Manual vim

Spis treści

Dokument zawiera skrót poleceń do Vim i oczywiście powstał w całości w tym edytorze :). Opisane są podstawowe polecenia w trybie poleceń, normalnym, wizualnym i inne. Jest to ekstrakt wiedzy z plików pomocy uzupełniony o wskazówki dotyczące "języka VIM".

Spis treści do pełnej pomocy dostępny jest po wykonaniu w Vim polecenia *:help usr_toc*.

Wbudowany w VIM tutorial, który prowadzi za rękę przez podstawy VIM można uruchomić w powłoce poleceniem: `$ vimtutor`, który jest też dostępny w języku polskim. Można go uruchomić poleceniem `$ vimtutor pl` i przećwiczyć podstawowe polecenia w programie.

A dla niecierpliwych chcących jak najszybciej edytor VIM opuścić - wystarczy wpisać *ZZ* lub *:x* i zatwierdzić przez <Enter>.

Autor: emkaminsk@gmail.com.

Wstęp

Vi powstał jako 'visual interface', wizualny edytor, nakładka na inny edytor: *ex*. *Ex* jest edytorem tekstu, który pierwotnie powstał dla obsługi teleksie, usłudze do przesyłania tekstu popularnej w latach 60-tych i 70-tych. Stąd tryb *ex* w vi i stąd pochodzą polecenia *':'*, *'$'*, *'^'*, *'%'*, oznaczanie zakresu przez *'a,b'*, *'g'*, *'s'*, *'m'* i inne.

Vim (VI iMproved) działa inaczej niż inne edytory tekstu. Podstawowe różnice to:

- ogromna konfigurowalność programu. Dostępne polecenia można łączyć w rozliczone kombinacje oraz tworzyć nowe polecenia dla przyspieszenia swojej pracy z tekstem,
- Vim dostarcza własny język poleceń, który pozwala pisać funkcje, skrypty i pluginy
- z poziomu Vim można wywoływać programy m. in. w Python do obróbki tekstu
- koncentracja na pracy z klawiaturą przy minimalizacji roli myszki do wybierania poleceń,
- rozróżnienie pomiędzy trybami pracy edytora, w których możliwe jest wydawanie różnych poleceń. Co więcej, sama edycja tekstu jest możliwa jedynie w trybie edycji, do którego można wejść wydając określone polecenia. To może stanowić barierę wejścia dla początkującego.

Dlatego należy rozróżnić kilka trybów działania Vim:

- **normalny** - polecenia składające się z jednego lub kilku znaków lub kombinacji klawiszy,
- **edycji** - tryb w którym można edytować tekst. Można go rozpoznać po tym, że w dolnej linii ekranu wyświetla się tekst --- INSERT ---,
- **poleceń**, tzw cmd-line, w którym wprowadza i wykonuje się tzw. polecenia ex. Są to polecenia rozpoczynające się od *:*, */*, *?* lub *!*,
- **wizualnej selekcji** (ang. visual mode),
- **wizualny** lub wizualnego zastępowania, w którym wpisywany tekst zastępuje tekst podświetlany (ang. select mode),
- **zastępowania** - tryb ten jest bardzo podobny do trybu edycji, ale każdy wpisany znak zastępuje znak na pozycji kursora.
- **ex**, który jest w istocie rozszerzonym trybem poleceń. Jest to tryb służący do pisania skryptów z poleceń stosowanych w trybie poleceń składających się z wielu linii.

W większości z tych trybów można poruszać się po tekstcie za pomocą strzałek. Do wychodzenia z różnych trybów do trybu normalnego służy klawisz *Esc*, wyjątkiem jest tu tryb ex, do wyjścia z niego należy wykonać komendę *:vi*.

Istnieje jeszcze jeden rzadko wspominany tryb pracy programu:

- **zawieszenia** w oczekiwaniu na polecenie; Vim wchodzi w ten tryb w trakcie wpisywania polecenia w trybie normalnym, pomiędzy operatorem a ruchem/obiektem (patrz Język Vim poniżej). Istnieje możliwość zdefioniowania mapowań klawiszy, które będą dedykowane do tego trybu. Więcej poniżej w sekcji 'Mapowania'.

Instalacja i wersje Vim

Oficjalne repozytorium kodu Vim znajduje się na GitHub. Na stronie repozytorium można znaleźć więcej opcji pobrania i instalacji Vim na różnych systemach.

Vim jest również dostępny jako plugin do innych popularnych IDE, np. VSCode, IntelliJ czy Pycharm.

Sekcja poniżej prezentuje wybrane opcje instalacji Vim w Windows oraz Linux.

Linux

`$ sudo apt install vim`	W dystrybucjach opartych na Debian (jak Ubuntu)

Windows

MobaXterm	Oprogramowanie z wieloma funkcjami, zastępujące okno poleceń i dostarczające wiele funkcji terminala Linux, w tym Vim. Do pobrania free ze strony https://mobaxterm.mobatek.net/download.html

Podstawowe polecenia

Uruchamianie Vim

`$ vim {nazwa_pliku}`	podstawowy sposób uruchamiania Vim - otwarcie do edycji pliku o podanej nazwie
`$ evim {nazwa_pliku}`	uruchamia vim w wersji easy (Easy VIM). W tej wersji Vim bardziej przypomina Notatnik w Windows z menu, ikonami i wyłączonymi trybami innymi niż tryb edycji
`$ gvim {nazwa_pliku}`	uruchamia vim w wersji GUI. Jest to pełny Vim z wszystkimi trybami i funkcjonalnościami, ale dodatkowo zaopatrzony w menu, skróty poleceń pod ikonami paska narzędzi
`$ view {nazwa_pliku}`	otwiera plik tylko do odczytu
`$ ex {nazwa_pliku}`	uruchamia Vim w trybie ex (bez wizualizacji treści pliku). Można przejść do Vim wykonując komendę *vi*.

