find: wyszukiwanie plików po atrybutach

Poziom: podstawy/średni **Dotyczy: wszystkie dystrybucje** **Czas: ~14 min**

`find` przechodzi drzewo katalogów i sprawdza każdy plik względem listy warunków, które sam zdefiniujesz. To narzędzie do szukania po atrybutach — nazwie, typie, dacie, rozmiarze, uprawnieniach — a nie po zawartości.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
ERE	Extended Regular Expression: rozszerzona odmiana wyrażeń regularnych, bez ukośnika wstecznego przed nawiasami i plusem
NFS	Network File System: udostępnianie katalogów po sieci w świecie Linuksa
POSIX	Portable Operating System Interface: wspólny zestaw zasad, dzięki któremu polecenie działa tak samo na różnych Uniksach
UID	User ID: numer użytkownika w Linuksie

Fiszka

TYLKO ODCZYT

Dopasowanie nazwy pliku: -name

```
find /var/log -name "*.log"
```

Dopasowujemy nazwę pliku do wzorca glob (`*`, `?`, `[...]`), z rozróżnieniem wielkości liter. Wzorzec musi stać w cudzysłowie — inaczej rozwinie go powłoka, zanim `find` w ogóle wystartuje.

Przykład:

```
/var/log/messages.log
/var/log/aplikacja/debug.log
```

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

TYLKO ODCZYT

Dopasowanie nazwy bez rozróżniania wielkości liter: -iname

```
find /var/log -iname "*.LOG"
```

To samo co `-name`, ale ignorujemy wielkość liter. Przydaje się, gdy nie mamy pewności, czy plik nazywa się `raport.log` czy `raport.LOG`.

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

TYLKO ODCZYT

Dopasowanie całej ścieżki: -path

```
find . -path "*/config/*.conf"
```

W odróżnieniu od `-name` sprawdzamy wzorzec względem **całej ścieżki**, nie samej nazwy pliku — pozwala to wybrać albo wykluczyć cały katalog, niezależnie od tego, jak nazywa się plik w środku.

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

TYLKO ODCZYT

Dopasowanie ścieżki wyrażeniem regularnym: -regex

```
find . -regex ".*/[0-9]+\.txt"
```

Jak `-path`, ale zamiast wzorca glob bierzemy wyrażenie regularne — i od razu dopasowujemy je do **całej ścieżki**. Domyślnie `find` używa własnego dialektu Emacs, nie POSIX ERE, co bywa zaskoczeniem dla kogoś, kto zna wyrażenia regularne z `grep -E`.

POSIX = Portable Operating System Interface: wspólny zestaw zasad, dzięki któremu polecenie działa tak samo na różnych Uniksach · **ERE** = Extended Regular Expression: rozszerzona odmiana wyrażeń regularnych, bez ukośnika wstecznego przed nawiasami i plusem · **find** = przeszukaj drzewo katalogów według nazwy, typu, daty albo rozmiaru

TYLKO ODCZYT

Filtrowanie po typie wpisu: -type

```
find /etc -type f
find / -type l
find /var/run -type s
```

Filtrujemy po rodzaju wpisu w systemie plików: `f` (zwykły plik), `d` (katalog), `l` (dowiązanie symboliczne), `s` (gniazdo), `b` / `c` (urządzenie blokowe/znakowe), `p` (potok nazwany). Bez `-type f` komenda licząca rozmiar czy wiek złapie też katalogi, co zwykle zaśmieca wynik.

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

TYLKO ODCZYT

Filtrowanie po czasie modyfikacji zawartości (w dniach): -mtime

```
find /tmp -type f -mtime +7
```

Filtrujemy po dacie ostatniej modyfikacji **zawartości** pliku, licząc w pełnych dobach. `+7` znaczy „starsze niż 7 dni", `-7` — „młodsze niż 7 dni", samo `7` bez znaku — „dokładnie siódmego dnia wstecz". Trzy różne zbiory wyników pod trzema bardzo podobnymi zapisami.

