Codzienna praca

Poruszanie, edycja, szukanie i zamiana.

- Poruszanie się po pliku
- Edycja: kasowanie, kopiowanie, wklejanie, cofanie
- Szukanie i zamiana

Poruszanie się po pliku

Poziom: podstawy **Dotyczy: Vim 9.x** **Czas: ~20 min**

W trybie normalnym kursor przesuwamy dziesiątkami różnych sposobów, każdy dopasowany do innej odległości: znak, słowo, linia, cały plik. Ta strona zbiera je wszystkie razem z mnemotechniką, skąd wzięła się każda nazwa.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
WMS	Warehouse Management System: system zarządzania magazynem
YAML	YAML Ain't Markup Language: format plików konfiguracyjnych oparty na wcięciach

Fiszka

TYLKO ODCZYT

Ruch o jeden znak: h j k l

```
h    j    k    l
```

Lewo, dół, góra, prawo. Układ pochodzi z klawiatury terminala ADM-3A z 1976 roku, gdzie nie było osobnych strzałek kursora, tylko producent narysował groty wprost na tych czterech literach. Strzałki na klawiaturze działają w Vimie tak samo, ale `hjkl` ma przewagę: nie trzeba przenosić dłoni znad liter, więc działa szybciej i nie gubi się nawet na słabym łączu SSH, które czasem gubi kody wysyłane przez strzałki.

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera

TYLKO ODCZYT

Ruch o słowo: w b e

```
w    b    e
```

`w` (word) skacze na początek następnego słowa, `b` (back) na początek poprzedniego, `e` (end) na koniec bieżącego słowa. Trzy litery pokrywają wszystkie kierunki poruszania się słowo po słowie bez konieczności liczenia znaków.

TYLKO ODCZYT

Początek i koniec linii: 0 ^ $

```
0    ^    $
```

`0` (zero) skacze na sam początek linii, łącznie z ewentualnymi spacjami wcięcia. `^` skacze na pierwszy niepusty znak w linii, czyli pomija wcięcie, co jest przydatne w plikach z dużymi wcięciami jak YAML. `$` (koniec, jak w wyrażeniach regularnych) skacze na koniec linii.

YAML = YAML Ain't Markup Language: format plików konfiguracyjnych oparty na wcięciach

TYLKO ODCZYT

Cały plik: gg G

```
gg    G
```

`gg` (podwojone g od go) skacze na pierwszą linię pliku, `G` (wielkie, od go to end) na ostatnią. Ten sam wzorzec liter działa też w `less`, opisanym na stronie Wiele plików: bufory, okna, karty.

TYLKO ODCZYT

Konkretna linia: :42 i 42G

```
:42
42G
```

Dwukropek z numerem linii albo numer linii z wielkim `G` skaczą bezpośrednio do wskazanej linii. To odruch, gdy komunikat błędu mówi, że problem jest w konkretnej linii pliku konfiguracyjnego.

Przykład:

```
syntax error at line 42
```

TYLKO ODCZYT

Cały ekran: Ctrl+f Ctrl+b

```
Ctrl+f    Ctrl+b
```

`Ctrl+f` (forward) przewija ekran o całą stronę w przód, `Ctrl+b` (backward) w tył. Szybsze niż wielokrotne `j` przy przeglądaniu długiego pliku logów.

TYLKO ODCZYT

Pasujący nawias: %

```
%
```

Kursor postawiony na jednym z nawiasów `( )`, `[ ]` albo `{ }` i naciśnięcie `%` skacze od razu na jego parę. Przydatne przy szukaniu, gdzie kończy się blok w pliku konfiguracyjnym z zagnieżdżonymi nawiasami klamrowymi, na przykład w definicji kontenera.

TYLKO ODCZYT

Znaczniki: ma i 'a

```
ma
'a
```

`ma` (mark) zapisuje bieżącą pozycję pod literą `a`, `'a` (apostrof plus ta sama litera) wraca do niej w dowolnym momencie później, nawet po wielu innych ruchach. Nazwa dowolna litera od `a` do `z`, przydatna przy pracy w dwóch odległych miejscach tego samego dużego pliku na przemian.

Z życia wzięte

□□ Z ŻYCIA WZIĘTE

Marek z magazynu zgłasza Tomkowi błąd w skrypcie startowym WMS na `ora01`, z numerem linii w komunikacie: `syntax error at line 87`. Tomek otwiera plik, wpisuje `:87`, ląduje dokładnie tam, gdzie trzeba, i naprawia literówkę w kilka sekund zamiast przewijać cały plik strzałką w dół.

