Co zajmuje miejsce: df, du, lsblk, lsscsi

Poziom: podstawy **Dotyczy: wszystkie dystrybucje** **Czas: ~20 min**

`df`, `du`, `lsblk` i `lsscsi` to cztery różne warstwy tego samego pytania „co się dzieje z dyskiem": ile miejsca zostało, co je zajmuje, jakie urządzenia w ogóle istnieją i co do nich fizycznie podłączono. Gdy dysk się zapełni, serwis przestaje działać — stąd te cztery komendy są pierwszym odruchem, nie ostatnią deską ratunku.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
GB	Gigabyte: miliard bajtów, jednostka pojemności pamięci i plików, tu najczęściej rozmiar modelu albo pamięci karty graficznej
LVM	Logical Volume Manager: warstwa w Linuksie składająca dyski w elastyczne wolumeny
NFS	Network File System: udostępnianie katalogów po sieci w świecie Linuksa
PHP	PHP: Hypertext Preprocessor, język programowania, w którym napisany jest frontend Zabbixa i wiele innych aplikacji webowych
SCSI	Small Computer System Interface: standard podłączania dysków
UUID	Universally Unique Identifier: długi, praktycznie niepowtarzalny identyfikator
XFS	system plików stworzony przez Silicon Graphics, domyślny w Rocky Linux i RHEL

Fiszka

TYLKO ODCZYT

Sprawdzenie zajętości dysku: -h

```
df -h
```

Wyświetlamy użycie wszystkich zamontowanych systemów plików w formacie czytelnym dla człowieka (`-h` = *human-readable*): zamiast liczby `15360000` widzimy `15G`. To pierwsza komenda przy każdym alarmie o zapełnionym dysku.

Przykład:

```
Filesystem                 Size  Used Avail Use% Mounted on
/dev/mapper/vgOS-lvOS      20G   14G  5.5G  72% /
/dev/mapper/vgVAR-lvVAR       30G   28G  1.2G  96% /var
/dev/sda1                  1014M  312M  702M  31% /boot
tmpfs                      3.9G     0  3.9G   0% /dev/shm
```

Bez `-h` wynik jest w blokach 1 KB i dla większych dysków praktycznie nieczytelny na pierwszy rzut oka.

df = disk free: ile miejsca wolnego na dyskach

TYLKO ODCZYT

Jednostki dziesiętne (SI): -H

```
df -H
```

To samo co `-h`, ale liczymy w potęgach 1000 zamiast 1024 — czyli tak, jak producenci dysków opisują ich pojemność na pudełku. Dysk „500 GB" w `df -H` pokaże bliżej 500G, a w zwykłym `-h` bliżej 465G. Różnica bywa źródłem pytań typu „gdzie zniknęło 35 GB z nowego dysku" — nigdzie, to tylko inna definicja gigabajta.

GB = Gigabyte: miliard bajtów, jednostka pojemności pamięci i plików, tu najczęściej rozmiar modelu albo pamięci karty graficznej · **df** = disk free: ile miejsca wolnego na dyskach

TYLKO ODCZYT

Wyświetlenie typu systemu plików: -T

```
df -T
```

Dokładamy kolumnę z typem systemu plików (`ext4`, `xfs`, `btrfs`, `nfs4`...). Przydaje się, gdy serwer ma zamontowane różne typy i trzeba wiedzieć, którym poleceniem administracyjnym się nimi zajmować — `resize2fs` działa tylko na ext, `xfs_growfs` tylko na XFS.

Przykład:

```
Filesystem                  Type     Size  Used Avail Use% Mounted on
/dev/mapper/vgOS-lvOS       xfs       20G   14G  5.5G  72% /
/dev/mapper/vgVAR-lvVAR       xfs       30G   28G  1.2G  96% /var
fs01.zielona.example:/export  nfs4     500G  410G   90G  83% /mnt/nfs
```

Praktyczne połączenie obu flag: `df -Th` daje jednocześnie typ i czytelny rozmiar — to najczęstsza forma używana przez administratorów na co dzień.

XFS = system plików stworzony przez Silicon Graphics, domyślny w Rocky Linux i RHEL · **df** = disk free: ile miejsca wolnego na dyskach

TYLKO ODCZYT

Zawężenie do jednego punktu montowania: -h

```
df -h /var/log
```

Zamiast przeglądać całą listę, pytamy od razu o system plików, na którym leży dana ścieżka. `df` sam znajduje najbliższy punkt montowania nadrzędny wobec podanej ścieżki — nie trzeba wiedzieć z góry, że `/var/log` leży akurat na osobnym woluminie `/var`.

Przykład:

```
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/vgVAR-lvVAR    30G   28G  1.2G  96% /var
```

df = disk free: ile miejsca wolnego na dyskach

TYLKO ODCZYT

Wybór własnych kolumn: --output

```
df --output=source,fstype,pcent,target -x tmpfs
```

Zamiast sztywnego formatu, wybieramy dokładnie te pola, które chcemy zobaczyć (`source`, `size`, `used`, `avail`, `pcent`, `target`, `fstype`, `itotal`, `iused`...). Flaga `-x tmpfs` dodatkowo wyklucza z wyniku systemy plików danego typu — wygodne, żeby nie zaśmiecać wyniku tymczasowymi `tmpfs`/`devtmpfs`.

