Uprawnienia klasyczne: chmod, chown, chgrp

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

`chmod`, `chown` i `chgrp` to trzy komendy, którymi zarządza się klasycznym modelem uprawnień: kto jest właścicielem, do jakiej grupy plik należy i co wolno zrobić właścicielowi, grupie i reszcie świata. Ta strona przechodzi przez wszystkie warianty zapisu i pokazuje, dlaczego katalog bez prawa `x` jest zablokowany, nawet gdy ma prawo `r`.

Skróty użyte na tej stronie (5)

Skrót	Co znaczy
ACL	Access Control List: lista, kto ma jakie prawa do pliku lub zasobu
B2B	business to business: sprzedaż i systemy skierowane do innych firm, nie do klientów indywidualnych
PHP	PHP: Hypertext Preprocessor, język programowania, w którym napisany jest frontend Zabbixa i wiele innych aplikacji webowych
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

Fiszka

TYLKO ODCZYT

Odczyt bieżących uprawnień: ls -l

```
ls -l NazwaPliku
```

Zanim cokolwiek zmienimy, sprawdzamy stan wyjściowy. Format `-rwxr-xr--` rozbija się na cztery części: typ pliku, uprawnienia właściciela, uprawnienia grupy, uprawnienia pozostałych — pełny rozkład w sekcji *Jak to działa*.

Przykład:

```
-rwxr-xr-- 1 adm_user deweloperzy 812 wrz 17 09:12 raport.txt
```

ls = list: wypisz zawartość katalogu

ZMIENIA SYSTEM

Dodanie prawa wykonania właścicielowi: chmod u+x

```
chmod u+x NazwaPliku
```

`u` (user, czyli właściciel), `+` (dodanie), `x` (execute). Klasyczny odruch po napisaniu nowego skryptu — bez tego `./NazwaPliku` kończy się komunikatem `Permission denied`.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Odebranie zapisu grupie: chmod g-w

```
chmod g-w NazwaPliku
```

`g` (group), `-` (odebranie), `w` (write). Grupa nadal może czytać plik, jeśli miała `r`, ale nie może go już modyfikować.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Dodanie odczytu pozostałym: chmod o+r

```
chmod o+r NazwaPliku
```

`o` (others) obejmuje każdego, kto nie jest właścicielem ani nie należy do grupy pliku. Dodanie `r` udostępnia plik do odczytu całej reszcie systemu.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Wyzerowanie uprawnień pozostałych: chmod o=

```
chmod o= NazwaPliku
```

Znak `=` to przypisanie bezwzględne, nie dodawanie ani odejmowanie — cokolwiek `o` miało wcześniej, po tej komendzie ma dokładnie nic. Różnica względem `o-rwx` jest kosmetyczna przy pełnym wyzerowaniu, ale `=` przydaje się, gdy chcemy ustawić konkretny zestaw praw niezależnie od stanu wyjściowego, np. `chmod o=r`.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Nadanie odczytu wszystkim trzem grupom naraz: chmod a+r

```
chmod a+r NazwaPliku
```

`a` (all) to skrót na `ugo` razem — dodajemy prawo odczytu właścicielowi, grupie i pozostałym jedną komendą.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Pełna blokada dla grupy i pozostałych: chmod go-rwx

```
chmod go-rwx NazwaPliku
```

Łączenie liter (`go` zamiast osobnych `g` i `o`) i pełnego zestawu uprawnień w jednej komendzie. Po tym wywołaniu tylko właściciel ma jakikolwiek dostęp do pliku.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Zapis ósemkowy dla typowego pliku: chmod 644

```
chmod 644 NazwaPliku
```

Właściciel: `rw-` (6), grupa: `r--` (4), pozostali: `r--` (4). Standard dla plików tekstowych i konfiguracyjnych, które mają być czytelne, ale modyfikowalne tylko przez właściciela.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Zapis ósemkowy dla typowego skryptu/katalogu: chmod 755

```
chmod 755 NazwaPliku
```