Przykład:

```
/tmp/build-8821/cache.tmp
/tmp/upload-4471.part
```

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

TYLKO ODCZYT

Filtrowanie po czasie modyfikacji (w minutach): -mmin

```
find . -mmin -30
```

To samo co `-mtime`, tylko w minutach zamiast dni — sięgamy po to, gdy doba jest zbyt grubą jednostką, np. przy sprawdzaniu, co zmieniło się w ostatnim kwadransie.

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

TYLKO ODCZYT

Porównanie z czasem modyfikacji wskazanego pliku: -newer

```
find . -newer /tmp/znacznik.txt
```

Zamiast liczby dni czy minut porównujemy mtime każdego pliku z mtime wskazanego pliku-znacznika. Wygodne, gdy mamy konkretny punkt odniesienia w czasie zamiast chęci liczenia „ile to dni temu".

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

TYLKO ODCZYT

Filtrowanie po rozmiarze pliku: -size

```
find /var/www -size +10M
find . -size 0 -type f
```

Filtrujemy po rozmiarze. Jednostkę podajemy literą: `c` (bajty), `k` (kibibajty), `M` (mebibajty), `G` (gibibajty). Bez jednostki domyślną jednostką są bloki po 512 B, nie bajty — częsty błąd, opisany niżej w *Częste błędy*. `-size 0` (bez znaku) to pliki o rozmiarze dokładnie zero.

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

TYLKO ODCZYT

Filtrowanie po właścicielu: -user

```
find /home -user nazwa_usera
```

Znajdujemy pliki należące do wskazanego użytkownika — przydatne przy audycie albo przy sprzątaniu po koncie, które właśnie zwolniono.

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

TYLKO ODCZYT

Filtrowanie po grupie: -group

```
find /srv/dane -group NazwaGrupy
```

Jak `-user`, tylko według grupy właścicielskiej pliku, nie konkretnego użytkownika.

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

TYLKO ODCZYT

Wyszukanie plików bez właściciela w bazie systemu: -nouser

```
find / -xdev -nouser
```

Znajdujemy pliki, których UID właściciela nie odpowiada żadnemu kontu w `/etc/passwd` — typowy ślad po usuniętym koncie, którego pliki zostały na dysku. `-xdev` ogranicza przeszukiwanie do jednego systemu plików, żeby nie wchodzić w `/proc` i montowania sieciowe.

UID = User ID: numer użytkownika w Linuksie · **find** = przeszukaj drzewo katalogów według nazwy, typu, daty albo rozmiaru

TYLKO ODCZYT

Filtrowanie po uprawnieniach: -perm

```
find /usr/bin -perm -4000
find . -perm -002
find . -perm 644
```

Trzy tryby zapisu dają trzy różne pytania: `-perm 644` (bez prefiksu) — uprawnienia muszą być **dokładnie** takie; `-perm -4000` (prefiks `-`) — wskazane bity muszą być ustawione, reszta dowolna (tu: bit setuid); `-perm /tryb` (prefiks `/`) — wystarczy, że pasuje którykolwiek z wymienionych bitów. Drugą linią szukamy plików z prawem zapisu dla „innych".

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

TYLKO ODCZYT

Wyszukanie pustych plików i katalogów: -empty

```
find /var/spool -empty
```

Dopasowujemy plik o rozmiarze zero albo katalog bez żadnej zawartości. Częste zastosowanie: czyszczenie artefaktów po nieudanych zadaniach wsadowych.

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

TYLKO ODCZYT

Ograniczenie głębokości w dół: -maxdepth

```
find /etc -maxdepth 1 -type f
```

Nie każemy `find` schodzić głębiej niż podana liczba poziomów od ścieżki startowej. `-maxdepth 1` ogranicza wynik do plików bezpośrednio w `/etc`, pomijając `/etc/nginx/`, `/etc/dnf/` i resztę podkatalogów. W GNU find musi stać przed pozostałymi testami w wyrażeniu.