Przykład:

```
Filesystem                   Type  Use% Mounted on
/dev/mapper/vgOS-lvOS       xfs    72% /
/dev/mapper/vgVAR-lvVAR        xfs    96% /var
/dev/sda1                     xfs    31% /boot
```

df = disk free: ile miejsca wolnego na dyskach

TYLKO ODCZYT

Filtrowanie i sortowanie wyniku `df`

```
df -h | grep -v tmpfs
df -hT | sort -k6 -rn
```

Dwa odruchy zamiast czytania całego wyniku wzrokiem: pierwszy chowa tymczasowe systemy plików (`tmpfs`, `devtmpfs`), rzadko istotne przy diagnozie; drugi sortuje po procencie zajęcia malejąco (kolumna 6 przy `-T`), więc najbardziej problematyczny wolumen ląduje na górze. Podobnie działa `df -h | awk '$5+0 > 80 {print}'` — pokazuje tylko dyski zapełnione powyżej 80%.

Przykład:

```
Filesystem                  Type  Size  Used Avail Use% Mounted on
/dev/mapper/vgVAR-lvVAR       xfs    30G   28G  1.2G  96% /var
/dev/mapper/vgOS-lvOS      xfs    20G   14G  5.5G  72% /
```

df = disk free: ile miejsca wolnego na dyskach

TYLKO ODCZYT

Sprawdzenie zajętości inode'ów: -i

```
df -i
```

Ta sama tabela co `df -h`, ale liczymy nie bajty, tylko wolne i zajęte i-węzły (inode'y). Można mieć wolne miejsce w `df -h` i jednocześnie zero wolnych inode'ów w `df -i` — system wtedy odmówi tworzenia nowych plików, mimo że miejsca na dysku jest mnóstwo.

Przykład:

```
Filesystem              Inodes  IUsed    IFree IUse% Mounted on
/dev/mapper/vgOS-lvOS  1310720 842113   468607   65% /
/dev/mapper/vgVAR-lvVAR   1966080 1965998      82  100% /var
```

Powyższy przykład to dokładnie ten alarmowy stan: `/var` ma `IUse% = 100%` przy niepełnym `df -h`.

df = disk free: ile miejsca wolnego na dyskach

TYLKO ODCZYT

Sprawdzenie rozmiaru katalogu: -sh

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

Wypisujemy podsumowanie (`-s`, *summary*) rozmiaru całego katalogu w jednej liczbie, w formacie czytelnym. To pierwsza komenda, gdy `df` powiedział „pełne", a trzeba znaleźć, który katalog za to odpowiada.

Przykład:

```
14G     /var/log
```

du = disk usage: ile miejsca zajmuje katalog

TYLKO ODCZYT

Podsumowanie każdego elementu katalogu: -sh *

```
cd /var && du -sh *
```

Bez `-s` `du` schodzi rekursywnie do każdego podkatalogu i wypisze tysiące linii. Gwiazdka rozwijana przez powłokę sprawia, że `du` liczy osobne podsumowanie dla każdego pliku i katalogu na najwyższym poziomie `/var`, zamiast dla `/var` jako całości.

Przykład:

```
4.0K    account
1.2M    cache
14G     log
28K     opt
```

```
9.4G    lib
16K     spool
```

cd = change directory: przejdź do katalogu

TYLKO ODCZYT

Ograniczenie głębokości liczenia: --max-depth=1

```
du -h --max-depth=1 /var
```

To samo zadanie co `du -sh *`, ale przez oficjalną flagę zamiast rozwijania gwiazdki przez powłokę — działa identycznie niezależnie od tego, ile jest plików w katalogu i czy któryś zaczyna się od kropki. Krótszy zapis tej samej flagi to `du -h -d 1`.

Przykład:

```
4.0K    /var/account
1.2M    /var/cache
14G     /var/log
9.4G    /var/lib
16K     /var/spool
14G     /var
```

Ostatnia linia to zawsze podsumowanie samego katalogu wskazanego jako argument — doliczamy ją przy porównywaniu z `du -sh`.

du = disk usage: ile miejsca zajmuje katalog

TYLKO ODCZYT

Ranking największych plików i katalogów: -a | sort -rh

```
du -ah /var | sort -rh | head -20
```

`-a` (*all*) każe `du` wypisać rozmiar **każdego** pliku, nie tylko katalogów. `sort -rh` sortuje po rozmiarze w formacie czytelnym dla człowieka malejąco (`-h` w `sort` rozumie `K`, `M`, `G` tak jak w `du -h`). Efekt to gotowa lista podejrzanych: 20 największych plików i podkatalogów w jednym poleceniu.

Przykład:

```
9.1G    /var/lib/mysql/ibdata1
4.2G    /var/log/app/app.log
2.8G    /var/log/app/app.log.1
1.1G    /var/lib/docker/overlay2
```

du = disk usage: ile miejsca zajmuje katalog

TYLKO ODCZYT

Zatrzymanie liczenia na granicy systemu plików: -x

```
du -sh -x /
```

`-x` każe `du` zostać na tym samym systemie plików, na którym leży punkt startowy, i nie schodzić do zamontowanych pod spodem innych woluminów, sieciowych zasobów NFS ani `/proc`, `/sys`. Bez tej flagi `du -sh /` potrafi wisieć minutami, licząc zawartość wolnego zasobu sieciowego zamontowanego gdzieś głęboko w drzewie katalogów, i zwraca liczbę, która myli rozmiar lokalnego dysku z rozmiarem wszystkiego, co pod niego podpięto.