Właściciel: `rwx` (7), grupa: `r-x` (5), pozostali: `r-x` (5). Domyślny zestaw dla plików wykonywalnych i katalogów, które mają być przechodnie dla wszystkich, ale edytowalne tylko przez właściciela.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Ograniczenie dostępu do samego właściciela: chmod 700

```
chmod 700 NazwaPliku
```

Właściciel: pełny dostęp, grupa i pozostali: zero. Wariant dla skryptów i katalogów, które nie powinny być widoczne dla nikogo poza właścicielem — np. katalog `~/.ssh`.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Zabezpieczenie sekretów plikowych: chmod 600

```
chmod 600 NazwaPliku
```

Właściciel: `rw-`, reszta: nic. Ten zestaw jest obowiązkowy dla wszystkiego, co zawiera hasła, tokeny czy klucze prywatne — patrz *Częste błędy*, komunikat sshd o niezabezpieczonym kluczu.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Rekurencyjna zmiana uprawnień: chmod -R

```
chmod -R 755 /var/www/NazwaKatalogu
```

`-R` (recursive) schodzi do każdego pliku i podkatalogu wewnątrz wskazanej ścieżki. Wygodne, ale ryzykowne — patrz pułapkę z rekurencyjnym trybem numerycznym w *Częste błędy*.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Rekurencyjna naprawa z dziedziczeniem prawa wykonania: chmod -R u+rwX

```
chmod -R u+rwX /home/nazwa_usera/NazwaKatalogu
```

Małe `x` zawsze dodaje prawo wykonania. Wielkie `X` dodaje je tylko katalogom oraz plikom, które już mają `x` dla kogokolwiek — czyli nie zamieni przypadkiem zwykłego pliku tekstowego w plik wykonywalny, ale zachowa przechodniość wszystkich katalogów w drzewie. To jest właściwe narzędzie do rekurencyjnej naprawy uprawnień całego drzewa katalogów, w przeciwieństwie do numerycznego `chmod -R 644` — patrz *Częste błędy*.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Dodanie prawa wykonania bez litery grupy: chmod +x

```
chmod +x NazwaPliku
```

Pominięcie litery (`u`, `g`, `o`, `a`) jest równoważne `a+x`, ale zmodyfikowane przez `umask` procesu — w praktyce w większości powłok efekt jest identyczny z `a+x` dla plików, które już istnieją. Szybki odruch, gdy interesuje nas tylko „niech się da uruchomić”, bez rozpisywania, komu dokładnie.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Skopiowanie uprawnień z innego pliku: chmod --reference

```
chmod --reference=wzorcowy.txt kopia.txt
```

Zamiast liczyć tryb ręcznie, `chmod` bierze dokładnie taki sam zestaw uprawnień, jaki ma `wzorcowy.txt`, i nakłada go na `kopia.txt`. Przydatne przy odtwarzaniu stanu po nieudanej rekurencyjnej zmianie.

chmod = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Zmiana właściciela: chown user

```
chown nazwa_usera NazwaPliku
```

Zmieniamy właściciela pliku na `nazwa_usera`. Grupa pozostaje bez zmian. Na większości systemów Linux tylko root może zmienić właściciela pliku na kogoś innego — patrz *Częste błędy*.

chown = change owner: zmień właściciela

ZMIENIA SYSTEM

Zmiana właściciela i grupy naraz: chown user:grupa

```
chown nazwa_usera:NazwaGrupy NazwaPliku
```

Jedna komenda zamiast dwóch osobnych wywołań `chown` i `chgrp`. Dwukropek rozdziela właściciela od grupy bez spacji.

chown = change owner: zmień właściciela

ZMIENIA SYSTEM

Zmiana samej grupy przez chown: chown :grupa

```
chown :NazwaGrupy NazwaPliku
```

Pominięcie nazwy przed dwukropkiem zmienia wyłącznie grupę, zostawiając właściciela bez zmian — funkcjonalnie to samo co `chgrp NazwaGrupy NazwaPliku`.

chown = change owner: zmień właściciela

ZMIENIA SYSTEM

Rekurencyjna zmiana właściciela i grupy: chown -R

```
chown -R www-data:www-data /var/www
```

Typowy krok po rozpakowaniu archiwum aplikacji webowej z uprawnieniami roota na cały drzewo katalogów — nadajemy w ten sposób wszystko kontu, na którym faktycznie działa serwer WWW.

