Jak czytać dokumentację: man, --help, apropos, info, tldr

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

Odpowiedź na większość pytań o Linuksa leży już na dysku maszyny, przy której siedzisz. Ta strona uczy, gdzie dokładnie i jak to czytać, żeby nie musieć zgadywać flag ani ufać przypadkowemu wpisowi na forum sprzed ośmiu lat.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
EPEL	Extra Packages for Enterprise Linux: dodatkowe repozytorium pakietów dla RHEL i klonów
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL

Fiszka

TYLKO ODCZYT

Otworzenie strony podręcznika: `man`

```
man ls
```

Otwiera dokumentację narzędzia w pagerze. Pierwsza sekcja strony zawsze mówi, czym dana rzecz jest.

Przykład:

```
LS(1)                          User Commands                          LS(1)


NAME
       ls - list directory contents
```

man = manual: pokaż podręcznik systemowy do polecenia

TYLKO ODCZYT

Wybór konkretnej sekcji podręcznika: `man 5`

```
man 5 hosts
```

Numer przed nazwą wybiera konkretny tom dokumentacji. Sekcja 5 to formaty plików — tu akurat format `/etc/hosts`, a nie sposób wywołania programu.

Przykład:

```
HOSTS(5)                   File Formats Manual                HOSTS(5)


NAME
       hosts - static table lookup for hostnames


FORMAT
       IP_address canonical_hostname [aliases...]
```

man = manual: pokaż podręcznik systemowy do polecenia

TYLKO ODCZYT

Wyszukanie po opisie: `man -k`

```
man -k "disk usage"
```

Przeszukuje jednolinijkowe opisy wszystkich zainstalowanych stron, nie ich nazwy. Ratunek, gdy nie pamiętamy, jak narzędzie się nazywa.

Przykład:

```
du (1)               - estimate file space usage
ncdu (1)             - NCurses Disk Usage
duf (1)               - Disk Usage/Free Utility
```

man = manual: pokaż podręcznik systemowy do polecenia

TYLKO ODCZYT

Wyświetlenie jednolinijkowego opisu: `whatis`

```
whatis ls
```

Jedna linia z sekcją i opisem, bez otwierania pagera. Odpowiednik `man -f`.

Przykład:

```
ls (1)               - list directory contents
```

TYLKO ODCZYT

Wyświetlenie skróconej pomocy: `--help`

```
ls --help | head -n 6
```

Większość narzędzi GNU wypisuje zwięzłą listę flag prosto na standardowe wyjście — szybciej niż `man` i da się przefiltrować przez `grep`.

Przykład:

```
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
```

ls = list: wypisz zawartość katalogu

TYLKO ODCZYT

Wyświetlenie dokumentacji builtinów: `help`

```
help cd
```

Builtiny powłoki nie są plikami w `/usr/bin`, więc `man` ich nie widzi. Dokumentuje je wbudowany `help`.

Przykład:

```
cd: cd [-L|[-e][-P]] [dir]
    Change the shell working directory.
```

TYLKO ODCZYT

Otworzenie pełnej dokumentacji GNU: `info`

```
info coreutils 'ls invocation'
```

Projekt GNU traktuje strony man jako skrót; pełny opis bywa wielokrotnie dłuższy i jest dostępny przez `info`.

TYLKO ODCZYT

Wyświetlenie przykładów użycia: `tldr`

```
tldr systemctl
```

Kilka najczęstszych wywołań, bez formalnego opisu wszystkich flag. Do przypomnienia składni, nie do rozstrzygania szczegółów.

Przykład:

```
  systemctl


  Control the systemd system and service manager.


  - Start a service:
    sudo systemctl start service_name


  - Enable a service to start at boot:
```

```
sudo systemctl enable service_name
```

TYLKO ODCZYT

Nawigacja w pagerze

```
/wzorzec     szukaj w przód
n            następne trafienie
q            wyjście
```

`man` przepuszcza tekst przez `less`. Te trzy skróty załatwiają większość wizyt: szukamy flagi, przeskakujemy do kolejnego trafienia, wychodzimy.