Przykład:

```
14G     /
```

NFS = Network File System: udostępnianie katalogów po sieci w świecie Linuksa · **du** = disk usage: ile miejsca zajmuje katalog

TYLKO ODCZYT

Pominięcie wzorca przy liczeniu: --exclude

```
du -sh --exclude='*.log' /var/lib/app
```

Wykluczamy z liczenia pliki pasujące do podanego wzorca (glob, nie wyrażenie regularne). Przydaje się przy katalogach, w których część zawartości i tak zostanie usunięta albo jest liczona osobno — np. gdy chcemy poznać rozmiar danych aplikacji bez jej własnych logów.

du = disk usage: ile miejsca zajmuje katalog

TYLKO ODCZYT

Szukanie dużych plików bezpośrednio przez `find`

```
find /var/log -type f -size +500M -exec ls -lh {} \;
```

Gdy już wiadomo, w którym katalogu szukać, `find` z `-size` jest szybszy niż przeglądanie wyniku `du -ah | sort`, bo od razu filtruje po progu rozmiaru. Znak `+` przed liczbą znaczy „większe niż"; `-100M` znaczyłoby „mniejsze niż".

Przykład:

```
-rw-r--r-- 1 app app 1.2G lis 20 03:11 /var/log/app/app.log.3
```

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

TYLKO ODCZYT

Wypisanie drzewa urządzeń blokowych

```
lsblk
```

Pokazujemy wszystkie urządzenia blokowe w systemie — dyski, partycje, wolumeny LVM — w formie drzewa, razem z punktami montowania. Pierwsza komenda, gdy trzeba szybko zobaczyć „co jest podpięte i gdzie", zanim w ogóle sięgniemy po `df`.

Przykład:

```
NAME              MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                 8:0    0   40G  0 disk
├─sda1              8:1    0    1G  0 part /boot
└─sda2              8:2    0   39G  0 part
  ├─vgOS-lvOS   253:0    0   20G  0 lvm  /
  └─vgVAR-lvVAR    253:1    0   19G  0 lvm  /var
sdb                 8:16   0  100G  0 disk
└─sdb1              8:17   0  100G  0 part
  └─vgDB-lvOracle   253:2    0  100G  0 lvm  /var/lib/mysql
```

Wcięcia i znaki `├─`/`└─` pokazują zależności: partycja leży na dysku, wolumen logiczny leży na partycji oznaczonej jako fizyczny wolumen LVM. Pełne wyjaśnienie tej warstwy — na stronie o modelu LVM.

LVM = Logical Volume Manager: warstwa w Linuksie składająca dyski w elastyczne wolumeny · **lsblk** = list block devices: pokaż dyski i partycje

TYLKO ODCZYT

Wyświetlenie systemów plików, UUID i etykiet: -f

```
lsblk -f
```

Ta sama struktura co zwykłe `lsblk`, dodatkowo z typem systemu plików, jego etykietą (`LABEL`) i identyfikatorem `UUID`, którego używa `/etc/fstab` do trwałego wskazywania wolumenu — trwalszego niż nazwa `/dev/sdb1`, która potrafi się zmienić po restarcie z innym zestawem dysków.

Przykład:

```
NAME            FSTYPE FSVER LABEL UUID                                   MOUNTPOINT
sda
├─sda1          xfs                a1b2c3d4-e5f6-7890-abcd-ef1234567890 /boot
└─sda2          LVM2_member
  ├─vgOS-lvOS xfs                11112222-3333-4444-5555-666677778888 /
  └─vgVAR-lvVAR  xfs                  99998888-7777-6666-5555-444433332222 /var
```

UUID = Universally Unique Identifier: długi, praktycznie niepowtarzalny identyfikator · **lsblk** = list block devices: pokaż dyski i partycje

TYLKO ODCZYT

Zawężenie do urządzeń SCSI: -S

```
lsblk -S
```

Zawężamy wynik do samych dysków SCSI/SATA/NVMe (bez partycji i wolumenów), razem z modelem, rozmiarem i typem interfejsu (`sata`, `sas`, `nvme`). Odpowiednik `lsscsi` napisany jako flaga `lsblk`, przydatny na maszynach, gdzie pakietu `lsscsi` nie ma zainstalowanego.

Przykład:

```
NAME HCTL       TYPE VENDOR   MODEL             REV TRAN
sda  2:0:0:0    disk VMware   Virtual disk     2.0  spi
sdb  2:0:1:0    disk VMware   Virtual disk     2.0  spi
```

SCSI = Small Computer System Interface: standard podłączania dysków · **lsblk** = list block devices: pokaż dyski i partycje

TYLKO ODCZYT

Wybór własnych kolumn: -o

```
lsblk -o NAME,SIZE,FSTYPE,MOUNTPOINT
```

Podobnie jak przy `df --output`, wybieramy dokładnie te pola, które nas interesują, zamiast domyślnego zestawu. `lsblk -p` (osobna flaga) dokłada pełne ścieżki urządzeń (`/dev/sda` zamiast samego `sda`) — przydatne, gdy wynik ma trafić bezpośrednio do innej komendy jako argument.

Przykład:

```
NAME            SIZE FSTYPE      MOUNTPOINT
sda               40G
├─sda1             1G xfs         /boot
└─sda2            39G LVM2_member
  ├─vgOS-lvOS     20G xfs         /
  └─vgVAR-lvVAR     19G xfs         /var
```

lsblk = list block devices: pokaż dyski i partycje

TYLKO ODCZYT

Wypisanie urządzeń SCSI/SATA

```
lsscsi
```

Osobny pakiet (`lsscsi`), który pokazuje podłączone urządzenia SCSI/SATA płasko, bez drzewa partycji — sam kontroler, numer HCTL i węzeł urządzenia. Używamy go głównie do weryfikacji, że wszystkie dyski fizyczne, które maszyna wirtualna albo macierz powinna widzieć, faktycznie są widoczne w systemie po stronie jądra.

