` to ścieżka do aktywnego pliku logu, np. `/var/log/aplikacja/aplikacja.log`)

Zeruje zawartość pliku, zachowując sam plik (ten sam inode, te same uprawnienia) — proces, który ma go otwarty do zapisu, po prostu kontynuuje pisanie od nowa, od zera bajtów, bez przerwy w działaniu. To bezpieczniejsza operacja niż `rm` na pliku wciąż otwartym przez działający proces; pełne wyjaśnienie mechanizmu — dlaczego `rm` w takiej sytuacji nie zwalnia miejsca na dysku, mimo że plik znika z listingu `ls` — jest na stronie o rozjeździe `df` i `du`.

TYLKO ODCZYT

Czyszczenie plików tymczasowych przez systemd

```
sudo systemd-tmpfiles --clean
```

Uruchamia jednorazowo te same reguły czyszczenia, które `systemd-tmpfiles` normalnie wykonuje cyklicznie w tle — usuwa pliki tymczasowe starsze niż wiek zdefiniowany w konfiguracji, niekoniecznie ograniczone do samych logów.

systemd-tmpfiles = systemd tmpfiles: tworzy katalogi i pliki tymczasowe według plików w /etc/tmpfiles.d

ZMIENIA SYSTEM

Konfiguracja reguł czyszczenia w `/etc/tmpfiles.d/`

```
sudo tee /etc/tmpfiles.d/moja-aplikacja.conf <<'EOF'
d /var/log/moja-aplikacja 0750 aplikacja aplikacja 30d
EOF
```

Wiersz mówi: katalog (`d`) `/var/log/moja-aplikacja`, uprawnienia `0750`, właściciel i grupa `aplikacja`, a pliki wewnątrz starsze niż 30 dni (`30d`) mają być usuwane przy najbliższym przebiegu `systemd-tmpfiles`. Ten sam mechanizm, którym systemd sam sprząta `/tmp` i `/var/tmp`, daje się wykorzystać do własnych katalogów logów.

tee = zapisz strumień wejścia do pliku i jednocześnie wypisz go na ekran

TYLKO ODCZYT

Wszystkie polecenia

Polecenie	Co robi

`journalctl --disk-usage`	ile miejsca zajmuje journal
`journalctl --vacuum-size=N`	przycięcie journala do rozmiaru
`journalctl --vacuum-time=N`	usunięcie wpisów starszych niż N
`journalctl --vacuum-files=N`	pozostawienie tylko N plików archiwalnych
`journalctl --rotate`	wymuszenie rotacji przed czyszczeniem
`SystemMaxUse`, `SystemKeepFree`, `MaxRetentionSec`, `MaxFileSec`	trwała polityka w `journald.conf`
`du -sh /var/log`, `du -sh /var/log/* | sort -h`	cofaktyczniezajmujemiejsce
`find ... -mtime +N -delete`	kasowanie starych archiwów, z rozwagą
`logrotate -f`	wymuszona rotacja poza harmonogramem
`truncate -s 0`	wyzerowanie aktywnego logu bez usuwania pliku
`systemd-tmpfiles --clean`	jednorazowe uruchomienie reguł czyszczenia
`/etc/tmpfiles.d/*.conf`	trwałe reguły czyszczenia własnych katalogów

Uważaj

- `rm` na pliku logu, który wciąż trzyma otwarty działający proces, **nie zwalnia miejsca na dysku**, dopóki proces nie zostanie zrestartowany albo nie

zamknie deskryptora — `df` i `du` w tym momencie zaczynają się rozjeżdżać.
- `journalctl --vacuum-*` działa tylko na dzienniku systemd; logi aplikacji w zwykłych plikach tekstowych obsługuje `logrotate`, nie `journalctl`.
- Zanim skasujemy cokolwiek starszego niż kilka dni, sprawdzamy, czy nie obowiązuje wymóg zachowania logów — dowodowy albo z polityki firmy — dłuższy niż to, co właśnie zamierzamy usunąć.

Z życia wzięte

Z ŻYCIA WZIĘTE

Piątek, 15:30: `df -h /var` na app01 pokazuje 96% zajętości mimo że rano wyglądało normalnie, a Tomek w panice robi `rm` na rosnącym logu aplikacji Oli. `df` nadal pokazuje prawie pełny dysk minutę później, choć plik zniknął z `ls`, bo proces aplikacji wciąż trzyma go otwartym i dalej pisze do tego samego, usuniętego i-węzła. Dopiero `truncate -s 0` na tym samym pliku i pozostawienie go pod tą samą nazwą faktycznie zwalnia miejsce, bez restartu aplikacji.