> **Pojęcie: strona podręcznika (man page) i sekcje podręcznika.** Man page to dokumentacja jednej rzeczy — komendy, wywołania systemowego, formatu pliku — zainstalowana lokalnie razem z pakietem i dopasowana do jego wersji. Ponieważ ta sama nazwa bywa jednocześnie programem i formatem pliku konfiguracyjnego, strony są podzielone na ponumerowane sekcje: `man 5 crontab` i `man 1 crontab` to dwa różne teksty pod tą samą nazwą, i numer trzeba czasem podać wprost.

TYLKO ODCZYT

Wszystkie sekcje

Sekcja	Zawartość	Przykład
1	polecenia użytkownika	`man 1 crontab` — jak wywołać `crontab`
2	wywołania systemowe jądra	`man 2 open`
3	funkcje bibliotek (głównie libc)	`man 3 printf`
4	pliki specjalne w `/dev`	`man 4 random`
5	**formaty plików i konfiguracji**	`man 5 crontab` — składnia pięciu pól
6	gry	`man 6 fortune`
7	różne: konwencje, przeglądy, standardy	`man 7 signal`, `man 7 regex`
8	polecenia administracyjne	`man 8 mount`, `man 8 sshd`

Uważaj

- `man nazwa` bez numeru pokazuje **pierwszą** pasującą sekcję — zwykle 1. Konfiguracja tego samego narzędzia jest w 5 i możesz jej nigdy nie zobaczyć.
- `man -k` działa tylko przy aktualnym indeksie; na świeżej maszynie zwróci `nothing appropriate` dopóki nie uruchomisz `mandb`.
- Na obrazach kontenerowych RHEL/Rocky strony man są wycięte przy instalacji (`tsflags=nodocs`) — `man` będzie milczał, mimo że pakiet jest zainstalowany.

RHEL = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa

Z życia wzięte

Z ŻYCIA WZIĘTE

Czwartek, 10:15. Tomek nie pamięta, która flaga `find` usuwa puste katalogi, więc zamiast pytać na czacie, sprawdza `man find` i szuka po wzorcu przez `/empty`. Trzy minuty później ma `-empty` i uruchamia komendę na `app01`, zamiast czekać, aż ktoś odpowie na pytanie. Ola zauważa to i mówi, że to dokładnie ten nawyk, którego chciała nauczyć nowych ludzi w zespole.

Po co to istnieje

Potrzebujesz skasować pliki starsze niż 30 dni. Wpisujesz pytanie w wyszukiwarkę, trafiasz na wpis z 2014 roku, kopiujesz `find /var/log -mtime +30 -exec rm {} \;`. Działa. Nie wiesz, czy `+30` znaczy „starsze niż 30 dni" czy „starsze niż 31", nie wiesz, że istnieje `-delete`, i nie wiesz, że na tej wersji `find` jest `-execdir`, które nie ma podatności na wyścig przy zmianie katalogu.

Wszystko to jest w `man 1 find`, na maszynie, przy której siedzisz, w wersji dokładnie odpowiadającej binarce, którą za chwilę uruchomisz.

Dokumentacja lokalna ma nad internetem trzy przewagi, których nie da się nadrobić: opisuje **tę** wersję narzędzia, **tę** dystrybucję i jest dostępna, gdy serwer stoi w sieci bez wyjścia na zewnątrz. Umiejętność sprawnego czytania mana jest różnicą między administratorem, który rozwiązuje problem, a takim, który szuka kogoś, kto go już rozwiązał.

Analogia

Warsztat samochodowy ma trzy różne rzeczy, które potocznie nazywa się „dokumentacją".

Na ścianie wisi **schemat instalacji** — jedna kartka, wszystko naraz, do rzucenia okiem, gdy wiesz czego szukasz. To jest `--help`: lista flag, bez wyjaśnień, dla kogoś, kto zna narzędzie i chce sobie przypomnieć składnię.