Przykład:

```
[2:0:0:0]    disk    VMware   Virtual disk     2.0   /dev/sda
[2:0:1:0]    disk    VMware   Virtual disk     2.0   /dev/sdb
```

SCSI = Small Computer System Interface: standard podłączania dysków

TYLKO ODCZYT

Wyświetlenie rozmiaru dysków: -s

```
lsscsi -s
```

To samo, z dołożoną kolumną rozmiaru — szybkie porównanie „ile GB widać z poziomu systemu" z tym, co zostało przydzielone po stronie hypervisora albo macierzy dyskowej.

Przykład:

```
[2:0:0:0]    disk    VMware   Virtual disk     2.0   /dev/sda   42.9GB
[2:0:1:0]    disk    VMware   Virtual disk     2.0   /dev/sdb  107GB
```

GB = Gigabyte: miliard bajtów, jednostka pojemności pamięci i plików, tu najczęściej rozmiar modelu albo pamięci karty graficznej

TYLKO ODCZYT

Wypisanie identyfikatorów systemów plików

```
blkid
```

Wypisujemy `UUID`, typ systemu plików i etykietę dla każdego urządzenia blokowego z systemem plików — bez struktury drzewa, po prostu listę. Ten sam `UUID`, który pokazuje `lsblk -f`, tylko w formacie gotowym do wklejenia do `/etc/fstab`.

Przykład:

```
/dev/sda1: UUID="a1b2c3d4-e5f6-7890-abcd-ef1234567890" TYPE="xfs" PARTUUID="..."
/dev/mapper/vgOS-lvOS: UUID="11112222-3333-4444-5555-666677778888" TYPE="xfs"
/dev/mapper/vgVAR-lvVAR: UUID="99998888-7777-6666-5555-444433332222" TYPE="xfs"
```

UUID = Universally Unique Identifier: długi, praktycznie niepowtarzalny identyfikator · **blkid** = pokaż UUID i typ systemu plików na urządzeniu blokowym

TYLKO ODCZYT

Wyświetlenie punktów montowania jako drzewa

```
findmnt
```

Pokazujemy wszystko, co jest aktualnie zamontowane, razem z opcjami montowania, w postaci drzewa odzwierciedlającego rzeczywistą hierarchię punktów montowania — nie

kolejność w `/etc/fstab`, tylko stan faktyczny, odczytany z jądra. `findmnt /var` pyta o jeden konkretny punkt i jego opcje.

Przykład:

```
TARGET              SOURCE                    FSTYPE OPTIONS
/                   /dev/mapper/vgOS-lvOS   xfs    rw,relatime
├─/boot             /dev/sda1                 xfs    rw,relatime
└─/var              /dev/mapper/vgVAR-lvVAR    xfs    rw,relatime
```

findmnt = pokaż zamontowane systemy plików w czytelnej formie drzewa

TYLKO ODCZYT

Interaktywne przeglądanie zajętości: ncdu

```
ncdu /var
```

Nakładka na `du` z interfejsem tekstowym: liczy rozmiary raz, a potem pozwala klikać (strzałkami) w kolejne podkatalogi bez ponownego przeliczania całości od zera, tak jak trzeba by robić przy powtarzanym `du -h --max-depth=1`. Wymaga osobnego pakietu (`ncdu`), nie jest częścią `coreutils`.

Przykład:

```
--- /var ---------------------------------
   14.0 GiB [##########] /log
    9.4 GiB [######    ] /lib
    1.2 MiB [          ] /cache
```

TYLKO ODCZYT

Wszystkie flagi i narzędzia

> **Pojęcie: narzędzia inspekcji przechowywania.** `df`, `du`, `lsblk`, `lsscsi`, `blkid` i `findmnt` to nie warianty jednej komendy, tylko sześć osobnych programów patrzących na tę samą rzeczywistość z różnych warstw: `df` i `du` mierzą zajętość (systemu plików albo plików), `lsblk`, `lsscsi` i `blkid` opisują same urządzenia i ich identyfikatory, a `findmnt` pokazuje, co i gdzie jest w danej chwili zamontowane. Poniższa tabela zbiera

je do przeszukiwania wzrokiem.

Komenda	Warstwa	Co pokazuje
`df -h` / `df -T` / `df -i`	system plików	zajętość miejsca, typ, inode'y
`du -sh` / `du -ah | sort`	pliki i katalogi	
`lsblk` / `lsblk -f` / `lsblk -S`	urządzenia blokowe	drzewo dysk → partycja → LVM → punkt montowania
`lsscsi`	kontroler SCSI/SATA	fizyczne (albo wirtualne) dyski widziane przez jądro
`blkid`	identyfikatory	UUID i typ systemu plików do `/etc/fstab`
`findmnt`	stan montowania	co jest zamontowane teraz, z jakimi opcjami
`ncdu`	pliki i katalogi	interaktywny `du` bez ponownego liczenia

coś aktywnie zajmuje przestrzeń

Uważaj

- `df` i `du` **nie muszą się zgadzać** — pierwsza czyta metadane systemu plików, druga liczy pliki na dysku; różnica i jej najczęstsza przyczyna to temat następnej strony.
- `du` bez `-x` na katalogu, pod którym coś jest zamontowane (NFS, inny dysk, `/proc`), potrafi liczyć minutami i zwrócić mylącą sumę.
- Wolne miejsce w `df -h` przy pełnym `df -i` kończy się komunikatem `No space left on device` przy próbie utworzenia pliku — mimo realnie wolnych gigabajtów.