Jak to działa

Pojęcie: ruch. W terminologii Vima ruchem nazywamy każdą komendę, która przesuwa kursor: `w`, `$`, `gg`, `%` i wszystkie pozostałe z tej strony. Ruch sam w sobie tylko przemieszcza kursor, ale w połączeniu z operatorem (`d`, `c`, `y`) staje się częścią większej komendy, jak `dw` czy `d$`, co pokazuje strona Edycja: kasowanie, kopiowanie, wklejanie, cofanie.

Częste błędy

Objaw	Przyczyna	Co robimy
Strzałki nie działają albo działają z opóźnieniem przez wolne SSH	terminal gubi albo opóźnia sekwencje kodów wysyłane przez strzałki	przechodzimy na `hjkl`, które są zwykłymi literami i zawsze docierają natychmiast
`0` nie ląduje tam, gdzie zaczyna się tekst	`0` skacze na sam początek linii, łącznie z wcięciem spacjami	używamy `^`, które pomija wcięcie i ląduje na pierwszym widocznym znaku
`:42` nic nie robi albo zgłasza błąd	plik ma mniej niż 42 linie	sprawdzamy liczbę linii przez `:$` (skok na ostatnią linię) albo `gg` i `G`

□□ Ćwiczenia

1. Jesteśmy w linii 60. pliku o długości 200 linii i musimy poprawić coś w linii 15. Jakiej komendy użyjemy, żeby skoczyć tam bezpośrednio, bez wielokrotnego naciskania `k`?

Odpowiedź

`:15` albo `15G`. Obie wpisane w trybie normalnym skaczą wprost na piętnastą linię pliku, niezależnie od tego, gdzie akurat stoi kursor.

1. Dlaczego `hjkl` bywa szybsze i pewniejsze niż strzałki, mimo że robią to samo?

Odpowiedź

`hjkl` to zwykłe litery leżące w rzędzie środkowym klawiatury, więc palce nie muszą przenosić się znad liter. Strzałki wymagają sięgnięcia dłonią w bok, a na wolnym albo niestabilnym łączu SSH kody sterujące wysyłane przez strzałki bywają opóźnione albo pogubione, podczas gdy zwykłe litery docierają zawsze.

Powiązane: Pierwsze kroki: otwarcie, zapis, wyjście · Edycja: kasowanie, kopiowanie, wklejanie, cofanie · Wiele plików: bufory, okna, karty

Źródła: `man 1 vim` · `:help motion.txt` w samym Vimie · catonmat.net, „Here is why vim uses hjkl keys as arrow keys"

Edycja: kasowanie, kopiowanie, wklejanie, cofanie

Poziom: podstawy **Dotyczy: Vim 9.x** **Czas: ~20 min**

Vim nie ma osobnego klawisza na każdą możliwą operację edycji. Zamiast tego łączy operator z ruchem, a te same kilka liter daje kombinatorycznie ogromny zestaw komend. Ta strona pokazuje tę gramatykę i najczęściej używane skróty, które z niej korzystają.

Fiszka

TYLKO ODCZYT

Kasowanie znaku: x

```
x
```

Kasuje jeden znak pod kursorem. Najprostsza operacja edycji, bez operatora i ruchu, przydatna do poprawiania literówek.

TYLKO ODCZYT

Kasowanie linii: dd

```
dd
3dd
```

`dd` kasuje całą bieżącą linię. Licznik przed komendą powtarza ją tyle razy, ile wskazuje liczba: `3dd` kasuje trzy linie naraz, licząc od bieżącej w dół.

TYLKO ODCZYT

Kopiowanie linii: yy

```
yy
```

`yy` (yank) kopiuje bieżącą linię do rejestru, bez jej kasowania. Działa razem z `p` i `P` poniżej.

TYLKO ODCZYT

Wklejanie: p P

```
p
P
```

`p` wkleja zawartość rejestru po bieżącej linii (albo po kursorze, przy wklejaniu w środku linii), `P` przed nią. Rejestr wypełnia poprzednie `dd` albo `yy`.

TYLKO ODCZYT

Cofanie i ponawianie: u Ctrl+r

```
u
Ctrl+r
```

`u` (undo) cofa ostatnią zmianę, można nacisnąć wielokrotnie, żeby cofnąć się o kilka kroków. `Ctrl+r` (redo) ponawia zmianę cofniętą przez `u`.