`$ vimdiff {nazwa_pliku} {nazwa_pliku} {nazwa_pliku} {nazwa_pliku}`	uruchamia vim w wersji diff, która wyszukuje różnice pomiędzy plikami (w liczbie od 2 do 4), otwiera pliki w podzielonym pionowo oknie i podświetla te różnice
`$ vim -S {nazwa_sesji}`	uruchamia sesję Vim zapisaną poleceniem *mksession* (opis poniżej)
`$ vim -c /pattern {nazwa_pliku}`	otwiera plik i ustawia kursor na pierwszym wystąpieniu wzorca *pattern*

Pierwsze kroki w Vim

Sekcja prezentuje kilka podstawowych poleceń służących do zapisu, zamknięcia pliku i korzystania z wbudowanej pomocy.

:w 'plik'	zapis pliku, opcjonalnie można podać nazwę nowego pliku
:q	wyjście z programu
:saveas 'plik'	zapis bieżącego pliku pod nową nazwa 'plik'
:file 'plik'	zmiana nazwy obecnie edytowanego pliku
:e 'nazwa'	(edit) otwarcie następnego pliku 'nazwa' do edycji
ZZ	zatrzymuje program (tworzy plik swp z niezachowanymi zmianami!)
CTRL-Z	zawiesza Vim (zachowując wszystko - ustawienia, pliki itp., jak zminimalizowanie okna w Windows) i przenosi do powłoki shell. Do Vim można wrócić poleceniem `$ fg` w powłoce.
:help	otwarcie pomocy
:help 'word'	pomoc z danym słowem lub poleceniem; <*Tab*> lub *CTRL-D* wyświetla możliwe opcje
:help index	**spis wszystkich poleceń** we wszystkich trybach wraz z tagami, krótkim opisem i odnośnikiem do pełnej definicji
:help usr_toc.txt	wyświetla spis treści podręcznika Vim
:help usr_01.txt	otwarcie pierwszego rozdziału podręcznika
:help reference_toc	spis wszystkich plików pomocy Vim

.	powtórzenie wykonania poprzedniego polecenia wydanego w trybie normalnym.
CTRL-G	wyświetlenie podstawowych informacji: o nazwie pliku, numerze wiersza w którym jest kursor, procencie długości pliku od początku w którym stoi kursor

Opcje

W większości przypadków dodanie 'no' przed nazwą opcji wyłącza ją. Pełna lista opcji jest w pliku *options.txt*.

:set hls	włączenie podświetlania wyszukiwań
:nohlsearch	wyłączenie podświetlania
:set ic	włączenie trybu ignorecase wyszukiwania
:set number	włączenie numerów wierszy
:help 'hls'	pomoc na temat opcji hls
:set relativenumber	Włączenie trybu względnej numeracji. Numery linii wyswietlaja się względnie do pozycji kursora.
:set ve=all	Ustawia tryb virtualedit, w którym kursor może poruszać się gdziekolwiek (bez ograniczeń końcem linii itp.)

Język VIM

Język Vim to specyficzna składnia poleceń w trybie normalnym, której zrozumienie znacznie przyspieszy opanowanie edytora. Składa się z kombinacji działania i przedmiotu. Działaniem (ang. operator) jest akcja, np. *d* - delete (usuwanie), Przedmiotem może być:

- obiekt np. *w* - word (słowo) albo
- ruch (ang. motion), np. *j* (przejście linię w dół)
- funkcja.

Obiekt i ruch można poprzedzić liczbą. **Siłę Vim** stanowi fakt, że dowolna kombinacja działania, liczebnika i przedmiotu stanowi unikalne polecenie.

Więcej informacji (w tym np. bardziej kompletną listę operatorów i obiektów/ruchów) można znaleźć w pliku *motion.txt* (*:help motion*). Więcej informacji o funkcjach jest w *:help usr_41.txt*.

Akcje na obiektach większych niż jeden znak

Zazwyczaj służą do usuwania i zmiany tekstu. Więcej w *:help operator*

d	Delete - wycięcie tekstu i pozostanie w trybie normalnym. **Ważne** - usunięty tekst pozostaje w schowku do czasu kolejnej operacji usunięcia/wycięcia/skopiowania tekstu
c	Change - zmiana tekstu, czyli usunięcie go i przejście do trybu edycji
y	Yank - kopiowanie
>, <	Indent - wcięcie, unindent - zlikwidowanie wcięcia
g~, gu, gU	Zmiana dużych liter na małe (g~ wykonuje zamianę, gu - zmienia wszystkie litery na małe, gU - na duże).

Polecenia takie, jak np. *d*, *c* i *y* można poprzedzić oznaczeniem rejestru (np. *"x*), dzięki czemu wynik polecenia zostanie skopiowany do tego rejestru. Więcej - patrz Rejestry poniżej.

Ruchy (motion)

Definiują ruch do wykonania dla powyższych poleceń (np. *d* czy *y*)

$	Do końca linii
^, *0*	Do początku linii
+, -	Dwie linie: bieżąca i następna lub poprzednia
{a}G	Go - od kursora do linii o podanym numerze *a* lub (jeśli nie jest podana liczba) do końca pliku
f, F, t, T	Find - do wystąpienia kolejnego znaku, np. fa - do kolejnego 'a'. Kapitalik - szukanie wstecz. T = To - do znaku, ale bez uwzględnienia tego znaku.
h, j, k, l	Ruch w lewo, dół, górę i w prawo. *15j* Przykład ruchu - kolejne 15 linii. *10l* to kolejny przykład ruchu - 10 znaków w prawo.
H, L	Od pozycji kursora do góry ekranu lub do dołu ekranu

Warto podkreślić, że dwie poprzednie sekcje (czynności, ich krotności i ruchy) można składać w dowolne kombinacje, np. 9 czynności * 10 krotności * 10 ruchów daje w sumie 900 unikalnych poleceń, a to ułamek możliwości Vim.