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

TYLKO ODCZYT

Pominięcie płytkich poziomów: -mindepth

```
find / -mindepth 2 -maxdepth 2 -type d
```

Odwrotność `-maxdepth`: każemy `find` zignorować wpisy płytsze niż podany poziom. Para `-mindepth 2 -maxdepth 2` daje dokładnie drugi poziom katalogów, ani płycej, ani głębiej.

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

TYLKO ODCZYT

Odcięcie poddrzewa: -prune

```
find / -path /proc -prune -o -path /sys -prune -o -name "*.conf" -print
```

Każemy `find` w ogóle nie wchodzić do wskazanego katalogu, zamiast wejść i odfiltrować wynik dopiero po fakcie. Wymaga `-o` (OR) przed dalszą częścią wyrażenia, bo samo `-prune` nic nie drukuje.

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

TYLKO ODCZYT

Łączenie warunków: -o, !, nawiasy

```
find . -type f \( -name "*.log" -o -name "*.txt" \) -mtime -7
find . -type f ! -name "*.bak"
```

Bez operatora między testami `find` domyślnie łączy je przez AND. `-o` to OR, `!` to negacja, a nawiasy (zescapowane `\(` `\)`, bo swoje nawiasy ma też powłoka) grupują fragment wyrażenia — niezbędne, gdy mieszamy AND z OR, bo mają różny priorytet.

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

TYLKO ODCZYT

Ustalenie własnego formatu wyjścia: -printf

```
find . -maxdepth 1 -type f -printf "%TY-%Tm-%Td %TH:%TM  %8s  %p\n"
```

Zamiast samej ścieżki wypisujemy dowolny format złożony z placeholderów: `%p` ścieżka, `%s` rozmiar w bajtach, `%TY-%Tm-%Td %TH:%TM` data i godzina modyfikacji, `%m` uprawnienia ósemkowo, `%u` właściciel. Niezbędne, gdy wynik ma być dalej sortowany albo przetwarzany.

Przykład:

```
2026-09-10 08:14      1048576  ./raport-sierpien.csv
2026-09-15 22:03        52480  ./notatki.txt
```

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

TYLKO ODCZYT

Wszystkie testy

Test	Co robi	Kiedy używasz
`-name "wzorzec"`	dopasowuje nazwę, uwzględnia wielkość liter	znasz dokładną pisownię rozszerzenia
`-iname "wzorzec"`	jak wyżej, ignoruje wielkość liter	nie masz pewności co do wielkości liter
`-path "wzorzec"`	dopasowuje całą ścieżkę, nie samą nazwę	chcesz wykluczyć/wybrać cały katalog
`-regex "wzorzec"`	dopasowuje ścieżkę wyrażeniem regularnym	`-name` za mało elastyczne

Test	Co robi	Kiedy używasz
`-type f/d/l/b/c/p/s`	filtr wg typu wpisu	odróżnienie plików od katalogów, linków itd.
`-mtime ±N`	wg czasu modyfikacji zawartości, w dniach	szukasz starych/nowych plików danych
`-mmin ±N`	jak wyżej, w minutach	precyzja mniejsza niż doba
`-newer plik`	nowsze niż mtime wskazanego pliku	porównanie względem znacznika czasu
`-size ±N[jednostka]`	wg rozmiaru (c/k/M/G)	szukanie dużych logów, pustych plików
`-user` / `-group`	wg właściciela/grupy	audyt, sprzątanie po zwolnionym koncie
`-nouser`	brak właściciela w `/etc/passwd`	ślad po usuniętym koncie
`-perm tryb`	wg uprawnień (`-tryb`, `/tryb`, `tryb`)	szukanie setuid, plików z dziurawym `chmod`
`-empty`	plik lub katalog o rozmiarze zero	czyszczenie pustych artefaktów
`-maxdepth N` / `-mindepth N`	ogranicza głębokość rekursji	wydajność, brak schodzenia w podkatalogi
`-prune`	odcina poddrzewo z dalszego przeszukiwania	pomijanie `.git`, `node_modules`, `/proc`
`-printf format`	własny format wyjścia	sortowanie, dalsze przetwarzanie wyniku