Na półce stoi **instrukcja obsługi**: jak uruchomić, jak obsługiwać, co znaczą kontrolki, czego nie robić. To jest `man 1` — strona o tym, jak używać programu.

Obok niej leży **katalog części**: rysunki złączek, numery katalogowe, tabele momentów dokręcenia. Nikt tego nie czyta dla przyjemności, ale gdy potrzebujesz wiedzieć, jaki dokładnie format ma mieć uszczelka, to jedyne miejsce, gdzie ta informacja jest. To jest `man 5` — opis formatu plików, nie sposobu wywołania.

I tu leży pułapka, przez którą ludzie tracą godziny: **oba tomy mają na grzbiecie tę samą nazwę modelu**. Prosisz w magazynie o „dokumentację do crontaba" i dostajesz instrukcję obsługi, w której jest, jak otworzyć edytor i jak wylistować zadania — a Ty potrzebowałeś tabeli pięciu pól z katalogu części. Instrukcja jest prawidłowa, kompletna i całkowicie bezużyteczna dla Twojego pytania. Nie dlatego, że ktoś się pomylił, tylko dlatego, że nie podałeś numeru tomu.

Jak to działa

Rozwiń: jak to działa pod spodem

Anatomia strony man

Strony mają stałą kolejność sekcji. Znajomość tej kolejności jest ważniejsza niż czytanie od góry do dołu — po kilku razach skaczesz od razu tam, gdzie trzeba.

Sekcja	Co zawiera	Kiedy tam idziesz
NAME	nazwa i jedno zdanie opisu	sprawdzasz, czy to na pewno to narzędzie
SYNOPSIS	pełna składnia wywołania	90% przypadków — to jest to, po co przyszedłeś
DESCRIPTION	co program robi i jak	gdy zachowanie Cię zaskoczyło

Sekcja	Co zawiera	Kiedy tam idziesz
OPTIONS	każda flaga osobno	gdy `SYNOPSIS` nie wystarczył
EXIT STATUS	kody wyjścia i ich znaczenie	piszesz skrypt reagujący na błąd
ENVIRONMENT	zmienne, które program czyta	„dlaczego zachowuje się inaczej w cronie"
FILES	pliki konfiguracyjne i dane	szukasz, co edytować
EXAMPLES	gotowe wywołania	zawsze warto sprawdzić, czy istnieje
SEE ALSO	powiązane strony, z numerami sekcji	stąd trafiasz do `man 5`

Nie każda strona ma wszystkie sekcje. `EXAMPLES` bywa najbardziej wartościową częścią i jednocześnie tą, której ludzie nie sprawdzają — w `man 1 ss`, `man 1 tar` czy `man 8 firewall-cmd` jest gotowa odpowiedź na typowe zadanie.

SEE ALSO czyta się razem z numerami. Wpis `crontab(5)` nie jest ozdobnikiem — to instrukcja, żeby wpisać `man 5 crontab`.

Jak czytać SYNOPSIS

`SYNOPSIS` to formalny zapis. Ma własną, konsekwentną notację:

```
SYNOPSIS
       ls [OPTION]... [FILE]...

       tar [-] A --catenate --concatenate | c --create | d --diff --compare
           [OPTIONS] [FILE]...

       ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface] [-b bind_address]
           destination [command [argument ...]]
```

Zapis	Znaczenie
`[coś]`	**opcjonalne** — nawiasy kwadratowe nie są częścią komendy
`...`	można powtórzyć wiele razy

Zapis	Znaczenie
`a \	b`

alternatywa: albo a, albo b, nie oba

Zapis	Znaczenie
`{a\	b}`
`WERSALIKI` lub *kursywa*	miejsce na Twoją wartość, nie dosłowny tekst
tekst prosty	wpisujesz dokładnie tak

Z `ls [OPTION]... [FILE]...` odczytujesz: zero lub więcej opcji, potem zero lub więcej nazw plików. Oba fragmenty opcjonalne, więc samo `ls` jest poprawne.