Przykłady:

- *d0* - wycięcie całego tekstu od początku linii do kursora
- *y$* - skopiowanie tekstu od bieżącego miejsca kursora do końca linii
- *y2j* - kopiuje do schowka bieżącą linię oraz dwie poniżej

- *ctw* - change to 'w' - zmiana całego tekstu do następnego wystąpienia 'w'
- *dG* - wycina znaki od bieżącego wiersza do końca pliku
- *d20G* - wycina od bieżącego wiersza do wiersza 20

Obiekty tekstowe

Określenie obiektu tekstowego ma swoją składnię: pierwsza litera (opcjonalna) wskazuje:

- *i* - czy obiekt jest *exclusive*, tj. bez spacji przed i po,
- *a* - *inclusive* - włącznie ze spacjami przed i po. Użycie opcjonalnego określenia sprawia, że polecenie działa, gdy kursor jest gdziekolwiek w środku obiektu, niekoniecznie na jego początku.
- brak pierwszej litery oznacza, że działanie zostanie wykonane od miejsca kursora do końca obiektu.

Druga litera definiuje sam obiekt, np. *w* - słowo.

w	słowo (od kursora do końca słowa)
iw	Inner word - wewnątrz słowa; ogólnie *a* w *aw* oznacza obiekt razem ze spacjami dookoła niego, natomiast *i* w *iw* oznacza pominięcie spacji
aw	Word - słowo (razem ze spacjami)
e	do końca słowa (pozostawiając spację po słowie)
p	paragraph - akapit
s	sentence - zdanie
", ', _ `,),], _}, _b	wewnątrz cudzysłowiu, nawiasu, np. tagu HTML
i", _i', _i `	Inner quote - tekst otoczony cudzyslowiem danego typu bez cudzysłowi.
a), a], _a}, _ab	brackets - wewnątrz nawiasów (*tekst*) - włącznie ze znakami nawiasów. **Uwaga!** Znaczenie nawiasów tutaj się zmienia, bowiem użycie samych znaków *), }* ma inne znaczenie, niż gdy są poprzedzone przez *a* lub *i*.
i), i], _i}, _ib	Inner brackets - wewnątrz nawiasów (*tekst*) - bez znaków cudzysłowiu

Podobnie jak z ruchami, akcje i obiekty można dowolnie ze sobą łączyć produkując kolejne setki (jeśli nie tysiące) możliwych unikalnych poleceń.

Przykłady:

- *dw* - usunięcie znaków od kursora do końca słowa,

- *dap* - usunięcie całego akapitu od początku do końca razem z pustymi liniami,
- *yis* - skopiowanie całego zdania od początku do końca bez białych znaków z początku i końca zdania,
- *ci"* - zmiana całego tekstu pomiędzy znakami "

Inny przykład połączenia działania z funkcją:

- *d:call search("Następny")* - kasuje wszystko pomiędzy kursorem i następnym wystąpieniem słowa "Następny"

Działania (operatory) można też połączyć z wyszukiwaniem:

- *d/tekst* - usuwanie tekstu od pozycji kursora do kolejnego wystąpienia *tekst*
- *y?słowo* - kopiowanie tekstu od pozycji kursora wstecz do pierwszego wcześniejszego wystąpienia *słowo*

oraz ze znacznikami (ang. marks) - więcej o znacznikach poniżej. Przykład:

- *d'a* -usuwanie tekstu od bieżącej pozycji kursora do znacznika *a*

Tryb wizualny

Alternatywą dla składni operator - obiekt/ruch jest użycie trybu wizualnego. W tym trybie wpierw zaznacza się fragment tekstu, który zostaje podświetlony na ekranie, a potem wykonuje się akcję (operator).

v	Visually select - zaznaczenie tekstu do zmiany Więcej opcji wejścia w ten tryb opisanych jest poniżej.

Składnia polecenia złożonego

xay	składnia: operator - liczba - obiekt/ruch; a - liczba powtórzeń, x - operator, y - obiekt/ruch
raxby	na początku opcjonalnie można podać r - rejestr, do którego skopiowany zostanie wynik polecenia. Gdy podane są dwie liczby (*a* i *b*), polecenie wykonane zostanie *a*b* liczbę razy.

Działania na pojedynczych znakach

Mimo, że poniższe polecenia definiują czynności, nie przyjmują rzeczownika/obiektu do działania. Dlatego np. *'2x'* jest skończonym poleceniem (kasuje dwa znaki poczynając od znaku pod kursorem).

s	Substitute - zastępuje znak pod kursorem, po wykonaniu pozostaje w trybie wstawiania. Np. wpisując *3s* usuwamy trzy znaki licząc od kursora i możemy wpisać nowy tekst o dowolnej długości.
r	Replace - zastępuje jeden znak pod kursorem, po wykonaniu pozostaje w trybie normalnym
x	Cross out - usuwa pojedynczy znak pod kursorem, po wykonaniu pozostaje w trybie normalnym