LVM = Logical Volume Manager: warstwa w Linuksie składająca dyski w elastyczne wolumeny · **SCSI** = Small Computer System Interface: standard podłączania dysków · **NFS** = Network File System: udostępnianie katalogów po sieci w świecie Linuksa

Z życia wzięte

Z ŻYCIA WZIĘTE

Wtorek, 9:15. Tomek dostaje zgłoszenie, że serwer aplikacji app01 nie ma miejsca na dysku, ale sam nie wie, od czego zacząć sprawdzanie. Uruchamia po kolei df, du i lsblk, żeby zobaczyć to samo z trzech różnych stron: ile miejsca widzi system plików, co faktycznie zajmują pliki i jak wygląda sam dysk pod spodem. Dopiero zestawienie trzech wyników pokazuje, że katalog logów aplikacji urósł w nocy do dwudziestu gigabajtów.

Po co to istnieje

Pojęcie: system plików kontra urządzenie blokowe. Urządzenie blokowe to fizyczny albo wirtualny nośnik danych widziany przez jądro jako ciąg ponumerowanych bloków — dysk, partycja, wolumen LVM (`/dev/sda`, `/dev/sda1`, `/dev/mapper/vgOS-lvOS`). Sam w sobie nie ma pojęcia „pliku" ani „katalogu" — to tylko miejsce, w którym można zapisywać i odczytywać bloki bajtów pod numerem. System plików to struktura danych zapisana na takim urządzeniu, która nadaje temu ciągowi bloków znaczenie: dzieli je na pliki, katalogi, przechowuje nazwy, uprawnienia, znaczniki czasu i mapę, który blok należy do którego pliku. `mkfs.ext4 /dev/sdb1` bierze urządzenie blokowe i zapisuje na nim system plików; `mount` mówi jądru „od teraz to, co jest pod tą ścieżką w drzewie katalogów, tak naprawdę leży na tym urządzeniu, sformatowanym tym systemem plików". `lsblk` pokazuje pierwszą warstwę — urządzenia; `df` pokazuje drugą — systemy plików na nich zapisane.

Dysk, który się zapełnia, jest jednym z najstarszych i najbardziej przewidywalnych problemów administracji systemami — i jednym z najgroźniejszych, bo objawy potrafią być odległe od przyczyny: baza danych przestaje przyjmować zapisy, usługa się zawiesza przy próbie zapisania loga, `sshd` odmawia nowego połączenia, bo nie może dopisać wpisu do `/var/log/wtmp`. `df`, `du`, `lsblk` i `lsscsi` to komplet narzędzi do przejścia od objawu do przyczyny: `df` mówi, który system plików jest pełny, `du` mówi, co dokładnie go zapełnia, a `lsblk`/`lsscsi` pokazują, czy w ogóle jest fizyczna możliwość dołożenia miejsca.

Analogia

Wyobraź sobie **duży magazyn z recepcją przy wjeździe**.

Przy wjeździe stoi tablica, która pokazuje zajętość każdej hali: „Hala A: 72% zajęta, Hala B: 96% zajęta". Ta tablica to `df` — nie chodzi po halach, tylko odczytuje zbiorczy licznik, który magazyn utrzymuje sam dla siebie: ile palet zajętych, ile wolnych. Odczyt jest natychmiastowy, bo to jedna liczba na halę, aktualizowana przy każdym przyjęciu i wydaniu towaru.

`du` to magazynier, który fizycznie idzie wzdłuż regałów w hali i sumuje objętość każdej paczki, każdego pudła, każdej palety — piętro po piętrze, regał po regale. Dostaje dokładnie tę samą odpowiedź co tablica przy wjeździe, ale musi obejść cały teren, żeby ją policzyć, więc w wielkiej hali zajmuje mu to zauważalnie dłużej niż spojrzenie na tablicę.

`lsblk` i `lsscsi` to zupełnie inna warstwa — to inwentaryzacja samych hal i doków rozładunkowych: ile w ogóle jest budynków, które doki są podłączone do której hali, ile miejsca fizycznie da się tam jeszcze zbudować. Możesz mieć halę zapełnioną w 100% (problem `df`/`du`) i jednocześnie wolny plac obok, na którym da się postawić kolejną (temat strony o powiększaniu przestrzeni).

Analogia ma jedną granicę: tablica przy wjeździe (`df`) i magazynier liczący paczki (`du`) w prawdziwym magazynie zawsze zgadzają się co do liczby — w Linuksie, jak zobaczysz na następnej stronie, potrafią się rozjechać, bo tablica czasem liczy paczkę, której magazynier już fizycznie nie widzi.