Z `ssh ... destination [command [argument ...]]` odczytujesz: `destination` jest **obowiązkowy** (brak nawiasów), `command` opcjonalny, a argumenty można podać tylko wtedy, gdy podałeś komendę — bo ich nawias jest zagnieżdżony w nawiasie komendy.

Sekcje i dlaczego numer ma znaczenie

Ta sama nazwa może występować w kilku sekcjach, bo dotyczy różnych rzeczy.

```
man -f crontab
```

```
crontab (1)          - maintain crontab files for individual users
crontab (5)          - files used to schedule the execution of programs
```

`man crontab` bez numeru pokaże sekcję 1 — jak wywołać program `crontab`. Składnia pięciu pól (`* * * * *`), zmienne `MAILTO` i `PATH`, znaczenie `@reboot` są w sekcji **5**. To najczęstsza przyczyna zdania „w manie tego nie ma".

Inne pary, na które trafisz:

Nazwa	Sekcja 1 / 8	Sekcja 5
`crontab`	program do edycji zadań	format pliku z zadaniami
`passwd`	zmiana hasła	format `/etc/passwd`
`hosts`	—	format `/etc/hosts`
`sudo`	wywołanie `sudo` (1)	`man 5 sudoers` — reguły
`ssh`	klient (1)	`man 5 ssh_config`, `man 5 sshd_config`
`printf`	program `/usr/bin/printf` (1)	funkcja C (3)
`systemd`	menedżer (1)	`man 5 systemd.unit`, `man 5 systemd.exec`

Szybka metoda: `man -a nazwa` przechodzi przez wszystkie sekcje po kolei — po wyjściu z pierwszej `q` otwiera się następna.

Nawigacja: man to `less`

`man` nie wyświetla tekstu sam, tylko przepuszcza go przez pager — domyślnie `less`. Wszystkie skróty `less` działają:

Klawisz	Działanie
`Spacja` / `b`	strona w dół / w górę
`/wzorzec`	szukaj w przód
`?wzorzec`	szukaj wstecz
`n` / `N`	następne / poprzednie trafienie
`g` / `G`	początek / koniec dokumentu
`q`	wyjście
`h`	pomoc `less`

W praktyce najczęściej używasz jednego wzorca: szukania konkretnej flagi. Aby znaleźć opis `-z` w `man tar`, wpisz `/^\s*-z` i naciśnij Enter — kotwica `^` z białymi znakami trafia w linię definiującą flagę, a nie w każdą wzmiankę w tekście.

Pager zmienia się zmienną `MANPAGER`. Przykładowo `export MANPAGER='less -s -M +Gg'` pokazuje numer strony i procent postępu.

`--help` — szybsza alternatywa

Dla większości narzędzi GNU `--help` wypisuje zwięzłą listę flag prosto na standardowe wyjście:

```
ls --help | head -n 12
```

To jest szybsze niż `man` i — co ważniejsze — **da się przefiltrować**:

```
tar --help | grep -A2 'exclude'
ss --help 2>&1 | grep -i listen
```

Drugi przykład ma `2>&1`, bo część programów wypisuje pomoc na standardowe wyjście błędów. Dotyczy to zwłaszcza narzędzi, które przy braku argumentów drukują użycie i kończą się kodem różnym od zera.

Ograniczenie: `--help` nie jest standardem. Narzędzia BSD-owe i część starszych uniksowych używają `-h`, `-?` albo niczego. Dla `dd` czy `find` pomoc jest niekompletna i tam `man` jest jedyną drogą.