Polecenia w trybie normalnym

Poruszanie kursorem

Więcej pomocy w *:help motion.txt*

Poruszanie się w ramach wiersza lub pojedynczych znaków

h, j, k, l	poruszanie w czterech kierunkach (lewo, dół, góra, prawo)
^, *0*	początek bieżącego wiersza, przy czym ^ oznacza pierwszy znak nie będący białą spacją, a *0* to rzeczywiście początek linii
$	koniec bieżącego wiersza
w	(word) początek następnego słowa, polecenie złożone
b	(before, beginning) początek poprzedniego słowa
e	(end) przejście na koniec słowa
%	(gdy kursor wskazuje otwierający nawias) przeskok do zamykającego nawiasu
(,)	przeskok do początku lub końca zdania, w którym stoi kursor
{n}|	przeskok do *n*-tej kolumny w wierszu

Poruszanie się pomiędzy wierszami w pliku

{a}G	(go) przeskok do a-tej linii pliku
G	przeskok do ostatniej linii pliku
gg	przeskok na początek pliku (to samo co *1G*)

+	przejście do początku kolejnego wiersza
-	przejście do początku poprzedniego wiersza
CTRL-E	(extra) wyświetlenie dodatkowej linii (jedna linia w dół)
CTRL-Y	jedna linia w górę
{, }	przeskok do początku lub końca akapitu, w którym stoi kursor

Przeskoki

CTRL-]	wejście w link (wspomaga nawigację po plikach pomocy Vim)
CTRL-O	(older) powrót do poprzedniego miejsca po przeskoku (np po przeskoku do innej linii lub po wejściu w link). Można wykonywać wiele razy.
CTRL-I	przejście do nowszej pozycji w odwrotnej kolejności jak *CTRL-O*
``	powrót po przeskoku (cofnięcie do poprzedniej pozycji).
:jumps	wyświetlenie listy przeskoków

Przenoszenie kursora na wyświetlanej stronie

H	(Home) przeniesienie kursora na górę strony
M	(Middle) przeniesienie kursora na środek strony
L	(Last) przeniesienie kursora na dół strony

Przeskok o stronę lub pół strony

CTRL-U	(up) przewinięcie o pół strony w górę
CTRL-D	(down) przewinięcie o pół strony w dół
CTRL-F	(forward) przewinięcie tekstu w przód o stronę
CTRL-B	(backward) przewinięcie tekstu w tył o stronę

Zmiana widoku

zz	wycentrowanie ekranu na wierszu, w którym jest kursor
zt	(top) ustawienie ekranu tak, że linia z kursorem jest na górze ekranu

zb	(bottom) jw. ale jest na dole

Zmiany tekstu

Wiele z opisywanych tu poleceń przyjmuje argument numeryczny z przodu. Np. *10i{tekst}* po naciśnięciu <Esc> powtarza wpisany tekst dziesięciokrotnie.

Więcej dostępnych poleceń służących do zmiany tekstu (usuwania, wstawiania, przesuwania, formatowania, sortowania) jest opisanych w pliku pomocy *:help change*.

Wchodzenie w tryb edycji

i	(insert) wstawienie znaku w bieżącym miejscu
I	wstawianie znaków na początku bieżącego wiersza
o	(open) rozpoczęcie następnego wiersza
O	rozpoczęcie nowego wiersza w bieżącym wierszu
a	(add) dodanie znaku zaraz za kursorem
A	wejście w edycję na końcu bieżącego wiersza
~	zmiana rozmiaru znaku (z dużego na mały i odwrotnie)

Usuwanie i wycinanie

x	usuwanie znaku pod kursorem
d	(delete) wycinanie wiersza lub jego części - przeniesienie do schowka, polecenie złożone
D	wycinanie wiersza od pozycji kursora do końca linii (to samo co *d$*)
J	(join) łączenie dwóch wierszy że sobą - bieżącego i następnego. Działając z argumentem numerycznym z przodu łączy podaną liczbę wierszy.

Kopiowanie

y	(yank) kopiowanie do schowka, polecenie złożone
Y	kopiowanie całego bieżącego wiersza (inaczej niż w przypadku *D* czy *C*, *Y* jest synonimem *yy*, a kopiowanie do końca linii osiągamy tylko przez *y$*.

yy	to samo, działa jak Y
y$	kopiowanie od bieżącego znaku do końca linii
yl	kopiowanie znaku pod kursorem (lub określonej liczby znaków, gdy polecenie jest poprzedzone liczbą.

Wklejanie

Więcej w pomocy - *:help insert.txt*

p	(put) wklejanie zawartości schowka za kursorem lub poniżej bieżącego wiersza
P	wklejenie przed/powyżej bieżącego wiersza

Korekta

r	(replace), np. ra zastępuje bieżący znak pod kursorem przez literę 'a' i wraca d trybu poleceń
R	korekta wielu znaków (każdy napisany znak zastępuje znak pod kursorem). W tym trybie Backspace przywraca zmiany
c	(change) wycięcie tekstu i wejście w tryb edycji, polecenie złożone (składnia jak d czy y)
C	analogicznie do *D*, to polecenie działa jak *c$* - wycinanie do końca linii z jednoczesnym wejściem w tryb edycji
ce	(change to the end) pozwala skorygować bieżące słowo (usuwa je do końca i wchodzi w tryb edycji)
s	(substitute) zamień znak pod kursorem (tożsame z *cl*)
S	kasuje całą bieżącą linię i wchodzi w tryb edycji (tożsame z *cc*)
xp	zamiana kolejności dwóch liter (poprawa szwedzkiego błędu)
@{a}	wykonywanie korekty za pomocą sekwencji klawiszy zapisanej w makrze. Patrz sekcja 'Makra'

Modyfikatory poleceń złożonych

Patrz też sekcja 'Język VIM' powyżej. Poniżej zamieszczone są inne przykłady poleceń służących do

wycinania: *d* oraz kopiowania: *y*).