TYLKO ODCZYT

Powtórzenie ostatniej zmiany: .

```
.
```

Kropka powtarza ostatnią modyfikującą operację dokładnie w miejscu, w którym akurat stoi kursor. Przy powtarzalnej edycji w wielu miejscach pliku szybsze niż odtwarzanie całej sekwencji klawiszy za każdym razem.

🔒 TYLKO ODCZYT

Gramatyka: operator plus ruch

```
dw
d$
c$
dG
```

Operator sam w sobie nic nie robi, czeka na to, dokąd ma sięgnąć. `dw` znaczy usuń, przesuwając się jak `w`, czyli do początku następnego słowa. `d$` usuwa do końca linii, `c$` (change) usuwa do końca linii i od razu wchodzi w tryb wstawiania, `dG` usuwa od bieżącej linii do końca pliku.

Przykład:

```
3dw
```

Tabela operatorów i ruchów

Operator	Znaczenie	Przykładowy ruch	Wynik połączenia
`d`	usuń (delete)	`w`, `$`, `gg`, `G`	`dw` usuwa do końca słowa
`c`	zmień (change), usuwa i wchodzi w tryb wstawiania	`$`, `w`, `0`	`c$` zmienia resztę linii
`y`	kopiuj (yank)	`y`, `w`, `$`	`yy` kopiuje całą linię
`>`	zwiększ wcięcie	`>`, `G`	`>>` wcina bieżącą linię
`<`	zmniejsz wcięcie	`<`, `G`	`<<` cofa wcięcie bieżącej linii

Licznik przed całą komendą powtarza działanie: `3dw` usuwa trzy kolejne słowa, `2yy` kopiuje dwie linie.

Z życia wzięte

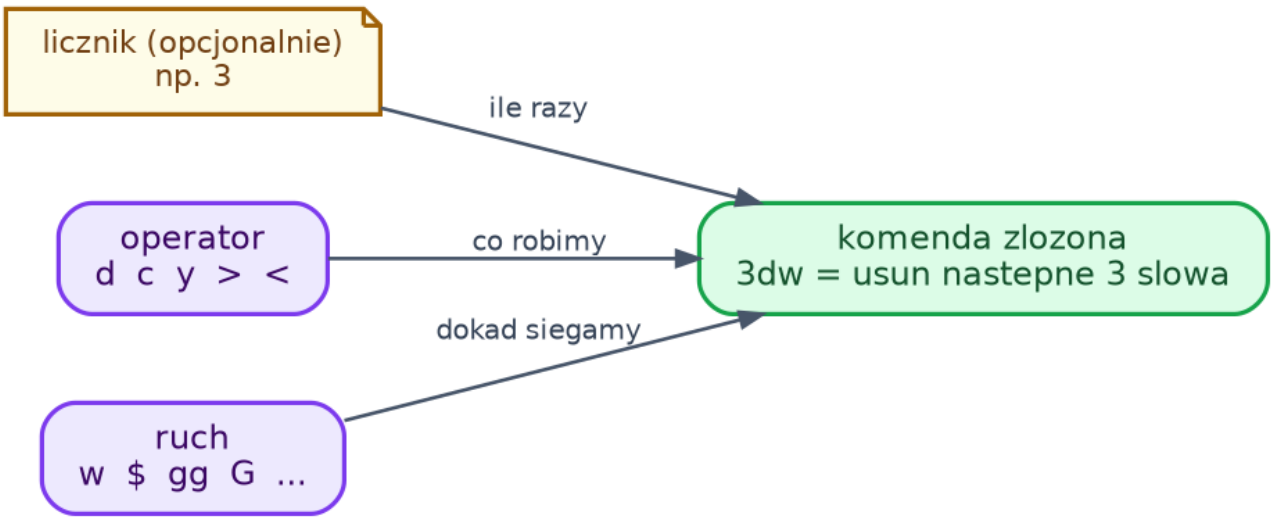
🔒 Z ŻYCIA WZIĘTE

> Ola prosi Tomka o usunięcie z pliku konfiguracyjnego pięciu kolejnych linii z przestarzałym ustawieniem. Tomek zamiast pięć razy naciskać `dd`, ustawia kursor na pierwszej z nich i wpisuje `5dd`, kasując wszystkie naraz jedną komendą.