Uważaj

- `find . -name *.log` bez cudzysłowu: jeśli w katalogu jest choć jeden plik `.log`, powłoka rozwinie `*.log` PRZED uruchomieniem `find` i komenda dostanie konkretną nazwę zamiast wzorca.
- `-mtime +7` to „starsze niż 7 dni", `-mtime -7` to „młodsze niż 7 dni", `-mtime 7` to dokładnie 7 dni — trzy różne zbiory wyników.
- `-maxdepth` musi stać przed pozostałymi testami w wyrażeniu, inaczej GNU find zgłosi błąd składni.

Z życia wzięte

> **Z ŻYCIA WZIĘTE**
>
> Piątek, 14:30. Ola prosi Tomka o znalezienie wszystkich plików `.log` starszych niż tydzień na `ci01`, z pominięciem repozytoriów Gitea. Tomek pisze `find /docker -path "*/.git" -prune -o -name "*.log" -mtime +7 -print` i dziwi się, że bez `-prune` `find` schodziłby też do wnętrza każdego `.git`, marnując czas na tysiące obiektów, których nikt nie chce ruszać. Po dodaniu `-prune` komenda kończy się w sekundę zamiast w minutę.

Po co to istnieje

Masz serwer, na którym `/var/log` puchnie od miesięcy, bo jedna aplikacja nie ma logrotate. Chcesz znaleźć wszystkie pliki `.log` większe niż 100 MB, których nikt nie ruszał od tygodnia. `ls` tego nie zrobi — pokaże zawartość jednego katalogu, bez rekursji i bez filtrów po rozmiarze czy dacie.

`find` powstał dokładnie do tego: przejść drzewo katalogów — dowolnie głębokie — i dla każdego napotkanego pliku czy katalogu sprawdzić listę warunków, które podajesz. Nazwa, typ, wiek, rozmiar, właściciel, uprawnienia. Wszystko, co system plików wie o pliku, poza jego zawartością — do tego służy `grep`, nie `find`. To rozróżnienie jest ważne i często mylone: `find` szuka *plików pasujących do atrybutów*, `grep -r` szuka *tekstu wewnątrz plików*. Można je łączyć (kolejna strona tego rozdziału pokazuje jak), ale to dwa różne narzędzia do dwóch różnych pytań.

Analogia

Wyobraź sobie **inwentaryzację w wielkim magazynie**, prowadzoną z listą kryteriów w ręku.

Chodzisz regał po regale — to jest rekurencyjny obchód drzewa katalogów. Przy każdej paczce sprawdzasz listę: czy etykieta pasuje do wzorca, czy data przyjęcia jest starsza niż tydzień, czy waga przekracza 10 kg, czy paczka w ogóle jest paczką, a nie pustym kartonem. Jeśli spełnia wszystkie warunki z listy — zapisujesz ją. Jeśli wystarczy spełnić jeden z dwóch warunków, mechanizm chodzenia regał po regale zostaje ten sam — zmienia się tylko sposób łączenia kryteriów.

Kluczowe ograniczenie tej analogii, które świetnie tłumaczy działanie `find`: **listę kryteriów sprawdzasz dla każdej paczki z osobna, w momencie, gdy do niej dochodzisz**, a nie z góry przygotowanego spisu całego magazynu. To dlatego, jeśli powiesz brygadziście „nie wchodź na to zaplecze" (`-prune`), on w ogóle nie zajrzy do środka, zamiast zajrzeć i dopiero potem odrzucić wszystko, co tam znajdzie — i dlatego kolejność kryteriów decydujących „czy w ogóle wchodzę w ten regał" (`-maxdepth`, `-prune`) musi być ustalona, zanim zaczniesz oceniać zawartość.