Niektóre z poniższych przykładów działają liniowo (ang. linewise), w odróżnieniu od wcześniejszych poleceń, które działały znakowo (ang. characterwise). Oznacza to, że działają na cały wiersz lub wiersze, a nie na fragmenty wierszy.

Poniższe przykłady oparto na poleceniu usuwania *d*, ale działają analogicznie z *c*, *y* i innymi operatorami.

dd	usunięcie (wycięcie) całego wiersza, 2dd - wycinanie dwóch całych linii (d2d działa tak samo)
dl	wycinanie jednego znaku pod kursorem (to samo co *x*)
dh	wycinanie jednego znaku przed kursorem (to samo co *X*)
dw	wycinanie całego słowa. Tak samo działa *W*, np. *d2W* - wycinanie dwóch słów. Powoduje wycinanie całego słowa (od miejsca kursora) włącznie z białym znakiem (bez usuwania pierwszego znaku następnego słowa).

Niewielka różnica między *w* i *W* dotyczy znaków niealfanumerycznych w środku słowa. Różnicę można zauważyć ćwicząc *dw* i *dW* na słowie 'doesn't'.

daw	(A Word) wycinanie całego słowa (niezależnie w którym miejscu jest kursor) włącznie z białym znakiem na końcu
diw	(Inner Word) wycinanie słowa bez usuwania białych znaków
das	(A Sentence) wycinanie całego zdania
dis	(Inner Sentence) wycinanie całego zdania
dap	(A paragraph) usunięcie całego akapitu
de	wycinanie całego słowa (od kursora do ostatniego znaku) pozostawiając białe znaki
d$	wycinanie od bieżącego miejsca do końca wiersza
*d^*	wycinanie od pierwszego znaku nie będącego białym znakiem w bieżącym wierszu do bieżącego znaku
d0	od początku wiersza do bieżącego znaku

Cofanie zmian

u	(undo) cofa ostatnią zmianę w pliku
U	przywraca linie do oryginalnego stanu
CTRL-R	(redo) cofa zmiany włącznie z undo (można wycofać się z undo)

Szukanie

Sekcja prezentuje różne polecenia, zarówno w trybie normalnym jak i poleceń cmd-line.

f	(find) wyszukanie pojedynczego znaku w bieżącym wierszu po bieżącym miejscu. Odmiany tego polecenia opisane wyżej to *F*, *t* i *T*.
;	Gdy klawisz ten zostaje wciśnięty po wyszukiwaniu za pomocą *f*, *F*, *t* i *T*, uruchamia następne wyszukanie w bieżącym wierszu
/word	wyszukanie w pliku kolejnego wzorca wyrażenia regularnego (wystąpienia 'word')
?word	wyszukanie w pliku poprzedniego wzorca
*	wyszukiwanie do przodu słowa, na którym właśnie stoi kursor
#	wyszukiwanie w tył słowa, na którym stoi kursor
n	(next) wyszukanie wzorca w pliku w tym samym kierunku (dla */* w przód, dla *?* w tył)
N	wyszukanie wzorca w pliku w przeciwnym kierunku co poprzednie wyszukiwanie
CTRL-O	powrót do miejsca gdzie rozpoczęte zostało wyszukiwanie
CTRL-I	przejście do przodu
/word/b+1	wyszukuje *'word'* i umieszcza kursor na drugiej pozycji od początku. Oprócz *'b'* można używać też innych poleceń: *'e'*(pozycja od końca), cyfra oznacza liczbę linii po znalezionym słowie
:help pattern.txt	więcej pomocy na temat wyszukiwania za pomocą wyrażeń regularnych (Perl). Można też uzych :help usr_27.txt

Modyfikatory poleceń w trybie normalnym

!!date	wstawia datę w bieżącym wierszu
!command	wykonanie polecenia w zewnętrznym programie, a dokładnie filtrowanie określonego zakresu wierszy przez ten program, np. *!5Gsort* sortuje linie od bieżącej do 5 (zauważmy, że po wpisaniu *!5G* polecenie to jest konwertowane do trybu cmd-line: *:.,/+3!*).

Polecenia ex

Aby znaleźć pomoc o danym poleceniu wystarczy wpisać *:help {polecenie}* podając całą jego nazwę lub tylko początek. Można użyć *CTRL-D* lub <*Tab*> aby wyświetlić opcje autouzupelniania. Ponowne naciśnięcie <*Tab*> lub *CTRL-P* pozwala przewijać pomiędzy opcjami.

Po wpisaniu *:*:

- klawisze <*Up*> i <*Down*> scrollują historię komend.
- Za pomocą symbolu `||` można wykonać więcej niż dwie komendy na raz, np. `:w \| !ls` zapisuje plik i wyświetla zawartość bieżącego katalogu.
- Kombinacja *CTRL-F* otwiera okno poleceń, w którym widać np. historię poprzednich poleceń. Można z niego wyjść wpisując *:quit* lub wciskając *CTRL-C*.

Szczegółowa pomoc o poleceniach ex, tj. o trybie poleceń, edytowaniu, zakresach działania, flagach, znakach specjalnych znajduje się w pliku *:help cmdline.txt*.

Listę wszystkich poleceń ex można znaleźć w *:help ex-cmd-index*.