Jak to działa

> **Pojęcie: bufor i rejestr.** Każda operacja kasowania (`dd`, `x`) i kopiowania (`yy`) odkłada usunięty albo skopiowany fragment do miejsca w pamięci edytora zwanego rejestrem. Domyślny, bezimienny rejestr działa jak schowek, z którego korzystają `p` i `P`. Kolejna operacja kasowania czy kopiowania nadpisuje ten sam rejestr: `dd` wykonane po `yy` bezpowrotnie zastępuje to, co skopiowaliśmy wcześniej. Do świadomego trzymania kilku fragmentów naraz służą rejestry nazwane literami, opisane na stronie Makra, rejestry, bloki i inne sztuczki.

Poniższy schemat pokazuje trzy elementy, z których składa się każda komenda edycji: opcjonalny licznik, operator i ruch.



Gramatyka Vima: licznik, operator i ruch łączą się w jedną komendę (kliknięcie otwiera pełny rozmiar)

Częste błędy

Objaw	Przyczyna	Co robimy
`dd` wykonane po `yy` wkleja nie to, co było skopiowane	oba operatory dzielą ten sam bezimienny rejestr, `dd` go nadpisało	do trzymania kilku fragmentów naraz używamy rejestrów nazwanych, patrz Makra, rejestry, bloki i inne sztuczki
Wpisanie `d` i nic się nie dzieje	`d` to operator, czeka na ruch, sam nic nie kasuje	dokańczamy komendę ruchem, na przykład `dw` albo `d$`, albo naciskamy `Esc`, żeby anulować
`.` powtarza coś innego, niż się spodziewaliśmy	`.` powtarza ostatnią modyfikującą operację, nie ostatnią komendę w ogóle, ruch bez zmiany tekstu się nie liczy	sprawdzamy, jaka była ostatnia rzeczywista zmiana tekstu przed użyciem `.`

Ćwiczenia

1. Chcemy usunąć cztery kolejne linie zaczynając od bieżącej, jedną komendą. Jak ją zapiszemy?

Odpowiedź

`4dd`. Licznik `4` przed `dd` powtarza kasowanie całej linii cztery razy, licząc od bieżącej w dół.

1. Na czym polega różnica między `dw` a `d$` wykonanymi w środku linii?

Odpowiedź

`dw` usuwa od kursora do początku następnego słowa, czyli fragment ograniczony ruchem `w`. `d$` usuwa od kursora do samego końca linii, niezależnie od tego, ile słów po drodze. Oba to ten sam operator `d`, różny tylko ruch, który do niego dołączymy.

Powiązane: Poruszanie się po pliku · Makra, rejestry, bloki i inne sztuczki · Szukanie i zamiana

Źródła: `man 1 vim` · `:help change.txt` w samym Vimie

Szukanie i zamiana

Poziom: podstawy **Dotyczy: Vim 9.x** **Czas: ~20 min**

Szukanie fragmentu tekstu i podmiana go w całym pliku to jedna z najczęstszych czynności administratora w plikach konfiguracyjnych. Vim robi to bez wychodzenia z edytora i bez otwierania osobnego narzędzia.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)

Fiszka

TYLKO ODCZYT

Szukanie w przód: /wzorzec

```
/localhost
```

`/` wchodzi w tryb szukania w dół pliku, wpisujemy szukany tekst i naciskamy Enter. Kursor skacze na pierwsze dopasowanie od bieżącej pozycji w dół, z zawijaniem na początek pliku, gdy dojdzie do końca.

TYLKO ODCZYT

Szukanie w tył: ?wzorzec

```
?localhost
```

`?` robi to samo, tylko szuka w górę pliku od bieżącej pozycji. Przydatne, gdy wiemy, że szukany fragment jest wcześniej w pliku, niż stoi kursor.

◫◫ TYLKO ODCZYT

Kolejne dopasowanie: n N

```
n
N
```

`n` powtarza ostatnie wyszukiwanie w tym samym kierunku, w jakim je rozpoczęliśmy, `N` w kierunku przeciwnym. Działa niezależnie od tego, czy szukaliśmy przez `/`, czy przez `?`.

◫◫ TYLKO ODCZYT

Szukanie słowa pod kursorem: *

```
*
#
```

`*` szuka w dół kolejnego wystąpienia słowa, na którym akurat stoi kursor, bez wpisywania go ręcznie. `#` szuka tego samego słowa w górę. Szybszy sposób na sprawdzenie, gdzie jeszcze w pliku pojawia się nazwa zmiennej czy usługi.

◫◫ TYLKO ODCZYT

Zamiana w jednej linii: :s

```
:s/stare/nowe/
```

`:s` (substitute) zamienia w bieżącej linii pierwsze dopasowanie `stare` na `nowe`. Bez `g` na końcu zamienia tylko jedno wystąpienie w linii, nawet jeśli jest ich więcej.