Jak to działa

Rozwiń: jak to działa pod spodem

> **Pojęcie: punkt montowania.** Punkt montowania to katalog w istniejącym drzewie systemu plików, pod którym jądro „podpina" zawartość innego systemu plików z innego

urządzenia. Zanim wykonasz `mount /dev/sdb1 /var`, katalog `/var` jest zwykłym, pustym (albo niepustym — wtedy jego dotychczasowa zawartość staje się chwilowo niewidoczna) katalogiem na tym samym urządzeniu co `/`. Po `mount` wszystko, co zapiszesz do `/var/cokolwiek`, faktycznie trafia na `/dev/sdb1`, a nie na dysk, na którym leży `/`. `df` pokazuje właśnie tę granicę w kolumnie `Mounted on` — każdy wiersz to jeden punkt montowania, czyli jeden osobny licznik zajętości.



Każde narzędzie diagnostyczne widzi inną warstwę tej samej drogi od dysku do plików w katalogu. (kliknięcie otwiera pełny rozmiar)

`df` działa tanio, bo nie czyta zawartości plików — pyta jądro (przez wywołanie systemowe `statfs()`) o metadane systemu plików: całkowitą liczbę bloków, liczbę wolnych bloków, rozmiar bloku. Każdy system plików utrzymuje te liczniki sam, aktualizując je przy każdym zapisie i usunięciu, więc odczyt jest natychmiastowy niezależnie od tego, ile plików leży na dysku.

`du` działa drogo, bo robi dokładnie to, czego `df` unika: rekurencyjnie wchodzi do każdego katalogu, odpytuje każdy plik o jego rzeczywisty rozmiar na dysku (nie logiczny rozmiar w bajtach, tylko liczbę faktycznie zaalokowanych bloków — stąd małe różnice przy plikach rzadkich, *sparse files*) i sumuje. Na katalogu z milionami małych plików to potrafi trwać minuty, bo koszt rośnie z liczbą plików, nie z ich łącznym rozmiarem.

Pojęcie: i-węzeł (inode) i wyczerpanie inode'ów. I-węzeł to struktura danych, którą system plików tworzy dla każdego pliku i katalogu — przechowuje właściciela, uprawnienia, znaczniki czasu, rozmiar i wskaźniki do bloków danych, ale **nie** nazwę pliku (ta leży osobno, w katalogu, jako para nazwa→numer inode'u). Liczba dostępnych inode'ów jest ustalana raz, przy tworzeniu systemu plików (`mkfs`), i na klasycznym ext4 nie da się jej później swobodnie zwiększyć bez przebudowy systemu plików od zera. Jeśli katalog zawiera miliony malutkich plików — sesje PHP, cache miniatur, kolejki wiadomości — możliwe jest wyczerpanie puli inode'ów, mimo że fizycznego miejsca na dysku zostało mnóstwo, bo każdy, nawet zerobajtowy, plik zużywa jeden inode. Wtedy próba utworzenia kolejnego pliku kończy się komunikatem `No space left on device` — dokładnie tym samym, co przy braku miejsca na bloki — i jedynym sposobem, żeby to odróżnić, jest `df -i` obok `df -h`.

XFS radzi sobie z tym inaczej niż ext4: alokuje inode'y dynamicznie w miarę potrzeby (do limitu zależnego od rozmiaru systemu plików), więc wyczerpanie inode'ów na typowej instalacji XFS

zdarza się rzadziej, ale wciąż jest teoretycznie możliwe przy naprawdę ekstremalnej liczbie małych plików.

Rezerwa 5% dla roota. Systemy plików ext2/ext3/ext4 domyślnie rezerwują 5% swojej pojemności wyłącznie dla procesów działających jako root — żeby przy zapełnieniu dysku do 100% administrator wciąż mógł się zalogować i posprzątać, zamiast zostać odcięty od własnego systemu. Ta rezerwa tłumaczy, dlaczego zwykły użytkownik dostaje `No space left on device` przy pliku o 500 MB, mimo że `df` pokazuje więcej wolnego miejsca — te ostatnie procenty są zarezerwowane, nie dla niego. Rozmiar rezerwy sprawdzisz i zmienisz przez `tune2fs`:

```
tune2fs -l /dev/mapper/vgVAR-lvVAR | grep -i reserved
tune2fs -m 1 /dev/mapper/vgVAR-lvVAR
```

Pierwsza komenda pokazuje aktualny procent i liczbę zarezerwowanych bloków, druga obniża rezerwę do 1% — częsta zmiana na dużych wolumenach danych (np. `/var/lib/mysql`), gdzie 5% z kilku terabajtów to realnie dziesiątki gigabajtów, których root i tak zwykle nie potrzebuje. XFS nie ma tego mechanizmu w tej formie.

Krok po kroku

1. Alarm: dysk pełny. Sprawdź, który system plików.

```
df -h
```

```
Filesystem                 Size  Used Avail Use% Mounted on
/dev/mapper/vgOS-lvOS      20G   14G  5.5G  72% /
/dev/mapper/vgVAR-lvVAR       30G   29G  512M  98% /var
```

`/var` jest niemal pełne. Reszta diagnozy skupia się właśnie na nim.

2. Zejdź na najwyższy poziom katalogów wewnątrz.

```
du -h --max-depth=1 /var 2>/dev/null | sort -rh
```

```
14G     /var/log
9.4G    /var/lib
2.1G    /var/cache
28K     /var/opt
16K     /var/spool
```

```
26G     /var
```

`/var/log` i `/var/lib` razem odpowiadają za większość zajętości.