WWW = World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS · **chown** = change owner: zmień właściciela

TYLKO ODCZYT

Zmiana samej grupy: chgrp

```
chgrp NazwaGrupy NazwaPliku
```

Odpowiednik `chown :NazwaGrupy`, osobna komenda z czasów, zanim `chown` nauczył się przyjmować dwukropek. Zwykły użytkownik może zmienić grupę pliku, którego jest właścicielem, ale tylko na grupę, do której sam należy — nie na dowolną grupę w systemie.

ZMIENIA SYSTEM

Warunkowa zmiana właściciela: chown --from

```
chown --from=root www-data /var/www/index.html
```

Zmieniamy właściciela tylko wtedy, gdy obecny właściciel zgadza się z podanym w `--from`. Bezpieczne narzędzie do masowej zmiany: jeśli ktoś już ręcznie poprawił część plików, `--from=root` ich nie ruszy, bo nie są już własnością roota.

chown = change owner: zmień właściciela

TYLKO ODCZYT

Wypisanie uprawnień w formacie do skryptów: stat -c

```
stat -c '%a %U %G' NazwaPliku
```

`ls -l` jest czytelne dla człowieka, ale niewygodne do parsowania w skrypcie. `stat -c` z własnym formatem zwraca dokładnie te pola, które są potrzebne — tu: tryb ósemkowy,

właściciel, grupa — bez konieczności wycinania kolumn z `ls`.

Przykład:

```
644 adm_user deweloperzy
```

TYLKO ODCZYT

Sprawdzenie całego łańcucha uprawnień do pliku: namei -l

```
namei -l SciezkaDoPliku
```

Pokazujemy uprawnienia każdego segmentu ścieżki, od korzenia w dół — dokładnie to narzędzie, które odpowiada na pytanie „mam `rwx` na pliku, a i tak dostaję `Permission denied` — który katalog po drodze mnie blokuje?”.

Przykład:

```
f: /var/www/html/index.php
drwxr-xr-x root     root     /
drwxr-xr-x root     root     var
drwxr-xr-x root     root     www
drwx------ root     root     html
-rw-r--r-- apache   apache   index.php
```

W tym przykładzie `html` ma `drwx------` — sam plik ma poprawne `644`, ale katalog nadrzędny blokuje wejście każdemu poza właścicielem.

TYLKO ODCZYT

Wszystkie flagi i formy zapisu

Forma	Przykład	Co robi
symboliczna, dodanie	`chmod u+x`	dodaje uprawnienie, reszta bez zmian
symboliczna, odjęcie	`chmod g-w`	odbiera uprawnienie, reszta bez zmian
symboliczna, przypisanie	`chmod o=`	ustawia dokładnie taki zestaw, nadpisując poprzedni

Forma	Przykład	Co robi
ósemkowa	`chmod 644`	pełny, bezwzględny zestaw dla u/g/o naraz
`-R`	`chmod -R`, `chown -R`	rekurencja po całym drzewie katalogów
`--reference`	`chmod --reference=wzór.txt`	kopiuje tryb z innego pliku
`--from`	`chown --from=root nazwa_usera NazwaPliku`	zmienia właściciela warunkowo
wielkie `X`	`chmod u+rwX`	`x` tylko tam, gdzie już jest (katalogi, istniejące programy)

Uważaj

- `chmod -R 644 NazwaKatalogu` odbiera prawo `x` również katalogom — drzewo przestaje być przechodnie. Do rekurencji używamy `u+rwX`, nie trybu numerycznego, gdy w drzewie są i pliki, i katalogi.
- Zwykły użytkownik nie zmieni właściciela pliku na kogoś innego przez `chown` — to prawo ma tylko root.
- `chmod 777` jako odruch naprawczy na `Permission denied` prawie zawsze maskuje inny, prawdziwy problem — patrz *Częste błędy*.