◫◫ TYLKO ODCZYT

Zamiana w całym pliku: :%s/stare/nowe/g

```
:%s/localhost/docker01/g
```

`%` na początku oznacza cały plik, `g` (global) na końcu oznacza wszystkie wystąpienia w każdej linii, nie tylko pierwsze. Bez `g` zamieni tylko jedno dopasowanie w każdej linii, co jest częstym powodem wrażenia, że „zamiana zadziałała tylko połowicznie".

TYLKO ODCZYT

Zamiana z potwierdzeniem: /gc

```
:%s/localhost/docker01/gc
```

Dokłada potwierdzenie przy każdym dopasowaniu: `y` zamienia, `n` pomija, `a` zamienia wszystkie pozostałe bez dalszych pytań, `q` przerywa. Bezpieczniejsze, gdy wzorzec może trafić w coś niezamierzonego.

Przykład:

```
replace with docker01 (y/n/a/q/l/^E/^Y)?
```

TYLKO ODCZYT

Zamiana w zakresie linii: :N,Ms/a/b/g

```
:10,40s/localhost/docker01/g
```

Ogranicza zamianę do linii od 10 do 40, przydatne w dużych plikach, gdzie ten sam wzorzec powtarza się w niepowiązanych sekcjach i nie chcemy dotknąć wszystkich naraz.

Z życia wzięte

Z ŻYCIA WZIĘTE

Ola migruje konfigurację testową z serwera `pg01` na nowy adres i musi podmienić wszystkie wystąpienia starego hostname w pliku `docker-compose.yaml`. Zamiast przeklikiwać każdą linię ręcznie, wpisuje `:%s/pg01-stary/pg01/gc` i przechodzi przez kilkanaście dopasowań, potwierdzając każde osobno, bo dwa z nich okazują się komentarzem, który ma zostać bez zmian.

Jak to działa

> **Pojęcie: wyrażenie regularne.** Wzorzec, który opisuje nie jeden konkretny tekst, tylko całą rodzinę pasujących do niego ciągów znaków, na przykład dowolny numer portu albo dowolny adres IP. Vim rozumie własną odmianę wyrażeń regularnych, zbliżoną do wariantu rozszerzonego. Pełne omówienie trzech odmian wyrażeń regularnych i ich różnic jest na stronie Wyrażenia regularne: BRE, ERE, PCRE, tu wystarczy znajomość zwykłego tekstu bez znaków specjalnych, jak w przykładach powyżej.

Częste błędy

Objaw	Przyczyna	Co robimy
Zamiana działa tylko w części linii, w których wystąpień jest kilka	brak flagi `g` na końcu, `:s` bez niej zamienia tylko pierwsze dopasowanie w linii	dopisujemy `g`: `:%s/stare/nowe/g`
Zamiana dotyka linii, których nie chcieliśmy ruszać	użyliśmy `%` (cały plik) zamiast zakresu	ograniczamy do konkretnych linii: `:10,40s/stare/nowe/g`
`/wzorzec` nie znajduje nic, mimo że tekst na pewno jest w pliku	literówka we wzorcu albo szukanie w złym kierunku (`/` zamiast `?`)	sprawdzamy pisownię, próbujemy `?` zamiast `/`

Ćwiczenia

1. Chcemy zamienić każde wystąpienie `10.20.10.42` na `10.20.10.43` w całym pliku, bez pytania o potwierdzenie. Jakiej komendy użyjemy?

Odpowiedź

`:%s/10.20.10.42/10.20.10.43/g`. `%` obejmuje cały plik, `g` zamienia wszystkie wystąpienia w każdej linii, nie tylko pierwsze.

1. Dlaczego warto czasem dopisać `c` do komendy zamiany, skoro wydłuża to pracę o potwierdzanie każdego dopasowania?

Odpowiedź

Bo wzorzec bywa szerszy, niż nam się wydaje, i trafia też w miejsca, których nie chcieliśmy zmieniać, na przykład komentarze albo fragmenty innych nazw zawierające ten sam ciąg znaków. Potwierdzenie każdego dopasowania pozwala pominąć te przypadkowe trafienia,

zamiast poprawiać je ręcznie po fakcie.

Powiązane: Edycja: kasowanie, kopiowanie, wklejanie, cofanie · Wyrażenia regularne: BRE, ERE, PCRE · Makra, rejestry, bloki i inne sztuczki

Źródła: `man 1 vim` · `:help pattern.txt`, `:help change.txt` w samym Vimie