Podstawowe polecenia

:m	(move) przesunięcie wierszy
:d	(delete) usunięcie wierszy
:co, *:t*	(copy) skopiowanie wierszy
:p	(print) wydrukowanie u dołu ekranu wiersza
:760,780m20	przesunięcie wierszy od 760 do 780 do pozycji wiersza 20
:m+1	przesunięcie bieżącego wiersza o jedną linię w dół.
:m10	przesunięcie bieżącego wiersza do wiersza o numerze 10.
:p5	wypisanie bieżącego i czterech kolejnych wierszy na dole ekranu

:w	zapisanie zmian w bieżącym buforze do bieżącego pliku
:10,15w nowy_plik	zapisanie linijek od 10 do 15 w nowym pliku
:10,15w >> plik	doklejenie linijek od 10 do 15 do pliku *plik*
:r plik	odczytanie zawartosci pliku *plik* i wstawienie go począwszy od linii pod kursorem
:{polecenie}!	modyfikator polecenia wymuszający jego wykonanie bez zapisania zmian, np. *:q!* - wyjście z programu z porzuceniem zmian
:!{polecenie}	wykonuje polecenie z powłoki. Przykład: *:!ls -al*

Zastępowanie

Dokładny opis w *:help substitute*. Opcje modyfikujące proces zamiany takie jak *g* czy *c* opisane są w *:help s_flags*

:[range]s[ubstitute]/from/to/[flags]	Ogólną składnia polecenia do zamiany tekstu 'from' na 'to'.
:s/b/A	(substitute) zamiana b na A w bieżącym wierszu (jeden raz)
:s/a/A/g	zamiana a na A w bieżącym wierszu (wszystkie wystąpienia)
:%s/a/A/g	zamiana w całym pliku
:%s/a/A/gc	zamiana w całym pliku z potwierdzeniem każdej zamiany. Wciskając *y* lub *n* można potwierdzić lub odrzucić daną zmianę, *a* potwierdza wszystkie późniejsze zmiany, *q* i *l* kończy procees zamian.
:5,10s/a/A/g	zamiana w wierszach od 5 do 10
:.,$s/a/A/g	zamiana w wierszach od bieżącego do końca pliku
5:s/a/A/g	zamiana w pięciu wierszach licząc od bieżącego

Różne polecenia w ex

:!python %	wykonanie otwartego obecnie skryptu w python. Symbol % symbolizuje nazwę aktywnego pliku.
:r 'nazwa'	(read) odczytanie pliku 'nazwa' i wstawienie go w bieżącym pliku w pozycji kursora
:read !ls lub *r! {polecenie}*	wczytuje wynik polecenia ls do bieżącego pliku

:history	wyświetlenie historii komend w trybie cmd-line
q:	otwiera okno poleceń
:history /	wyświetlenie historii wyszukiwań
:browse oldfiles	wyświetla listę uprzednio edytowanych plików, można podać numer i otworzyć wybrany plik do edycji

Wielokrotne wykonywanie poleceń

Więcej pomocy w *:help usr_26.txt* oraz *:help multi-repeat*

.	powtórzenie ostatniej operacji edycji/kasowania/korekty. **Bardzo** przydatna funkcją do szybkiego ponowienia tej samej komendy skrótem.
:[rangę][:]g[lobal]/{pattern}/{command}	(global) wyszukanie wzorca pattern i wykonanie w każdym pasującym wierszu polecenia command.

W powyższym:

- znak *:* pomiędzy zakresem i słowem global jest opcjonalny,
- *Command* to polecenie w trybie cmd-line. Wydając polecenie *normal* można zdefiniować polecenie w trybie normalnym.
- różne przykłady *range* zostały omówione w sekcji dot. zastępowania (np. % - cały plik),
- *Pattern* to wyrażenie regularne w odpowiednim dla wersji Vim standardzie (np. POSIX). Więcej: *:help pattern*.

*:2,20g/txt/normal 0i**	Przykład multi-polecenia. W wierszach od 2 do 20 wyszukiwany jest tekst 'txt'. W wierszach, gdzie został on znaleziony wykonywane jest polecenie 0i* - co powoduje wstawienie na początku wiersza znaku '*'.
:g/^/m 0	^ pasuje do każdego wiersza w pliku a 'm' przesuwa wiersz na początek pliku. W efekcie następuje odwrócenie kolejności wierszy w całym pliku.
:g/^$/d	usunięcie wszystkich pustych linii z pliku

Wizualna selekcja

W tym trybie wiele poleceń zyskuje nowe znaczenie.

Wejście w tryb wizualnej selekcji

v	wchodzi w tryb selekcji, po zaznaczeniu tekstu można na nim wykonać polecenie zwykłe lub polecenie ex (np *:w* nazwa zapisze fragment w pliku nazwa)
V	tryb selekcji, można zaznaczać całe linie
CTRL-V	tryb blokowy, w którym zaznacza się prostokątny obszar

Edycja w trybie wizualnej selekcji

o/O	(other) w trybie wizualnej selekcji powzwala na przejście kursorem na drugi koniec zaznaczonego obszaru
I{tekst}	w trybie blokowym polecenie pozwala na wstawienie przed blokiem w każdym wierszu tego samego tekstu
c{tekst}	w trybie blokowym polecenie pozwala na wstawienie zamiast bloku w każdym wierszu tego samego tekstu
A{tekst}	w trybie blokowym polecenie pozwala na wstawienie za blokiem w każdym wierszu tego samego tekstu
~	zamiana małych liter na duże i odwrotnie
r{a}	zamiana każdej litery na {a}

Polecenia w trybie edycji (tryb Insert)

W tym trybie można wykonać wiele poleceń za pomocą klawiszy funkcyjnych lub skrótów z CTRL.