Z życia wzięte

□□ Z ŻYCIA WZIĘTE

W piątek o 15:40 Ola wgrywa nową wersję sklepu B2B na app01, a strona zaczyna zwracać błąd 500 zaraz po rozpakowaniu archiwum. Tomek uruchamia `namei -l` na ścieżce do pliku wejściowego i widzi, że katalog aplikacji dostał `drwx------` zamiast `755`, więc proces serwera nie może do niego wejść, choć sam plik ma poprawne prawa. Po `chmod 755` na katalogu strona wraca do działania bez restartu czegokolwiek.

Po co to istnieje

Pojęcie: uprawnienie i trójka rwx. Każdy plik i katalog przechowuje trzy niezależne bity uprawnień dla trzech odbiorców: właściciela (`u`), grupy (`g`) i pozostałych (`o`). Bity to `r`

> (read — odczyt), `w` (write — zapis) i `x` (execute — wykonanie dla pliku, wejście dla katalogu). Dziewięć bitów łącznie (trzy odbiorcy razy trzy uprawnienia) to cały model klasycznych uprawnień uniksowych — wszystko, co robią `chmod`, `chown` i `chgrp`, sprowadza się do ustawiania tych dziewięciu bitów i dwóch identyfikatorów właściciela.

Bez tego modelu każdy plik na wielożytkownikowym systemie byłby dostępny dla każdego — albo, w drugą stronę, każda operacja wymagałaby pytania administratora o zgodę. Uniksowy model uprawnień z 1969 roku pozwala rozstrzygnąć dostęp lokalnie, bez centralnej bazy reguł, kosztem pewnej sztywności: trzy kategorie odbiorców i tyle. Gdy trzeba dać dostęp jednej dodatkowej osobie bez zmiany grupy pliku, klasyczny model się kończy — od tego jest ACL, temat osobnej strony tego rozdziału.

Analogia

Pomyśl o **budynku z trzema poziomami kluczy**: kluczem głównym właściciela mieszkania, kluczem od klatki dla współlokatorów i klamką od drzwi wejściowych dla listonosza.

Właściciel mieszkania (`u`) ma klucz, który otwiera wszystko — może wejść, zostawić coś w środku i, jeśli to garaż, wjechać samochodem (`x`). Współlokatorzy (`g`) mają klucz do wspólnej klatki — mogą wejść, ale niekoniecznie do każdego pokoju, i niekoniecznie mogą tam zostawiać swoje rzeczy, jeśli właściciel im tego nie pozwolił (`w`). Listonosz (`o`) nie ma żadnego klucza — może najwyżej włożyć list przez szczelinę w drzwiach, jeśli właściciel taką szczelinę zostawił (`r` na skrzynce), ale do środka nie wejdzie.

Kluczowy, często pomijany szczegół tej analogii: klucz do klatki schodowej (`x` na katalogu) to zupełnie inna rzecz niż zgoda na przeczytanie tablicy ogłoszeń w holu (`r` na katalogu). Można mieć wgląd w tablicę ogłoszeń przez szybę, nie mając klucza do wejścia — czyli widzieć nazwy plików przez `ls`, nie mogąc wejść do środka katalogu i otworzyć żadnego z nich. To dokładnie tłumaczy najważniejszą pułapkę tej strony, opisaną niżej.

Jak to działa

Rozwiń: jak to działa pod spodem

Rozbicie `ls -l` na czynniki pierwsze:

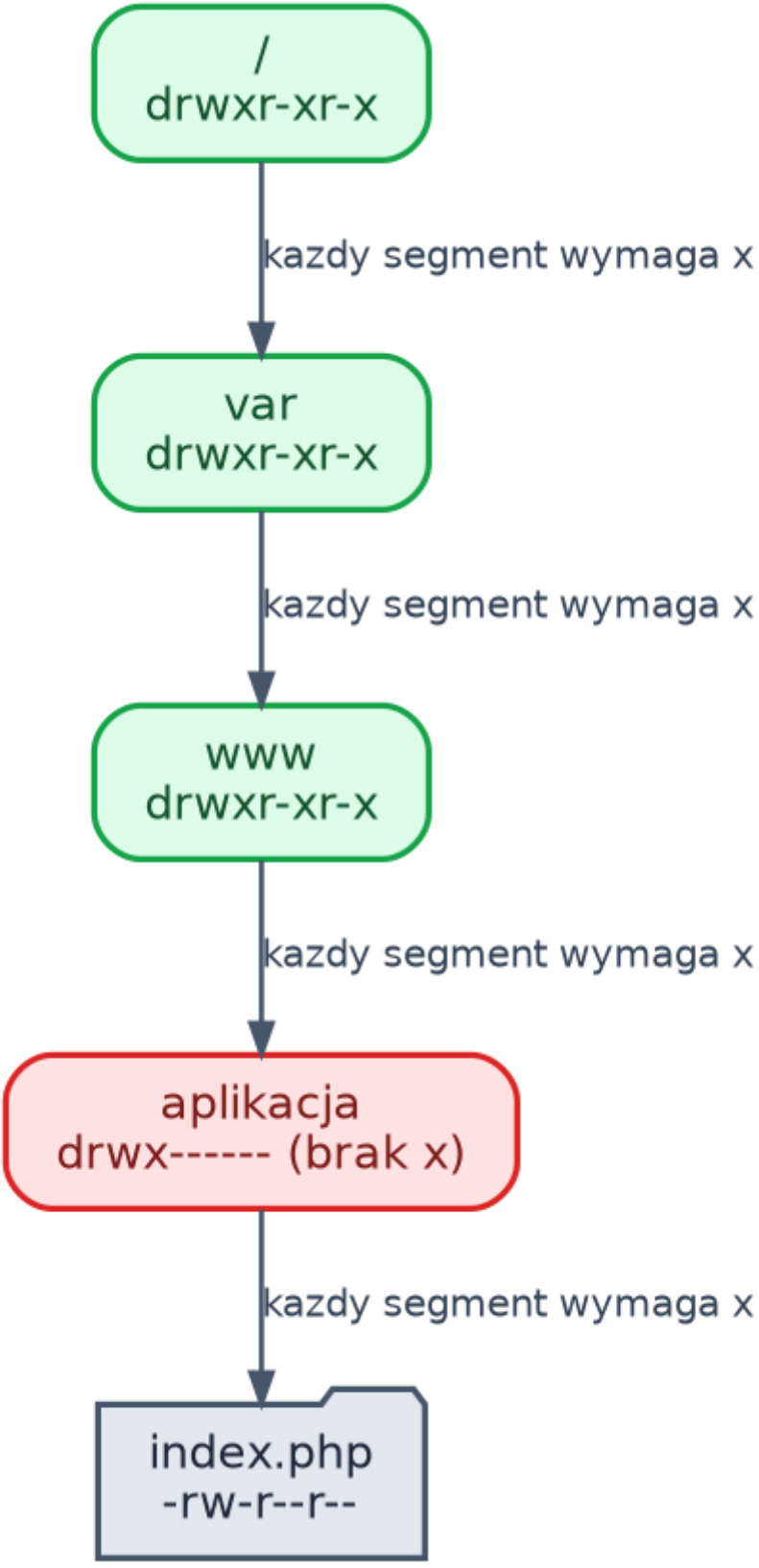
```
-rwxr-xr--
```

1. Pierwszy znak — typ pliku: `-` zwykły plik, `d` katalog, `l` link symboliczny, `c`/`b` urządzenie znakowe/blokowe.
2. Znaki 2–4 — uprawnienia właściciela: `rwx`.
3. Znaki 5–7 — uprawnienia grupy: `r-x`.
4. Znaki 8–10 — uprawnienia pozostałych: `r--`.

Zapis ósemkowy koduje to samo trzema cyframi, po jednej na odbiorcę. Każde uprawnienie ma stałą wartość liczbową — `r` = 4, `w` = 2, `x` = 1 — i sumuje się je dla danego odbiorcy:

Suma	Zapis	Znaczenie
7	`rwx`	4+2+1 — pełny dostęp
6	`rw-`	4+2
5	`r-x`	4+1
4	`r--`	tylko odczyt
0	`---`	brak jakichkolwiek praw

`755` to więc `rwx` dla właściciela, `r-x` dla grupy i pozostałych — dokładnie ten sam stan co `-rwxr-xr-x`.