Jak to działa

Rozwiń: jak to działa pod spodem

Składnia ogólna:

```
find ŚCIEŻKA WYRAŻENIE
```

`ŚCIEŻKA` to punkt startowy obchodu (może być ich kilka). `WYRAŻENIE` to ciąg testów i akcji połączonych operatorami logicznymi — bez podanego operatora między testami `find` domyślnie łączy je przez **AND** (odpowiednik `-a`), więc plik musi spełnić wszystkie naraz.

`find` przechodzi drzewo w porządku „od korytarza w głąb" (depth-first) i dla każdego napotkanego wpisu ocenia wyrażenie od lewej do prawej. Gdy warunek się nie zgadza, przestaje oceniać dalsze testy dla tego wpisu i przechodzi do następnego — to samo leniwe wartościowanie, które znasz z `&&` w powłoce.

Cudzysłów przy wzorcach nazwy jest obowiązkowy, i to nie jest kwestia stylu. `-name` i `-iname` oczekują wzorca glob (`*`, `?`, `[...]`) jako **jednego argumentu tekstowego**, który sam `find` przetworzy wewnętrznie. Jeśli napiszesz `find . -name *.log` bez cudzysłowu, to **powłoka**, a nie `find`, zobaczy `*.log` jeszcze przed uruchomieniem komendy i rozwinie go na listę pasujących plików w bieżącym katalogu. Jeśli taki plik istnieje, `find` dostanie konkretną nazwę zamiast wzorca — komenda „zadziała", ale przeszuka tylko ten jeden plik. To pułapka opisana szerzej na stronie *Cudzysłowy, escapowanie i ekspansja*: ekspansja wzorców (globbing) należy do powłoki, nie do `find`, a cudzysłów to jedyny sposób, żeby powłoka zostawiła wzorzec w spokoju.

`-regex` działa inaczej niż `-name` — dopasowuje **całą ścieżkę**, a nie samą nazwę pliku, i domyślnie używa składni Emacs, nie POSIX ERE.

Testy czasu operują na trzech różnych znacznikach z i-węzła:

Test	Co mierzy
`-mtime`	czas ostatniej modyfikacji **zawartości** pliku
`-atime`	czas ostatniego **odczytu** pliku
`-ctime`	czas ostatniej zmiany **metadanych** i-węzła (uprawnienia, właściciel, link, ale też sama modyfikacja zawartości)

`-atime` bywa niewiarygodne — wiele systemów montuje partycje z opcją `relatime` albo `noatime`, żeby nie zapisywać czasu dostępu przy każdym odczycie. `-ctime` zmienia się częściej

niż się wydaje: `chmod`, `chown` i zmiana liczby twardych linków aktualizują `ctime`, mimo że zawartość pliku się nie zmieniła.

Pułapka `+N` / `-N` / `N` dotyczy wszystkich testów liczbowych (`-mtime`, `-mmin`, `-size`, `-links`...), nie tylko czasu:

- `N` (bez znaku) — dokładnie N (dla `-mtime`: w przedziale doby N dni temu, bo `find` liczy w pełnych 24-godzinnych okresach)
- `+N` — więcej niż N
- `-N` — mniej niż N

`find /tmp -mtime 7` nie znaczy „starsze niż tydzień" — znaczy „zmodyfikowane dokładnie siódmego dnia wstecz", czyli wąski przedział 24 godzin sprzed tygodnia. To najczęściej mylony test w całym `find`.

Rozmiar (`-size`) przyjmuje jednostkę literą: `c` (bajty), `k` (kibibajty, 1024 B), `M` (mebibajty), `G` (gibibajty). **Jeśli pominiesz jednostkę, domyślną jednostką są bloki po 512 bajtów**, nie bajty. `find . -size +100` znajdzie pliki większe niż 100 bloków, czyli 51200 bajtów. To druga najczęstsza pułapka po `-mtime`.