Po co to istnieje

Pojęcie: retencja. Retencja to czas, przez jaki dane — tu: wpisy dziennika systemowego i pliki logów — są świadomie przechowywane, zanim zostaną usunięte. Retencja nie jest tym samym co „ile się zmieści" — to jest decyzja, nie efekt uboczny zapełnionego dysku. Dobrze zaprojektowany system logowania ma jawnie ustaloną retencję (np. 90 dni dla logów aplikacji, rok dla logów audytu) i mechanizmy, które tę retencję egzekwują same, zanim ktokolwiek musi ręcznie kasować pliki w panice, bo `df` pokazuje 100%.

Dwa światy, które ta strona łączy, na pierwszy rzut oka są sprzeczne: polityka zachowania danych (retencja, czasem wymóg dowodowy czy regulacyjny każący trzymać logi miesiącami albo latami) i miejsce na dysku (fizyczny limit, który nie rośnie sam z siebie w miarę jak logi się kumulują). Serwer, na którym `/var/log/journal` rośnie bez ograniczeń, wcześniej czy później zapełni cały wolumin i zabierze miejsce potrzebne aplikacjom — ale bezmyślne, automatyczne kasowanie wszystkiego co starsze niż tydzień może naruszyć wymóg trzymania logów bezpieczeństwa przez rok, wynikający np. z polityki firmy albo wymogów audytu.

Analogia

Retencja logów działa jak **archiwum firmowej dokumentacji papierowej** w piwnicy biura.

Dokumenty księgowe trzeba trzymać przez określoną liczbę lat, bo tak nakazuje przepis — nie dlatego, że ktoś ich regularnie potrzebuje, tylko na wypadek kontroli. Notatki ze spotkań można wyrzucić po kwartale, bo nikt po nich nie sięga. Piwnica ma jednak skończoną pojemność, więc ktoś musi zdecydować: co idzie do niszczarki po ustalonym czasie, a co zostaje, bo wymóg prawny jest dłuższy niż to, na co starczyłoby miejsca. Osoba, która w panice, widząc zapełnioną piwnicę, wyrzuca losowe pudła bez sprawdzenia etykiet, ryzykuje wyrzucenie akurat tych dokumentów księgowych, których kontrola zażąda za pół roku.

Druga część analogii, dokładnie odpowiadająca pułapce z `rm` na otwartym pliku: wyrzucenie pudła z piwnicy nie zwalnia miejsca, jeśli w tym samym czasie ktoś z działu księgowości trzyma kopię tych samych dokumentów rozłożoną na swoim biurku i wciąż do niej dopisuje — fizyczna przestrzeń w piwnicy owszem, zwalnia się na papierze (pudło zniknęło z listy), ale realne zużycie miejsca (biurko wciąż zawalone) nie zmienia się, dopóki ktoś nie skończy pracy nad tymi dokumentami i nie odłoży ich na dobre.

Jak to działa

Rozwiń: jak to działa pod spodem

Dziennik `systemd-journald` to plik binarny (a właściwie zestaw plików, po jednym na okres rotacji) w `/var/log/journal/`, zupełnie inny mechanizm niż tradycyjne pliki tekstowe w `/var/log` obsługiwane przez `logrotate`. `journalctl --vacuum-*` operuje wyłącznie na tych binarnych plikach dziennika systemd — nie dotyka logów aplikacji piszących bezpośrednio do własnych plików tekstowych (np. serwera WWW piszącego do `/var/log/httpd/access_log`), którymi zajmuje się osobny mechanizm, `logrotate`, opisany na własnej stronie.

> **Pojęcie: polityka zachowania danych kontra miejsce na dysku.** Konfiguracja w `journald.conf` pozwala ustawić oba typy ograniczeń naraz i niezależnie: `SystemMaxUse` / `SystemKeepFree` to ograniczenia **przestrzenne** (ile miejsca journal może zająć, ile musi zostawić wolnego), a `MaxRetentionSec` / `MaxFileSec` to ograniczenia **czasowe** (jak długo wpis może żyć, zanim zostanie usunięty niezależnie od zajmowanego miejsca). Journal egzekwuje oba naraz — wpis znika, gdy tylko jeden z warunków zostanie przekroczony, który by nie był pierwszy. Jednorazowe polecenia (`--vacuum-size`, `--vacuum-time`) robią dokładnie to samo, ale ad hoc, w danej chwili, bez trwałej zmiany zachowania na

przyszłość — dziennik po takim jednorazowym czyszczeniu będzie dalej rósł tak samo jak wcześniej, dopóki ktoś nie ustawi trwałych limitów w `journald.conf` albo nie powtórzy czyszczenia ręcznie.