CTRL-Left	przeskok o całe słowo w lewo (tak samo działa z Shift, w prawo z drugą strzałką)
CTRL-Home	przeskok na początek pliku
CTRL-End	przeskok na koniec pliku
CTRL-P	autouzupelnianie (Vim zgaduje resztę słowa na podstawie innych wpisanych w pliku słów oraz innych plików)
CTRL-N	autouzupelnianie, ale Vim szuka słów z przodu Inne

CTRL-X CTRL-F	autouzupelnienie nazwami plików. Inne opcje autouzupelniania są w helpie *usr_24.txt*
CTRL-X CTRL-L	autouzupelnianie całymi liniami
CTRL-A	powtórzenie ostatniej edycji w trybie Insert. Dobry skrót, aby wykonać tę samą modyfikację w wielu miejscach. *CTRL-2* (lub *CTRL-@*) wykonuje to samo i jednocześnie wychodzi z trybu edycji.
CTRL-Y	kopiuje znak powyżej kursora
CTRL-W	usunięcie ostatniego napisanego słowa (słowa tuż przed kursorem)
CTRL-U	usunięcie całego wiersza od początku do miejsca kursora
CTRL-V{znaki}	pozwala na wstawienie znaków specjalnych. Liczba trzycyfrową (od 000 do 255) pozwala na wpisanie znaków ascii. Wpisując *'x'* możemy podać liczbę w układzie szesnastkowym (np. *CTRL-Vxff* - bez spacji w środku) a *'o'* - ósemkowym. Podając u lub U możemy wstawić znak Unicode.
CTRL-K{znaki}	wstawienie symboli. Lista symboli jest dostępna komenda :digraphs. Np. *CTRL-KC** produkuje Ξ (ponownie - bez spacji w środku). Inny przykład to *CTRL-KCo* ©.
:CTRL-O{polecenie}	pozwala na wykonanie w trybie edycji jednego polecenia z trybu normalnego (bez wychodzenia z trybu edycji).

Inne polecenia w różnych trybach

Wybrane polecenia rozpoczynające się od *'g'*

Polecenia te najczęściej modyfikują znaczenie komendy wymienionej po *g*. Szczegółową lista jest w pliku *index.txt*.

ga	wyświetla wartość ASCII znaku pod kursorem
g8	wyświetla wartość hex znaku UTF-8 pod kursorem
gm	przeskoczenie kursorem na środek ekranu
gM	przeskoczenie kursorem na środek bieżącej linii

{N}{"x}gp	(put) wstawienie N razy tekstu że schowka (lub rejestru *{x}*, jeśli podany)

Formatowanie tekstu

:set textwidth={x}	ustawienie szerokości linii. Jeśli nowe słowo spowoduje że linia będzie dłuższa niż maksimum, zostanie wstawiony znak nowej linii
gqap	uporządkowanie akapitu tak aby w każdej linii znalazło się maksimum słów względem dostępnej szerokości linii
gq}	jw.
:{zakres}center {szerokość}	wyśrodkowanie tekstu w liniach opisanych zakresem. {szerokość} opisuje szerokość linii użyta do wyśrodkowania
:{zakres}right {szerokość}	jw. ale dosunięcie tekstu do prawej
:{zakres}left {margines}	jw ale dosunięcie do lewej. Margines określa liczbę spacji po lewej stronie tekstu.
:8,15le4	przykład jak wciąć tekst z 4 spacjami na początku każdego wiersza od nr 8 do 15

Znaczniki (ang. Marks)

Oznaczenia pozwalające definiować zakres pliku lub miejsca do których można przeskoczyć. Znaczniki nie są widzialne, są tylko pozycjami w pliku. Znaczniki i rejestry nie są przechowywane w tym samym miejscu, można mieć jednocześnie znacznik *'a* i rejestr *"a* bowiem są czymś innym

m{t}	wstawienie znacznika *t* w bieżącej pozycji. Znaczniki można nazywać małymi lub dużymi literami
'{t}	przejście do pierwszego znaku linii, w której jest znacznik *t*
`{t}	przejście do właściwego znaku, na którym jest znacznik *t*
:marks	wyświetla listę aktywnych znaków, przede wszystkich znaków globalnych (o numerach 0-9), które są tworzone przy każdym wyjściu z Vim
'0	przejście do miejsca, gdzie ostatnio Vim został zamknięty
:delm {marks}	usunięcie znacznika

'<, '>	początek i koniec zakresu wizualnej selekcji

Rejestry

Pozwalają zapisać fragment tekstu do przeklejenia lub wykonać ten tekst jako polecenie.

Więcej w pliku pomocy change.txt - *:help registers*

Rejestry użytkownika

Rejestry użytkownika oznaczane są małymi literami. Użycie dużej litery pozwala na doklejenie kolejnego tekstu do istniejącego rejestru.

"{x}	Użyj rejestru *'x'* do następnej czynności (takiej jak *'d'*, *'y'* lub *'p'*)
"{x}y{ruch/obiekt}	skopiuj do rejestru *'x'* wynik następnego ruchu; np. *"ay$* kopiuje do rejestru *'a'* tekst do końca linii; w przypadku podania obiektu czynność wykonana jest na tym obiekcie
"ayas	skopiowanie całego bieżącego zdania do rejestru *'a'*
"ap	wklejenie zawartości rejestru *'a'* w bieżącym miesjcu
CTRL-R{x}	w trybie edycji wstawienie zawartości rejestru *'x'*

Rejestry wbudowane

"{cyfra}	Rejestry oznaczone cyframi od 1 do 9 zawierają ostatnich dziewięć schowków. Np. *"3p"* wstawia do linii poniżej kursora trzecią ostatnio skopiowaną linię tekstu.