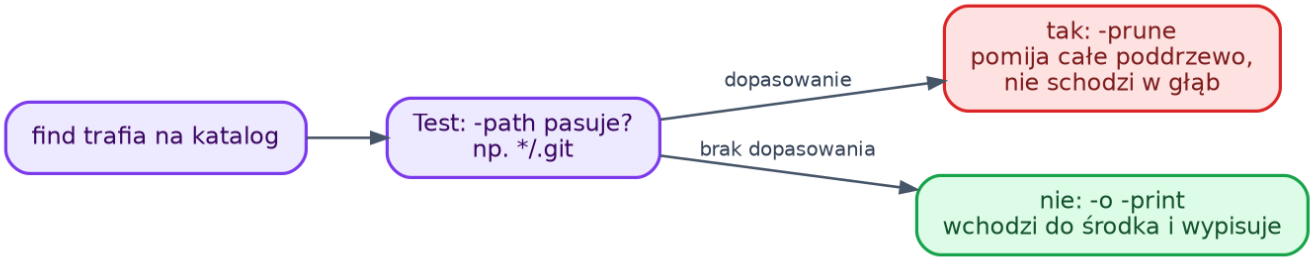
Uprawnienia (`-perm`) mają trzy tryby zapisu:

- `-perm 644` (bez prefiksu) — uprawnienia muszą być **dokładnie** takie
- `-perm -644` (prefiks `-`) — wszystkie wskazane bity muszą być ustawione, inne bity mogą być dowolne
- `-perm /644` (prefiks `/`) — **którykolwiek** ze wskazanych bitów wystarczy

`find / -perm -4000` to klasyczne polowanie na pliki z bitem setuid — plik z setuid uruchamia się z uprawnieniami właściciela, nie użytkownika, który go odpala, więc audyt bezpieczeństwa regularnie to sprawdza.

`-maxdepth` i `-mindepth` kontrolują, jak głęboko `find` schodzi w drzewo. GNU find wymaga, żeby stały **przed** innymi testami w wyrażeniu (to ograniczenie tej konkretnej implementacji, nie standardu POSIX) — inaczej dostaniesz błąd w rodzaju `find: paths must precede expression` albo ostrzeżenie o kolejności opcji.

`-prune` każe `find` w ogóle nie wchodzić do danego katalogu. To różni się fundamentalnie od odfiltrowania wyników po fakcie — `-prune` oszczędza cały czas, jaki zajęłoby przejście poddrzewa. Typowy idiom: `find . -path "*/.git" -prune -o -print` — pomiń `.git`, wypisz resztę. Zwróć uwagę na `-o` (OR) między `-prune` a `-print`: to nie przypadek, tylko wymóg logiki wyrażenia, bo `-prune` samo w sobie nic nie drukuje.



Decyzja find przy każdym katalogu: dopasowanie do -prune pomija poddrzewo, brak dopasowania wchodzi i wypisuje (kliknięcie otwiera pełny rozmiar)

Krok po kroku

1. Znajdź wszystkie pliki konfiguracyjne w `/etc`, bez schodzenia w podkatalogi.

```
find /etc -maxdepth 1 -type f -name "*.conf"
```

```
/etc/resolv.conf
/etc/sudo.conf
```

Bez `-maxdepth 1` `find` wszedłby też do `/etc/dnf/`, `/etc/nginx/` i dziesiątek innych podkatalogów.

2. Znajdź pliki tymczasowe starsze niż tydzień.

```
find /tmp -type f -mtime +7
```

Zwróć uwagę na `-type f` — bez niego `find` dopasuje też katalogi, których mtime zmienia się przy każdym utworzeniu pliku w środku, co zaśmieci wynik.

3. Znajdź duże pliki logów, posortowane od największego.

```
find /var/log -type f -name "*.log" -printf "%s %p\n" | sort -rn | head -10
```

```
524288000 /var/log/aplikacja/debug.log
104857600 /var/log/messages
```

`-printf` daje kontrolę nad formatem wyjścia — `%s` to rozmiar w bajtach, `%p` to ścieżka. `sort -rn` sortuje malejąco numerycznie, bo domyślne sortowanie tekstowe potraktowałoby `104857600` jako „mniejsze" niż `52428800`, porównując znak po znaku.