Pojęcie: wymóg dowodowy logów. W wielu środowiskach logi bezpieczeństwa (logowania, zmiany konfiguracji, zdarzenia audytu opisane na osobnej stronie tego rozdziału) muszą być przechowywane przez określony czas nie dlatego, że są do czegoś potrzebne na co dzień, tylko dlatego, że mogą być potrzebne jako dowód przy dochodzeniu incydentu bezpieczeństwa, kontroli albo sporze prawnym. Skasowanie takich logów przed upływem wymaganego okresu — nawet przypadkowo, przez zbyt agresywny `--vacuum-time` albo pochopny `find -delete` — może być naruszeniem polityki firmy albo, w niektórych branżach, wymogu regulacyjnego. Dlatego pierwsze pytanie przed jakimkolwiek czyszczeniem logów bezpieczeństwa nie brzmi „ile miejsca to zwolni", tylko „czy ktokolwiek może tych logów jeszcze potrzebować, i przez jak długo z mocy polityki albo przepisu".

Osobny mechanizm, `systemd-tmpfiles`, sprząta pliki tymczasowe (nie tylko logi) na podstawie reguł wieku zdefiniowanych w `/etc/tmpfiles.d/` — to ten sam system, który domyślnie czyści `/tmp` i `/var/tmp` ze starych plików, i który można wykorzystać do własnych katalogów, dopisując regułę z odpowiednim wiekiem (`30d` w przykładzie z Fiszki).

Krok po kroku

1. Sprawdź, ile faktycznie zajmuje journal i cały `/var/log`, zanim zaczniesz.

```
journalctl --disk-usage
du -sh /var/log
du -sh /var/log/* | sort -h
```

2. Ustal, czy jest polityka retencji, której musisz dotrzymać.

Nieformalne pytanie do zespołu albo sprawdzenie dokumentacji wewnętrznej — zanim skasujesz cokolwiek, upewnij się, że nie naruszasz wymogu dłuższego niż to, co zamierzasz zostawić.

3. Wymuś rotację, potem przytnij jednorazowo do bezpiecznego rozmiaru.

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

4. Ustaw trwałą politykę, żeby nie trzeba było robić tego ręcznie za miesiąc.

```
[Journal]
SystemMaxUse=1G
SystemKeepFree=2G
MaxRetentionSec=90day
```

```
sudo systemctl restart systemd-journald
```

5. Dla logów aplikacji poza journalem — sprawdź i, jeśli trzeba, wymuś `logrotate`.

```
cat /etc/logrotate.d/moja-aplikacja
sudo logrotate -f /etc/logrotate.d/moja-aplikacja
```

6. Zweryfikuj efekt.

```
journalctl --disk-usage
du -sh /var/log
df -h /var/log
```

Częste błędy

`rm` na aktywnym pliku logu nie zwalnia miejsca.

```
$ sudo rm /var/log/aplikacja/aplikacja.log
$ df -h /var
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/vg-var   20G   19G  980M  96% /var
```

Plik znika z listingu `ls`, ale `df` nadal pokazuje niemal pełny dysk — bo proces aplikacji wciąż trzyma otwarty deskryptor do usuniętego pliku i dalej do niego pisze; jądro nie zwalnia bloków dysku, dopóki ostatni proces trzymający plik otwarty go nie zamknie.



Rm na aktywnym pliku usuwa tylko wpis katalogowy i nie zwalnia miejsca, dopóki proces pisze; truncate zeruje ten sam i-węzeł i proces kontynuuje bez przerwy (kliknięcie otwiera pełny rozmiar)

Naprawa: `truncate -s 0` zamiast `rm` na aktywnym logu, albo restart procesu po `rm`, żeby zamknął stary deskryptor i otworzył nowy plik od zera. Pełny mechanizm tego zjawiska (i jak je zdiagnozować przez `lsof`) opisuje strona o rozjeździe `df` i `du`.

Ręczne kasowanie plików w `/var/log/journal/` zamiast `--vacuum-*`.

```
sudo rm -rf /var/log/journal/*
```

Działa, ale ryzykownie — jeśli w momencie kasowania `systemd-journald` akurat pisze do jednego z usuwanych plików, proces może zakończyć się błędem albo zostawić dziennik w niespójnym stanie

przy następnym starcie. `journalctl --vacuum-size`/`--vacuum-time` robi to samo bezpiecznie, przez oficjalny interfejs, który wie, które pliki są aktywne, a które można bezpiecznie usunąć.