Builtiny powłoki nie mają stron man

```
man cd
```

```
No manual entry for cd
```

To nie błąd instalacji. `cd` nie jest programem w `/usr/bin` — jest wbudowane w basha. Musi być, bo zmiana katalogu w osobnym procesie nie miałaby żadnego efektu dla powłoki, która go uruchomiła. Dokumentację builtinów daje builtin `help`:

```
help cd
help test
help | head -n 20
```

Sprawdzenie, z czym masz do czynienia:

```
type -a cd echo test [
```

```
cd is a shell builtin
echo is a shell builtin
echo is /usr/bin/echo
test is a shell builtin
test is /usr/bin/test
[ is a shell builtin
[ is /usr/bin/[
```

`echo` i `test` istnieją w obu postaciach i **różnią się zachowaniem** — `man 1 echo` opisuje wersję z coreutils, a Twoja powłoka używa builtina. Stąd rozbieżności przy `echo -e` między bashem, `sh` i `/usr/bin/echo`. Gdy dokumentacja i rzeczywistość się nie zgadzają, sprawdź najpierw `type` .

`info` — pełna dokumentacja GNU

Projekt GNU traktuje strony man jako skrót, a pełną dokumentację publikuje w formacie `info`. Dla `coreutils`, `sed`, `awk`, `make` czy `gcc` różnica objętości jest wielokrotna.

```
info coreutils 'ls invocation'
info sed
```

Nawigacja jest inna niż w `less`: `n` następny węzeł, `p` poprzedni, `u` wyżej, `Enter` wejście w odnośnik, `q` wyjście. Na RHEL/Rocky pakiet nazywa się `info` (z `texinfo`), na Debianie i Ubuntu również `info`. Jeśli go nie ma, `man` odsyła do niego w sekcji `SEE ALSO`.

Praktyczny przykład różnicy: `man 1 date` wymienia formaty `%`, ale pełny opis względnych dat (`--date='next friday'`, `--date='2 days ago'`) jest dopiero w `info coreutils 'date invocation'`.

`tldr` i `cheat` — przykłady i nic więcej

Strony man opisują wszystko. Czasem chcesz zobaczyć trzy najczęstsze wywołania i nic poza tym.

```
tldr tar
```

```
tar

Archiving utility.

- Create an archive from files:
  tar cf target.tar file1 file2 file3

- Create a gzipped archive:
  tar czf target.tar.gz file1 file2 file3

- Extract an archive in a target directory:
  tar xf source.tar -C directory
```

Instalacja zależy od dystrybucji: na Debianie i Ubuntu `apt install tldr`, na RHEL/Rocky przez EPEL albo `pip install --user tldr`, uniwersalnie `npm install -g tldr`. Alternatywa `cheat` działa podobnie i pozwala trzymać własne ściągi w `~/.config/cheat/`.

Ograniczenie jest istotne: `tldr` to **wiki**, utrzymywane niezależnie od wersji narzędzia na Twojej maszynie. Do przypomnienia składni — świetne. Do rozstrzygnięcia, czy dana flaga istnieje w Twojej wersji — bezużyteczne.

`/usr/share/doc` i pliki pakietu

Pakiety często wnoszą dokumentację, której nie ma w manie: przykładowe konfiguracje, `CHANGELOG`, `README` z opisem migracji między wersjami.

```
ls /usr/share/doc/ | head
ls /usr/share/doc/httpd/
```

Żeby zobaczyć, co konkretny pakiet przyniósł:

```
# RHEL / Rocky / SLES
rpm -qd httpd                    # tylko pliki dokumentacji
rpm -ql httpd | grep /man/       # strony man z tego pakietu

# Debian / Ubuntu
dpkg -L apache2 | grep -E '/doc/|/man/'
```

Odwrotny kierunek — „do jakiego pakietu należy ta strona man" — bywa równie przydatny:

```
rpm -qf "$(man -w 5 crontab)"
dpkg -S "$(man -w 5 crontab)"
```

Na obrazach kontenerowych RHEL/Rocky dokumentacja jest wycinana: `/etc/dnf/dnf.conf` ma `tsflags=nodocs`. Jeśli potrzebujesz manów w kontenerze, zainstaluj pakiet z `--setopt=tsflags=''` i doinstaluj `man-db`.