4. Znajdź pliki setuid w systemie (audyt bezpieczeństwa).

```
sudo find / -xdev -type f -perm -4000 -printf "%m %u %p\n"
```

```
4755 root /usr/bin/sudo
4755 root /usr/bin/passwd
4711 root /usr/bin/pkexec
```

`-xdev` ogranicza przeszukiwanie do jednego systemu plików — bez tego `find /` wejdzie też w `/proc`, `/sys` i ewentualne NFS, co jest wolne i bez sensu przy audycie lokalnych binarek.

Częste błędy

Niezacytowany wzorzec `-name`.

```
find . -name *.log
```

```
find: paths must precede expression: raport.log
find: this option is not valid when watching processes
```

Przy jednym pasującym pliku powłoka rozwinie wzorzec przed uruchomieniem `find` i komenda dostanie `find . -name raport.log` — zadziała, ale tylko dla tego jednego pliku. Przy kilku pasujących plikach `find` dostanie kilka argumentów po `-name` i posypie się błędem jak wyżej, bo kolejny plik zostanie zinterpretowany jako dodatkowa ścieżka startowa. Naprawa: zawsze cytuj wzorce glob i regex przekazywane do `find`.

```
find . -name "*.log"
```

Pomylenie `-mtime +7` z `-mtime -7` (albo z samym `-7`).

Ktoś chce „pliki starsze niż tydzień, do wywalenia" i pisze:

```
find /var/tmp -mtime -7
```

To da dokładnie odwrotny zestaw — pliki zmodyfikowane **w ciągu ostatnich** 7 dni, czyli świeże. Jeśli trafi to do `-delete` (kolejna strona tej sekcji), skasujesz świeże dane, a stare śmieci zostaną nietknięte. Właściwy zapis dla „starsze niż 7 dni":

```
find /var/tmp -mtime +7
```

`-size` bez jednostki, licząc na bajty.

```
find . -size +100
```

Autor tej komendy chciał plików większych niż 100 bajtów. Dostał pliki większe niż 100 bloków po 512 B, czyli powyżej 51200 bajtów. Naprawa: zawsze pisz jednostkę.

```
find . -size +100c        # bajty
find . -size +100k        # kibibajty
```

Zapomniane nawiasy przy mieszaniu `-a`, `-o` i `!`.

```
find . -name "*.tmp" -o -name "*.bak" -mtime +7
```

Wygląda jak „pliki `.tmp` LUB `.bak`, starsze niż 7 dni", ale `-o` ma niższy priorytet niż domyślne AND między sąsiednimi testami — wyrażenie parsuje się jako `-name "*.tmp" -o (-name "*.bak" -a -mtime +7)`. Efekt: **każdy** plik `.tmp` przechodzi niezależnie od wieku, bo warunek wieku jest doklejony tylko do `.bak`. Naprawa — jawne grupowanie (nawiasy trzeba zescapować, żeby powłoka nie potraktowała ich jako grupowania poleceń):

```
find . \( -name "*.tmp" -o -name "*.bak" \) -mtime +7
```

□□ Ćwiczenia

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

Ćwiczenie 1 — drzewo testowe

Przygotuj testowe drzewo katalogów w `/tmp`.

```
mkdir -p /tmp/lab/{a,b/c}
touch /tmp/lab/a/raport.log /tmp/lab/a/dane.LOG /tmp/lab/b/notatka.txt
touch /tmp/lab/b/c/archiwum.tar.gz
dd if=/dev/zero of=/tmp/lab/b/c/duzy.bin bs=1M count=5 2>/dev/null
touch -d '10 days ago' /tmp/lab/a/raport.log
```

1. Znajdź wszystkie pliki `.log` niezależnie od wielkości liter, w całym `/tmp/lab`.
2. Znajdź pliki większe niż 1 MB.
3. Znajdź pliki starsze niż 5 dni.
4. Znajdź wszystko w `/tmp/lab/b`, pomijając poddrzewo `c`.