Zbyt agresywny `--vacuum-time` kasujący logi objęte wymogiem retencji.

```
sudo journalctl --vacuum-time=7d
```

Uruchomione na serwerze, na którym polityka bezpieczeństwa firmy wymaga trzymania logów uwierzytelniania przez 180 dni, bo administrator chciał tylko szybko zwolnić miejsce. Naprawa nie istnieje po fakcie — dane są nieodwracalnie usunięte. Zapobieganie: sprawdzenie polityki retencji **przed** ustawieniem limitu czasowego, nie po.

Skasowanie starych archiwów `.gz` bez sprawdzenia, co się w nich znajduje.

```
find /var/log -name '*.gz' -mtime +90 -delete
```

Uruchomione bez wcześniejszego przeglądu wyniku (bez `-delete`, samym `find`) na serwerze, gdzie akurat w tym katalogu leżały też archiwalne logi audytu wymagane do zachowania przez rok, nie 90 dni. Naprawa: zawsze najpierw `find /var/log -name '*.gz' -mtime +90` bez `-delete`, przejrzenie listy, dopiero potem dopisanie `-delete` do tego samego polecenia.

□□ Ćwiczenia

> Wszystko poniżej wykonujesz w katalogu `/tmp`, na własnoręcznie utworzonych plikach — żadne ćwiczenie nie czyści prawdziwego dziennika systemowego ani realnych logów aplikacji.

Ćwiczenie 1 — symulacja rozjazdu `rm` na otwartym pliku

```
cd /tmp
dd if=/dev/zero of=duzy.log bs=1M count=50 2>/dev/null
du -sh duzy.log

# w jednym terminalu: proces trzymający plik otwarty
tail -f duzy.log &
TAIL_PID=$!
```

```
# usunięcie pliku, mimo że tail go wciąż czyta
rm duzy.log
ls -la duzy.log 2>&1
lsof -p $TAIL_PID 2>/dev/null | grep duzy || echo "(lsof niedostępny albo proces już nie trzyma pliku)"

# sprzątanie
kill $TAIL_PID
```

Czy plik `duzy.log` istnieje jeszcze na liście `ls` po `rm`? Co by pokazało `df` na systemie, gdzie taki plik miałby kilkadziesiąt gigabajtów zamiast 50 megabajtów?

Odpowiedź

`ls` nie pokaże już pliku — wpis w katalogu został usunięty natychmiast. Ale dopóki proces `tail -f` (uruchomiony w tle) trzyma otwarty deskryptor do tego pliku, bloki dysku zajmowane przez jego zawartość nie są zwalniane — przy pliku wielkości kilkudziesięciu gigabajtów `df` pokazywałby dokładnie tyle samo zajętego miejsca co przed `rm`, mimo że `du` (licząc po wpisach katalogowych) już by go nie widział. Dopiero `kill $TAIL_PID` (zamknięcie ostatniego procesu trzymającego plik) faktycznie zwalnia przestrzeń.

Ćwiczenie 2 — porównanie `rm` i `truncate` na aktywnym zapisie

```
cd /tmp
: > log-truncate.log
( for i in $(seq 1 5); do echo "linia $i" >> log-truncate.log; sleep 0.2; done ) &
WRITER_PID=$!
sleep 0.5
truncate -s 0 log-truncate.log
wait $WRITER_PID
cat log-truncate.log
```

Ile linii zawiera plik na końcu, mimo że proces w tle próbował dopisać ich pięć? Dlaczego `truncate` nie przerwał procesu piszącego, w przeciwieństwie do tego, co zrobiłoby `rm` w połączeniu z próbą ponownego otwarcia pliku przez tę samą ścieżkę?

Odpowiedź

Plik będzie zawierał tylko linie dopisane **po** momencie `truncate` (zwykle 2–3 ostatnie z pięciu, w zależności od dokładnego czasu) — `truncate -s 0` wyzerował zawartość, ale proces piszący nadal ma otwarty ten sam deskryptor i kontynuuje zapis od nowej, zerowej pozycji, bez żadnego błędu. Gdyby zamiast tego użyto `rm` na pliku i coś próbowało go **ponownie otworzyć** pod tą samą nazwą (np. kolejna instancja procesu), dostałoby zupełnie nowy plik (nowy inode) — a stary proces, wciąż piszący do starego, usuniętego inode'u, pisałby donikąd, do przestrzeni, której nikt już nie odczyta.