Man online kontra man lokalny

Strony w internecie są wygodne, ale opisują jakąś wersję, niekoniecznie Twoją. `man 8 ip` na Debianie 12 i na RHEL 9 to inne teksty, bo `iproute2` jest w innej wersji. Podobnie `systemd` między wersją 239 (RHEL 8) a 252 (RHEL 9) zyskał opcje, których w starszej po prostu nie ma.

Objaw jest zawsze ten sam: kopiujesz flagę z internetu i dostajesz `unrecognized option`. Zanim uznasz, że coś jest zepsute, sprawdź wersję i lokalny man:

```
komenda --version
man -w komenda
```

Drugi problem to tłumaczenia. Jeśli masz `LANG=pl_PL.UTF-8` i zainstalowany pakiet `man-pages-pl`, `man` pokaże polską wersję — często przetłumaczoną kilkanaście lat temu i opisującą flagi, których już nie ma. Oryginał wymusisz przez locale:

```
LC_ALL=C man ls
man -L C ls
```

Krok po kroku

1. Nie znasz nazwy narzędzia — szukaj po opisie.

```
man -k "block device"
apropos -s 8 firewall
```

2. Znasz nazwę — sprawdź, w ilu sekcjach występuje.

```
man -f passwd
```

```
passwd (1)          - update user's authentication tokens
passwd (5)          - the password file
```

3. Wejdź w konkretną sekcję i skocz do tego, co Cię interesuje.

```
man 5 passwd
```

W pagerze: `/EXAMPLES`, potem `n` do kolejnych trafień.

4. Sprawdź `SEE ALSO` przed wyjściem.

Naciśnij `G` (koniec dokumentu). Tam jest lista powiązanych stron z numerami sekcji — najszybsza droga do konfiguracji narzędzia, którego dopiero co czytałeś opis.

5. Gdy to builtin, użyj `help`.

```
type -a ulimit && help ulimit
```

Częste błędy

Czytanie sekcji 1 zamiast 5 i wniosek, że „w manie tego nie ma".

Konfigurujesz zadanie cron. Otwierasz `man crontab`, przewijasz całą stronę i znajdujesz tylko opis flag `-e`, `-l`, `-r`. Składni pięciu pól nie ma. Wniosek: dokumentacja jest niekompletna, trzeba szukać w internecie.

Dokumentacja jest kompletna — czytałeś zły tom:

```
man 5 crontab
```

```
       The time and date fields are:

              field          allowed values
              -----          --------------
              minute         0-59
              hour           0-23
              day of month   1-31
              month          1-12 (or names, see below)
              day of week    0-7 (0 or 7 is Sunday, or use names)
```

Zapobieganie: jeśli szukasz **formatu pliku**, a nie sposobu wywołania programu, zaczynaj od sekcji 5. Jeśli nie wiesz, w której sekcji szukać — `man -f nazwa` albo `man -a nazwa`.

`man -k` zwraca `nothing appropriate` na świeżej maszynie.

```
man -k compress
```

```
compress: nothing appropriate.
```

`man -k` nie przeszukuje stron man — przeszukuje **indeks** zbudowany przez `mandb`. Na świeżo zainstalowanym systemie albo po ręcznym dograniu pakietów indeks może nie istnieć lub być nieaktualny. Naprawa:

```
sudo mandb
```

Na RHEL/Rocky indeks jest zwykle odświeżany zadaniem systemowym po instalacji pakietów, ale po `rpm -i` z pominięciem `dnf` albo w kontenerze trzeba to zrobić ręcznie. Osobny przypadek: pakiet `man-db` w ogóle nie jest zainstalowany — wtedy `man -k` nie istnieje, choć `man` może działać przez okrojoną implementację.