3. Zagłębiaj się w największego podejrzanego.

```
du -h --max-depth=1 /var/log 2>/dev/null | sort -rh
```

```
12G     /var/log/app
1.8G    /var/log/audit
14G     /var/log
```

4. Znajdź konkretne pliki, nie tylko katalogi.

```
find /var/log/app -type f -size +500M -exec ls -lh {} \;
```

```
-rw-r--r-- 1 app app 8.1G lis 20 03:11 /var/log/app/app.log
```

Jeden nierotowany plik loga odpowiada za większość zajętości `/var`. Dalsza naprawa (kompresja, rotacja, sprawdzenie, czy proces trzyma go otwartym mimo usunięcia) — na kolejnej stronie.

5. Sprawdź od razu, czy to nie problem inode'ów.

```
df -i /var
```

```
Filesystem              Inodes  IUsed   IFree IUse% Mounted on
/dev/mapper/vgVAR-lvVAR  1966080 1200341  765739   61%   /var
```

Tutaj inode'y są w normie — problem to rzeczywiście miejsce, nie ich liczba. Gdyby `IUse%` było bliskie 100% przy sporym `Avail` w `df -h`, szukałbyś zamiast tego katalogu z gigantyczną liczbą małych plików:

```
find /var -xdev -printf '%h\n' | sort | uniq -c | sort -rn | head -10
```

```
842013 /var/lib/app/sessions
   412 /var/log/app
```

6. Zweryfikuj fizyczny layout, zanim zaczniesz dokładać miejsce.

```
lsblk -f
```

```
NAME            FSTYPE LABEL UUID                                 MOUNTPOINT
sda
├─sda1          xfs                a1b2...                        /boot
└─sda2          LVM2_member
  ├─vgOS-lvOS xfs                1111...                       /
  └─vgVAR-lvVAR  xfs                9999...                        /var
```

`lvVAR` leży na jednej partycji jednego dysku — jeśli trzeba będzie dołożyć przestrzeń, wystarczy powiększyć ten dysk albo dodać nowy do tej samej grupy wolumenów (strony o LVM i o powiększaniu przestrzeni).

Częste błędy

`du -sh /` wisi kilka minut albo liczy dziwną sumę.

```
du -sh /
```

Bez `-x` `du` schodzi też do zamontowanych zasobów sieciowych, obrazów kontenerów w `/var/lib/docker`, a nawet próbuje wejść do pseudo-systemów plików `/proc` i `/sys`. Naprawa:

```
du -sh -x /
```

`-x` zatrzymuje `du` na granicy systemu plików, z którego wystartował.

Porównywanie `du -sh` z `df -h` i oczekiwanie identycznej liczby.

```
du -sh /var
df -h /var
```

```
26G     /var
Filesystem                  Size  Used Avail Use% Mounted on
/dev/mapper/vgVAR-lvVAR        30G   29G  512M  98% /var
```

`du` policzył 26G plików, `df` mówi o 29G zajętych bloków. To nie błąd pomiaru — różnica bierze się z metadanych systemu plików, zarezerwowanych bloków i (najczęściej) z usuniętych, ale wciąż otwartych plików, których `du` w ogóle nie widzi. Pełne wyjaśnienie tego zjawiska — następna strona.

Odczytanie `df -h` bez sprawdzenia `df -i` przy `No space left on device`.

```
touch /var/lib/app/sessions/nowy_plik
```

```
touch: cannot touch '/var/lib/app/sessions/nowy_plik': No space left on device
```

`df -h /var` w tym momencie pokazuje tylko 63% zajętości — komunikat jest identyczny jak przy braku bloków, ale przyczyna jest inna:

```
df -i /var
```

```
Filesystem              Inodes  IUsed   IFree IUse% Mounted on
/dev/mapper/vgVAR-lvVAR  1966080 1965998      82  100%  /var
```

Wolnych inode'ów zostało 82. Naprawa to usunięcie masy drobnych plików (np. przeterminowanych sesji), nie zwalnianie miejsca w GB.

Zaufanie nazwie `/dev/sdb`, że zawsze wskazuje ten sam dysk.

```
echo '/dev/sdb1 /data xfs defaults 0 0' >> /etc/fstab
```

Nazwy `/dev/sdX` są przydzielane w kolejności wykrycia przez jądro przy starcie i po restarcie z inną liczbą dysków (albo po dodaniu nowego LUN-u z macierzy) mogą się zamienić miejscami. Wpis we `/etc/fstab` oparty na nazwie `sdb` czasem po restarcie montuje zupełnie inny dysk niż zamierzony. Bezpieczny wpis korzysta z UUID, który zobaczysz w `lsblk -f` albo `blkid`:

```
UUID=99998888-7777-6666-5555-444433332222  /data  xfs  defaults  0 0
```

□□ Ćwiczenia

> Wszystko poniżej wykonujesz w `/tmp`, na plikach obrazów systemu plików utworzonych na własną rękę — nic nie dotyka realnych dysków systemu. Punkty oznaczone `sudo` wymagają uprawnień administratora tylko do zamontowania obrazu; sam obraz i tak leży w Twoim `/tmp`.

Ćwiczenie 1 — df kontra du na tym samym katalogu

```
cd /tmp
mkdir -p cwiczenie1 && cd cwiczenie1
for i in $(seq 1 5); do dd if=/dev/urandom of=plik$i.bin bs=1M count=10 2>/dev/null; done
```

```
du -sh .
df -h .
```

`du -sh` pokaże ok. 50M. Dlaczego `df -h .` (system plików, na którym leży `/tmp`) pokazuje zupełnie inną, dużo większą liczbę w kolumnie `Used`?