/
drwxr-xr-x
kazdy segment wymaga x
var
drwxr-xr-x
kazdy segment wymaga x
www
drwxr-xr-x
kazdy segment wymaga x
aplikacja
drwx------ (brak x)
kazdy segment wymaga x
index.php
-rw-r--r--

Każdy segment ścieżki musi mieć prawo x, żeby dojść do pliku na końcu; brak x na jednym katalogu blokuje dostęp, nawet gdy sam plik ma poprawne uprawnienia (kliknięcie otwiera pełny rozmiar)

Co naprawdę znaczy `x` na katalogu. Na pliku `x` oznacza wprost prawo do uruchomienia go jako programu. Na katalogu `x` nie ma nic wspólnego z „wykonywaniem” — oznacza prawo **wejścia** do niego, czyli prawo, żeby użyć go jako części ścieżki: `cd katalog`, otworzyć plik `katalog/plik.txt`, sprawdzić metadane obiektu wewnątrz przez `stat`. `r` na katalogu to osobne, niezależne prawo — pozwala wylistować **nazwy** plików w środku (`ls katalog`), ale nic więcej.

Te dwa prawa działają niezależnie, co daje cztery kombinacje warte zapamiętania:

- `r-x` (typowe) — widzisz nazwy plików i możesz wejść do środka, żeby cokolwiek z nimi zrobić.
- `r--` (bez `x`) — widzisz listę nazw przez `ls`, ale `cd` do katalogu i otwarcie jakiegokolwiek pliku w środku kończy się `Permission denied` — nawet jeśli sam plik ma `rwxrwxrwx`. Katalog bez `x` jest **nieprzechodni** mimo `r`, dokładnie jak w analogii z tablicą ogłoszeń za szybą.
- `-wx` (bez `r`) — możesz wejść i otworzyć plik, jeśli **znasz jego dokładną nazwę**, ale `ls` katalogu nic nie pokaże. Rzadkie, ale spotykane w katalogach typu „drop box”, gdzie chcesz przyjmować pliki bez ujawniania, co już tam leży.
- `---` — katalog jest całkowicie zablokowany, niezależnie od uprawnień plików w środku.

To wyjaśnia też, dlaczego `namei -l` (patrz Fiszka) jest jedynym pewnym narzędziem diagnostycznym przy „mam dobre uprawnienia na pliku, a i tak dostaję odmowę” — problem prawie zawsze siedzi w jednym z katalogów **po drodze**, nie w samym pliku.

Krok po kroku

1. Zdiagnozuj, dlaczego serwer WWW nie widzi pliku.

```
namei -l /var/www/html/aplikacja/index.php
```

```
f: /var/www/html/aplikacja/index.php
drwxr-xr-x root  root  /
drwxr-xr-x root  root  var
drwxr-xr-x root  root  www
drwxr-xr-x root  root  html
drwx------ root  root  aplikacja
```

```
-rw-r--r-- apache apache index.php
```

Sam plik ma poprawne `644`, ale katalog `aplikacja` ma `700` i należy do `root` — proces `apache` nie ma prawa wejść do środka.

2. Napraw sam katalog, nie ruszając plików w innych miejscach.

```
chmod 755 /var/www/html/aplikacja
```

3. Sprawdź efekt i porównaj przed/po.

```
ls -ld /var/www/html/aplikacja
```

```
drwxr-xr-x. 3 root root 4096 wrz 17 10:02 /var/www/html/aplikacja
```

4. Jeśli problem dotyczy całego drzewa katalogów aplikacji naraz — rekurencja z wielkim `X`, nie z trybem numerycznym.

```
chmod -R u+rwX /var/www/html/aplikacja
chown -R www-data:www-data /var/www/html/aplikacja
```

To dwie osobne, uzupełniające się operacje: `chmod` naprawia przechodniość katalogów bez psucia plików, `chown` przenosi własność na konto, na którym faktycznie działa proces serwera.