Szukanie mana dla builtina powłoki.

```
man cd
man export
man source
```

```
No manual entry for cd
```

Trzy komunikaty o braku strony pod rząd prowadzą do wniosku, że system jest źle zainstalowany. Żadna z tych komend nie jest programem — wszystkie są wbudowane w powłokę i dokumentuje je `help cd`, `help export`, `help source`. Cała lista: `help`. Pełny opis wszystkich builtinów jest też w `man 1 bash`, w sekcji `SHELL BUILTIN COMMANDS` — do znalezienia przez `/^SHELL BUILTIN` w pagerze.

Wyjątek, który myli: `man 1 test` i `man 1 printf` **istnieją**, bo są też programy o tych nazwach w `/usr/bin`. Opisują jednak wersję z coreutils, a powłoka użyje builtina. Jeśli Twój warunek `[ ... ]` zachowuje się inaczej, niż mówi man, to dlatego.

Skopiowanie nawiasów kwadratowych z SYNOPSIS.

```
tar [-czf] archiwum.tar.gz katalog/
```

```
tar: invalid option -- '['
Try 'tar --help' or 'tar --usage' for more information.
```

Nawiasy w `SYNOPSIS` oznaczają „opcjonalne" i nie są częścią składni. To samo dotyczy `|` (alternatywa) i `...` (powtórzenie). Poprawnie: `tar -czf archiwum.tar.gz katalog/`.

Flaga z internetu, której nie ma w Twojej wersji.

```
ss -tulpn --no-header
```

```
ss: unrecognized option '--no-header'
```

Wpis na blogu dotyczył nowszego `iproute2`. Zanim zaczniesz szukać obejścia, sprawdź, co masz lokalnie:

```
ss --version
man 8 ss | grep -n 'no-header'
```

Jeśli `grep` nic nie zwróci, opcja w tej wersji nie istnieje i żadna literówka tego nie zmieni. Ta sama zasada dotyczy `systemctl`, `podman`, `firewall-cmd` i wszystkiego, co szybko ewoluuje między wydaniami dystrybucji.

□□ Ćwiczenia

> Wszystko poniżej wykonujesz na własnej maszynie. Nic nie wymaga uprawnień administratora.

Ćwiczenie 1 — dwie sekcje jednej nazwy

```
man -f crontab
man -f passwd
man 5 crontab | head -n 40
```

Znajdź w sekcji 5 odpowiedź: co oznacza `@reboot` i w której kolumnie zapisuje się dzień tygodnia.

Ćwiczenie 2 — czytanie SYNOPSIS

```
man 1 cp | sed -n '/^SYNOPSIS/,/^DESCRIPTION/p'
```

Odpowiedź

```
SYNOPSIS
       cp [OPTION]... [-T] SOURCE DEST
       cp [OPTION]... SOURCE... DIRECTORY
       cp [OPTION]... -t DIRECTORY SOURCE...
```

Trzy warianty wywołania. W pierwszym `SOURCE` i `DEST` są obowiązkowe (brak nawiasów) i występują po jednym razie. W drugim `SOURCE...` może się powtórzyć, ale ostatnim argumentem musi być katalog. Trzeci wariant odwraca kolejność przez `-t` — i to jest ten, którego używa się z `xargs`, bo katalog docelowy nie musi być na końcu.

`[OPTION]...` w każdym wariancie znaczy: zero lub więcej flag, w dowolnej liczbie.

Ćwiczenie 3 — builtin czy program

```
cd /tmp
for k in cd echo printf kill time test ls; do
    printf '%-8s -> ' "$k"
    type -t "$k"
done
```

Odpowiedź

`cd` to `builtin`. `echo`, `printf`, `kill` i `test` to `builtin`, ale istnieją też jako pliki w `/usr/bin` — sprawdzisz przez `type -a echo`. `time` to `keyword` (słowo kluczowe powłoki, nie builtin), a `ls` to `file` albo `alias`, zależnie od tego, czy dystrybucja ustawia `alias ls='ls --color=auto'`.

Praktyczny wniosek: dla `builtin` i `keyword` dokumentacją jest `help`, nie `man`. Dla `alias` — `alias ls` pokaże, co się naprawdę uruchomi.