Odpowiedź

```
find /tmp/lab -iname "*.log"
find /tmp/lab -type f -size +1M
find /tmp/lab -type f -mtime +5
find /tmp/lab/b -path "*/c" -prune -o -print
```

Pierwsza komenda znajdzie `raport.log` i `dane.LOG` — `-iname` ignoruje wielkość liter. Trzecia komenda złapie tylko `raport.log`, bo tylko jemu ustawiono datę wstecz przez `touch -d`.

Pytania kontrolne

1. Dlaczego `find . -name *.txt` może czasem "zadziałać", a czasem posypać się błędem, mimo że nie zmieniłeś ani jednej litery w komendzie?

Odpowiedź

To nie `find` decyduje o losie tego wzorca, tylko powłoka, zanim `find` wystartuje. Zero plików `.txt` — powłoka (bez `nullglob`) przekaże literalny napis dalej, przypadkiem jakby był cytowany. Jeden plik `.txt` — powłoka podstawi jego nazwę, komenda „zadziała", ale przeszuka tylko ten plik. Więcej niż jeden — `find` dostanie kilka argumentów po `-name` i zgłosi błąd, bo kolejne zostaną potraktowane jako dodatkowe ścieżki startowe. Rozwiązanie zawsze to samo: cytować wzorzec.

2. Masz katalog z plikami modyfikowanymi codziennie. Chcesz „wszystko starsze niż wczoraj". Napiszesz `-mtime -1` czy `-mtime +1`, i dlaczego żadne z nich nie jest dokładnie tym, o co pytasz?

Odpowiedź

`-mtime +1` da „starsze niż 2 dni", a `-mtime -1` da „młodsze niż 1 dzień". Żadne nie odpowiada dokładnie „starsze niż wczoraj", bo `-mtime` liczy w pełnych 24-godzinnych oknach od teraz, nie od północy kalendarzowej. Do cięcia „od konkretnej daty" używa się `-newer` z plikiem-znacznikiem, albo `-newermt "2026-09-16 00:00:00"` w GNU find.

3. Czym się różni `find . -perm 644`, `find . -perm -644` i `find . -perm /644`?

Odpowiedź

`-perm 644` wymaga uprawnień DOKŁADNIE `644` — jeśli plik ma `664`, nie pasuje. `-perm -644` wymaga, żeby wskazane bity były ustawione, ale dopuszcza dodatkowe — plik `664` już pasuje, bo zawiera wszystkie bity z `644`. `-perm /644` wymaga, żeby pasował którykolwiek z wymienionych bitów, więc złapie też plik z samym prawem odczytu dla właściciela. Do audytu „czy ktoś oprócz właściciela ma zapis" właściwy jest wariant z `/`, nie z `-` czy bez prefiksu.

4. Do czego służy `-prune` i dlaczego nie da się tego samego osiągnąć przez `! -path "*/wzorzec/*"`?

Odpowiedź

`-prune` każe `find` w ogóle nie wchodzić do wskazanego katalogu. `! -path "*/wzorzec/*"` odfiltrowuje wyniki dopiero PO tym, jak `find` już zszedł do środka i przetworzył każdy plik w tym poddrzewie — wynik końcowy jest ten sam, ale jest wolniejsze na dużych katalogach jak `node_modules` czy `.git`, gdzie liczba plików idzie w dziesiątki tysięcy.

Powiązane: Działanie na wynikach: -exec, -delete, xargs · Cudzysłowy, escapowanie i ekspansja · System plików, ścieżki, pwd, cd, ls · Drobnica, która oszczędza minuty: wc, nl, tac, rev, split

Źródła: `man 1 find` · `info find` · GNU findutils manual, sekcja „Finding Files"

Revision #19
Created 2026-09-17 15:06:49 UTC by Tomasz Konieczny
Updated 2026-10-03 19:07:49 UTC by Tomasz Konieczny