Częste błędy

`chmod 777` jako odruch naprawczy.

```
php index.php
```

```
Warning: file_put_contents(): failed to open stream: Permission denied
```

```
chmod -R 777 /var/www/html
```

Działa — dopóki ktoś nie zapyta, dlaczego katalog produkcyjny jest zapisywalny przez każdego użytkownika systemu. `777` prawie nigdy nie jest prawdziwą naprawą, tylko wyłączeniem modelu uprawnień w miejscu, które go potrzebowało. Realny problem w tym przykładzie to zwykle zły właściciel katalogu (proces PHP działa jako `apache`, katalog należy do `root`) — naprawa to `chown`, nie rozdanie zapisu wszystkim.

Rekurencyjny tryb numeryczny psuje przechodniość katalogów.

```
chmod -R 644 /srv/dane
cd /srv/dane/archiwum
```

```
bash: cd: archiwum: Permission denied
```

`644` odbiera prawo `x` absolutnie wszystkiemu w drzewie, także katalogom — a katalog bez `x` jest nieprzechodni, patrz *Jak to działa*. Naprawa: `chmod -R u+rwX`, które dokłada `x` tylko katalogom i plikom, które już miały je dla kogoś.

Komunikat sshd o niezabezpieczonym kluczu prywatnym.

```
ssh -i klucz_prywatny.pem adm_user@srv01
```

```
Permissions 0644 for 'klucz_prywatny.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
load pubkey "klucz_prywatny.pem": bad permissions
```

Klient SSH odmawia użycia klucza prywatnego, jeśli grupa albo pozostali mają do niego jakikolwiek dostęp — nie ma znaczenia, że dane są zaszyfrowane. Naprawa jest jedną komendą:

```
chmod 600 klucz_prywatny.pem
```

Próba zmiany właściciela pliku bez uprawnień roota.

```
chown jkowalski raport.txt
```

```
chown: changing ownership of 'raport.txt': Operation not permitted
```

Na standardowym Linuksie tylko root może przenieść własność pliku na innego użytkownika — nawet właściciel pliku nie może „oddać” go komuś innemu przez zwykłego `chown` (to zabezpieczenie przed obchodzeniem limitów przestrzeni dyskowej per-user). Zmiana grupy na taką, do której sam należysz (`chgrp`), nie ma tego ograniczenia. Naprawa: `sudo chown jkowalski raport.txt`.

Zapomniana rekurencja przy zmianie właściciela po rozpakowaniu archiwum.

```
tar -xzf aplikacja.tar.gz -C /var/www/html
chown www-data:www-data /var/www/html/aplikacja
```

Zmieniony jest tylko sam katalog `aplikacja` — wszystkie pliki i podkatalogi w środku nadal należą do konta, które rozpakowywało archiwum (często do `root`, jeśli operacja szła przez `sudo`). Serwer

WWW startuje, ale nie może zapisać żadnego pliku w podkatalogach. Naprawa: `chown -R www-data:www-data /var/www/html/aplikacja`.

□□ Ćwiczenia

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

Ćwiczenie 1 — katalog bez `x`

```
cd /tmp
mkdir schowek
echo "sekret" > schowek/plik.txt
chmod 777 schowek/plik.txt
chmod 600 schowek
ls -l schowek/plik.txt
cat schowek/plik.txt
```

Plik ma `rwxrwxrwx` — teoretycznie każdy może go czytać. Dlaczego `cat` mimo to się nie udaje?

Odpowiedź

```
cat: schowek/plik.txt: Permission denied
```

Katalog `schowek` ma `600` — brak `x` blokuje wejście do niego kompletnie, niezależnie od tego, co ma sam plik w środku. Naprawa: `chmod 700 schowek` (albo `755`, jeśli inni mają móc wchodzić) — dopiero wtedy uprawnienia pliku zaczynają cokolwiek znaczyć.