Ćwiczenie 4 — szukanie po opisie i sprawdzenie pakietu

```
sudo mandb >/dev/null 2>&1
man -k "file system" | head -n 10
```

```
man -w 8 mount
rpm -qf "$(man -w 8 mount)" 2>/dev/null || dpkg -S "$(man -w 8 mount)"
```

Pytania kontrolne

1. `man crontab` nie zawiera opisu pięciu pól czasowych. Dlaczego i gdzie on jest?

Odpowiedź

Bo `man` bez numeru pokazuje pierwszą pasującą sekcję — tutaj 1, czyli opis **programu** `crontab` i jego flag `-e`, `-l`, `-r`. Format pliku z zadaniami należy do innej kategorii dokumentacji: formaty plików to sekcja 5.

`man 5 crontab` zawiera tabelę pól, znaczenie `@reboot`, `@daily`, zasady dla `MAILTO` i `PATH`. `man -f crontab` pokazuje od razu obie sekcje, a `man -a crontab` przechodzi przez nie po kolei.

2. W SYNOPSIS widzisz `ssh [-46A] [-b bind_address] destination [command [argument ...]]`. Które elementy są obowiązkowe?

Odpowiedź

Tylko `destination`. Wszystko w nawiasach kwadratowych jest opcjonalne: flagi `-4`, `-6`, `-A`, opcja `-b` wraz z wartością, komenda i jej argumenty.

Zagnieżdżenie `[command [argument ...]]` niesie dodatkową informację: argumentów nie da się podać bez komendy, bo ich nawias leży w środku nawiasu komendy. `...` po `argument` oznacza, że argumentów może być wiele.

3. `man echo` opisuje flagę `-e`, ale w Twojej sesji `echo -e "a\tb"` zachowuje się inaczej niż w skrypcie z `#!/bin/sh`. Dlaczego man nie pomaga?

Odpowiedź

`man 1 echo` opisuje `/usr/bin/echo` z pakietu coreutils, a powłoka uruchamia własnego **builtina** o tej samej nazwie. Sprawdzisz to przez `type -a echo` — pokaże oba.

Builtin basha i builtin `dash` (czyli `/bin/sh` na Debianie i Ubuntu) obsługują `-e` inaczej: `dash` interpretuje sekwencje ucieczki domyślnie i nie zna `-e`. Dokumentacją builtina jest `help echo`, a dla przenośnych skryptów rozwiązaniem jest `printf '%s\t%s\n' a b`, które zachowuje się identycznie wszędzie.

4. Na nowym serwerze `man -k network` zwraca `nothing appropriate`, ale `man 8 ip` otwiera się bez problemu. Co jest nie tak?

Odpowiedź

Nic nie jest zepsute w samych stronach man. `man -k` (czyli `apropos`) nie przegląda stron, tylko indeks zbudowany przez `mandb`. Jeśli indeks nie został wygenerowany — bo system jest świeżo zainstalowany, bo pakiety wgrano z pominięciem menedżera pakietów albo bo to kontener — wyszukiwanie po opisie nic nie znajdzie, choć bezpośrednie otwarcie strony po nazwie działa.

Naprawa: `sudo mandb`. Przy pustym `/usr/share/man` na obrazach RHEL/Rocky przyczyną jest `tsflags=nodocs` w `/etc/dnf/dnf.conf` i pakiety trzeba przeinstalować z `--setopt=tsflags=''`.

Powiązane: Cudzysłowy, escapowanie i ekspansja · Zmienne środowiskowe i PATH · less i more — pagery · cron i systemd timers · Zarządzanie pakietami: dnf, apt, zypper

Źródła: `man 1 man` · `man 8 mandb` · `man 1 apropos` · `man 7 man-pages` · `help help` · `info info`

Revision #19
Created 2026-09-17 11:56:24 UTC by Tomasz Konieczny
Updated 2026-10-03 19:07:35 UTC by Tomasz Konieczny