Odpowiedź

Bo `df -h .` nie liczy Twojego katalogu — pokazuje zajętość **całego** systemu plików, na którym leży `/tmp` (zwykle `/` albo osobny wolumen), czyli sumę wszystkiego, co ktokolwiek na nim zapisał, nie tylko Twoich pięciu plików. Żeby zawęzić `df` do konkretnego katalogu, i tak dostaniesz odpowiedź dla całego wolumenu — `df` nie umie policzyć samego podkatalogu, do tego zawsze służy `du`.

Ćwiczenie 2 — wyczerpanie inode'ów bez ruszania prawdziwego dysku

```
cd /tmp
dd if=/dev/zero of=male-inody.img bs=1M count=16
mkfs.ext4 -N 50 male-inody.img
mkdir -p mnt-inody
sudo mount -o loop male-inody.img mnt-inody
df -i mnt-inody
```

`-N 50` każe `mkfs.ext4` utworzyć system plików z zaledwie 50 inode'ami, niezależnie od jego rozmiaru w bajtach. Teraz spróbuj utworzyć 60 plików w pętli i obejrzyj, co się stanie po drodze:

```
for i in $(seq 1 60); do touch mnt-inody/plik$i 2>&1 | tail -1; done
df -h mnt-inody
df -i mnt-inody
```

Odpowiedź

Po kilkudziesięciu plikach (mniej niż 50, bo część inode'ów zajmuje sam katalog główny systemu plików i katalogi zarezerwowane) pętla zaczyna wypisywać:

```
touch: cannot touch 'mnt-inody/plik47': No space left on device
```

`df -h mnt-inody` pokaże w tym momencie mnóstwo wolnego miejsca (16 MB obrazu ledwo tknięte), a `df -i mnt-inody` pokaże `IFree` bliskie zeru. To jest dokładnie scenariusz z sekcji *Częste błędy*, tylko odtworzony bezpiecznie, na sztucznie małym systemie plików zamiast na produkcyjnym `/var`.

Sprzątanie:

```
sudo umount mnt-inody
rm -f male-inody.img
```

Ćwiczenie 3 — inspekcja własnego systemu bez zmian

```
lsblk -f
findmnt /
blkid | head -3
df -Th | grep -v tmpfs
```

Znajdź UUID systemu plików, na którym leży `/`, trzema różnymi komendami i sprawdź, czy wszystkie trzy się zgadzają.

Odpowiedź

`lsblk -f`, `blkid` i `findmnt -no UUID /` powinny zwrócić dokładnie ten sam UUID dla urządzenia zamontowanego pod `/` — to jest ten sam identyfikator, odczytany z tego samego miejsca w nagłówku systemu plików, tylko trzema różnymi narzędziami. Rozjazd między nimi (poza literówką przy przepisywaniu) praktycznie się nie zdarza.

Pytania kontrolne

1. Dlaczego `df` odpowiada natychmiast nawet na dysku z milionami plików, a `du -sh` na tym samym dysku potrafi liczyć minutami?

Odpowiedź

`df` odpytuje jądro o gotowe liczniki systemu plików (całkowita i wolna liczba bloków) przez jedno wywołanie `statfs()` — koszt nie zależy od liczby plików. `du` musi rekurencyjnie odwiedzić każdy plik i katalog, żeby zsumować ich rozmiary — koszt rośnie wraz z liczbą plików, nie z ich łącznym rozmiarem.

2. System pokazuje `df -h` z 40% wolnego miejsca, a mimo to nie da się utworzyć nowego pliku. Jakie polecenie sprawdzisz w pierwszej kolejności i czego szukasz w wyniku?

Odpowiedź

`df -i` na tym samym punkcie montowania — szukasz kolumny `IFree` bliskiej zeru albo `IUse%` bliskiego 100%. To sygnał wyczerpania inode'ów, częsty przy katalogach z milionami drobnych plików (sesje, cache, kolejki).

3. Masz katalog `/data` z zamontowanym pod nim zasobem NFS `/data/backup`. Jak policzyć rozmiar samych lokalnych plików w `/data`, bez wliczania zawartości NFS?

Odpowiedź

`du -sh -x /data`. Flaga `-x` zatrzymuje `du` na granicy systemu plików, na którym leży punkt startowy, więc nie zejdzie do zamontowanego pod spodem NFS.

4. Czym różni się to, co pokazuje `lsblk`, od tego, co pokazuje `df`?

Odpowiedź

`lsblk` pokazuje urządzenia blokowe — fizyczne i wirtualne dyski, partycje, wolumeny LVM — niezależnie od tego, czy mają na sobie system plików i czy jest on zamontowany. `df` pokazuje wyłącznie systemy plików, które są aktualnie zamontowane, razem z ich zajętością. Dysk widoczny w `lsblk`, ale bez systemu plików albo niezamontowany, w ogóle nie pojawi się w `df`.

Powiązane: Gdy df i du nie zgadzają się: usunięte otwarte pliki · LVM: dysk fizyczny, grupa wolumenów, wolumen logiczny · Powiększanie przestrzeni dyskowej bez restartu · Uprawnienia i atrybuty plików

Źródła: `man 1 df` · `man 1 du` · `man 8 lsblk` · `man 8 lsscsi` · `man 8 blkid` · `man 8 findmnt` · `man 1 ncdu` · `man 5 ext4` (rezerwa dla roota) · `man 8 tune2fs`

Revision #16
Created 2026-09-17 21:43:00 UTC by Tomasz Konieczny
Updated 2026-10-03 19:08:40 UTC by Tomasz Konieczny