Sprawdzenie przez `namei -l schowek/plik.txt` pokaże dokładnie, który segment ścieżki blokuje dostęp.

Ćwiczenie 2 — rekurencja numeryczna kontra wielkie `X`

```
cd /tmp
mkdir -p drzewo/podkatalog
touch drzewo/plik1.sh drzewo/podkatalog/plik2.sh
chmod -R 755 drzewo
ls -lR drzewo
```

Teraz porównaj z:

```
chmod -R 644 drzewo
cd drzewo/podkatalog 2>&1
cd /tmp
chmod -R u+rwX drzewo
cd drzewo/podkatalog && echo "działa"
```

Który wariant rekurencji zepsuł przechodniość katalogu, a który ją przywrócił bez nadawania niepotrzebnego `x` plikom, które go nie miały?

Odpowiedź

`chmod -R 644` odbiera `x` również katalogom — `cd drzewo/podkatalog` kończy się `Permission denied`. `chmod -R u+rwX` przywraca `x` katalogom (bo katalogi zawsze kwalifikują się do wielkiego `X`) i plikom, które miały `x` dla kogokolwiek przed operacją — nie dodaje go plikom, które nigdy nie były wykonywalne. To różnica między naprawą a przypadkowym nadaniem uprawnień, których nikt nie chciał.

Pytania kontrolne

1. Plik ma uprawnienia `-rw-r--r--`, a katalog, w którym leży, ma `d---------`. Czy zwykły użytkownik, który nie jest właścicielem żadnego z nich, przeczyta ten plik?

Odpowiedź

Nie. Uprawnienia pliku pozwalałyby na odczyt (`r--` dla pozostałych), ale katalog bez żadnych uprawnień (`---------`) blokuje samo wejście do niego — nie da się nawet odwołać się do ścieżki pliku wewnątrz. Uprawnienia katalogu i uprawnienia pliku działają niezależnie i oba muszą pozwalać na dany dostęp.

2. Czym różni się `chmod o+r plik` od `chmod o=r plik`, jeśli `plik` miał wcześniej `o` ustawione na `-wx`?

Odpowiedź

`o+r` dodaje `r` do istniejącego zestawu — wynik to `rwx` (bo `w` i `x` zostają). `o=r` przypisuje dokładnie `r`, kasując wszystko inne — wynik to `r--`, `w` i `x` znikają. `+`/`-` modyfikują częściowo, `=` ustawia bezwzględnie.

3. Dlaczego `chgrp` może się udać tam, gdzie `chown` na zmianę właściciela kończy się `Operation not permitted`?

Odpowiedź

Zwykły użytkownik może zmienić grupę pliku, którego jest właścicielem, ale tylko na grupę, do której sam należy — to nie jest przekazanie kontroli nad plikiem komuś innemu, tylko wybór spośród własnych uprawnień grupowych. Zmiana właściciela na inną osobę oddaje kontrolę nad plikiem, więc jest zarezerwowana dla roota.

4. Masz skrypt z uprawnieniami `-rwxr-xr-x`, ale w katalogu `bin`, który ma `drwxr-x---` i należy do grupy `deweloperzy`. Kto realnie może go uruchomić?

Odpowiedź

Tylko właściciel i członkowie grupy `deweloperzy` — bo mimo że sam plik ma `x` dla wszystkich (`r-x` dla pozostałych), katalog nadrzędny nie pozwala „pozostałym” w ogóle wejść (`---` dla `o` w `drwxr-x---`). Uprawnienia pliku bez odpowiadających uprawnień każdego katalogu po drodze nie wystarczą — patrz `namei -l` w Fiszce.

Powiązane: Użytkownicy, grupy i pliki, w których system ich trzyma · umask, setuid, setgid, sticky bit · ACL — uprawnienia ponad model właściciel-grupa-inni

Źródła: `man 1 chmod` · `man 1 chown` · `man 1 chgrp` · `man 1 stat` · `man 1 namei`

Revision #16
Created 2026-09-17 21:42:48 UTC by Tomasz Konieczny
Updated 2026-10-03 19:08:28 UTC by Tomasz Konieczny