Ćwiczenie 3 — bezpieczna procedura kasowania starych archiwów

```
cd /tmp
mkdir -p archiwum-testowe
for i in 10 45 95 200; do
  touch -d "-$i days" "archiwum-testowe/log-$i-dni.gz"
done
ls -la archiwum-testowe

# krok 1: tylko podejrzyj, co zostałoby skasowane
find archiwum-testowe -name '*.gz' -mtime +90

# krok 2: dopiero po weryfikacji - faktyczne usunięcie
find archiwum-testowe -name '*.gz' -mtime +90 -delete
ls -la archiwum-testowe
```

Które pliki zniknęły po drugim `find`, a które zostały? Dlaczego kolejność (najpierw podgląd, potem `-delete`) ma znaczenie praktyczne, a nie tylko kosmetyczne?

Odpowiedź

Powinny zniknąć pliki starsze niż 90 dni (`log-95-dni.gz` i `log-200-dni.gz`), a zostać te młodsze (`log-10-dni.gz`, `log-45-dni.gz`). Kolejność ma znaczenie, bo `-mtime +90` samo w sobie nie usuwa niczego — to `-delete` na końcu wykonuje operację. Uruchomienie najpierw samego `find` bez `-delete` daje możliwość sprawdzenia listy wzrokiem i wycofania się, zanim dojdzie do nieodwracalnego usunięcia — różnica między jedną literówką w wyrażeniu a utratą właściwych

plików.

Pytania kontrolne

1. `journalctl --vacuum-size=1G` zwraca, że zwolnił 3 GB, ale `df` pokazuje, że wolnego miejsca wcale nie przybyło. Co sprawdzić w pierwszej kolejności?

Odpowiedź

Czy przed czyszczeniem wykonano `journalctl --rotate` — jeśli aktywny plik dziennika wciąż jest otwarty do zapisu, `--vacuum-size` mógł ograniczyć się do plików archiwalnych i nie doliczyć aktywnego pliku, więc realny efekt bywa mniejszy niż oczekiwany. Warto też sprawdzić, czy miejsce nie jest zajęte przez coś zupełnie innego niż journal — np. przez inny, wciąż otwarty i usunięty plik, jak w pierwszym ćwiczeniu.

2. Czym różni się `SystemMaxUse` od `MaxRetentionSec` w `journald.conf`?

Odpowiedź

`SystemMaxUse` to ograniczenie przestrzenne — twardy limit rozmiaru, jaki dziennik może zająć na dysku, niezależnie od wieku wpisów. `MaxRetentionSec` to ograniczenie czasowe — maksymalny wiek pojedynczego wpisu, po którym zostanie usunięty niezależnie od tego, ile miejsca dziennik zajmuje. Journal egzekwuje oba naraz; wpis znika, gdy tylko jeden z limitów zostanie przekroczony.

3. Dlaczego `truncate -s 0` jest bezpieczniejsze niż `rm` przy czyszczeniu logu aktywnie zapisywanego przez działający proces?

Odpowiedź

`truncate` zeruje zawartość pliku, zachowując ten sam inode — proces piszący kontynuuje zapis do tego samego, wciąż otwartego deskryptora, bez przerwy w działaniu i bez ryzyka pisania do „donikąd". `rm` usuwa wpis katalogowy, ale sam plik (i zajmowane przez niego miejsce) przetrwa, dopóki proces trzyma go otwartym — a jeśli coś w międzyczasie spróbuje otworzyć plik o tej samej nazwie ponownie, dostanie zupełnie nowy inode, rozjeżdżając stan na dwa niezależne pliki.

4. Zespół prawny informuje, że logi uwierzytelniania muszą być zachowane przez 180 dni. Jak to pogodzić z ograniczonym miejscem na dysku, zamiast wybierać jedno albo

drugie?

Odpowiedź

Ustawić `MaxRetentionSec` na co najmniej 180 dni (żeby nie skasować niczego przedwcześnie) i osobno zadbać o fizyczne miejsce przez odpowiednio duży wolumin albo przez archiwizację starszych logów poza serwer produkcyjny (np. do zewnętrznego systemu logowania) zamiast ich kasowania. Retencja i miejsce na dysku to dwa osobne wymagania — rozwiązanie polega na zaplanowaniu wystarczającej przestrzeni albo eksportu, a nie na skróceniu retencji poniżej wymogu.

Powiązane: journalctl i zarządzanie dziennikiem systemowym · logrotate — rotacja i kompresja logów · Rozjazd df i du — usunięte pliki trzymane przez procesy · Audyt i ślady logowań: auditd, last, lastb, journalctl

Źródła: `man 1 journalctl` · `man 5 journald.conf` · `man 8 logrotate` · `man 8 systemd-tmpfiles` · `man 5 tmpfiles.d` · notatki własne (Przeczyszczenie journala systemowego)

Revision #15
Created 2026-09-18 06:57:29 UTC by Tomasz Konieczny
Updated 2026-10-03 19:10:10 UTC by Tomasz Konieczny