Makra

Umożliwia zapisanie sekwencji poleceń i wykonanie ich wielokrotnie, przez np. 10@a

q{a}{sekw}q	zapisanie sekwencji klawiszy *'sekw'* w rejestrze 'a'
@{a}	wykonanie sekwencji klawiszy zapisanej w rejestrze. Sekwencja będzie wykonana w trybie poleceń
@@	wykonanie poprzedniej sekwencji klawiszy

Skróty

Więcej w pliku *help usr_24.txt*

:iabbrev {skrót} {tekst}	pozwala zdefiniować skrót. Wpisanie w tekście (w trybie edycji) skrótu i potem spacji spowoduje zastąpienie skrótu tekstem. Tekst może mieć jedno lub wiele słów. Jeśli na początku lub na końcu tekstu ma być spacja należy zdefiniować ją jako .
:iab {skrót} {tekst}	krótsza forma polecenia do definiwania skrótu.
:abbreviate	wyświetla listę zdefiniowanych skrótów

Okna

:[v]split 'plik'	podzielenie bieżącego okna na dwa; podając opcjonalna nazwę pliku można otworzyć inny plik w drugim oknie; *[v]* pozwala na podział pionowy
:[v]new	otwarcie nowego pustego okna
CTRL-W w	przejście do innego okna
CTRL-W hjkl	przechodzenie między oknami (lewo, dół, góra, prawo)
CTRL-W J	przemieszczenie obecnego okna w dół (tak samo pozostałe klawisze - *H*, *K* i *L*)
{x}CTRL-W +/-	zwiększenie/zmniejszenie wysokości bieżącego okna. Parametr {x} określa o ile linii następuje zmiana
:[vertical] resize {x}	zmiana rozmiaru bieżącego okna o wartość {x} (w pikselach). Słowo vertical pozwala na pionową zmianę
:close	zamknięcie bieżącego okna
:only	zamknięcie wszystkich okien z wyjątkiem bieżącego
:qall	całkowite wyjście z Vim; analogicznie *:wqall*

Bufory. Praca z plikami

:open 'plik'	otwarcie do nowego bufora pliku o nazwie *'plik'*
:buffers	wyświetlenie listy otwartych plików (tak samo działa *:ls*)

:buffer {x}	przełączenie się do pliku o numerze *{x}*; działa też skrót *'b'* lub *'bu'*
:bdel {x}	usunięcie z pamięci bufora o numerze *{x}*
:bn	(buffer next) edycja następnego otwartego pliku (w pętli)
:bp	(buffer previous) edycja poprzedniego otwartego pliku

Sesje

:mksession 'nazwa'	utworzenie sesji o nazwie 'nazwa'; opcjonalnie można użyć skrótu 'mk' zamiast 'mksession'
:mksession! 'nazwa'	nadpisanie sesji o nazwie 'nazwa'
:source 'nazwa'	wczytanie sesji o nazwie 'nazwa'

Zakładki (tabs)

Pomoc na temat zakładek - *:help tabpage.txt*

:tabe	(edit) tworzenie nowej zakładki (inaczej tabnew)
:tabc	(close) zamknięcie aktualnej zakładki
:tabn	(next) przejście do następnej zakładki
:tab {polecenie}	wykonuje polecenie w nowej zakładce (np otwarcie pliku pomocy poleceniem help)
:tab split	otwiera nową zakładkę z tym samym plikiem co bieżący
{x}gt	przejście do kolejnej zakładki; opcjonalnie x to numer zakładki
gT	przejście do poprzedniej zakładki

Mapowania klawiszy

Vim daje możliwość definiowania mapowań w wielu trybach pracy. Więcej w pliku **:help map.txt**

:map	Polecenie bez argumentu wyświetla wszystkie mapowania w trybach: normalnym, wizualnym i zawieszenia. Z jednym argumetem wyświetla mapowanie dla tego klawisza. Z dwoma tworzy nowe mapowanie.
:unmap	usuwa dane mapowanie

Inne polecenia służące do listowania i poprawiania istniejących mapowań są wymienione w pomocy.

Poniżej polecenia służące do tworzenia nowych mapowań. Ogólna składnia to *:map {klawisz} {sekwencja}* - to polecenie pozwala zmapować klawisz na sekwencję poleceń.

:nmap {k} {s}	mapowanie w trybie normalnym
:vmap {k} {s}	mapowanie w trybie wizualnej selekcji i zastępowania. *:smap {k} {s}* i *:xmap {k} {s}* pozwalają zdefiniowaniać mapowania osobno w tych dwóch trybach.
:omap {k} {s}	mapowanie w trybie zawieszenia po wprowadzeniu operatora
:imap {k} {s}	mapowanie w trybie wstawiania
:cmap {k} {s}	mapowanie w trybie wiersza poleceń

Przeglądarka plików

:edit .	otwiera zawatosc bieżącego katalogu w oknie
:Explore 'folder'	włączenie przeglądarki określonego katalogu, w tym katalogów sieciowych (ftp)
:split ~/	dzieli okno na dwa: przeglądarkę plików i puste okno
P	podgląd wybranego pliku w drugim oknie
o	horyzontalny podział okien i otwarcie pliku
v	otwarcie pliku w nowym vertykalnym oknie
t	otwarcie pliku w nowej zakładce
_ _	otwarcie pliku
CTRL-O	powrót do poprzedniej zawartości okna
s	zmiana sposobu sortowania
i	zmiana sposobu wyświetlania plików
r	odwrócenie kolejności sortowania

Przydatne pluginy

Vim posiada wiele pluginów, które rozszerzają jego funkcjonalności.

Surround

Plugin Surround pozwala w łatwy sposób dodawać znaki okalające dany tekst, np. tagi html,

pogrubienia w Markdown czy inne oznaczenia w podobnych formatach (np. AsciiDoc).

Instalacja i krótkie wprowadzenie do Surround zawarte są na stronie GitHub: https://github.com/tpope/vim-surround

cs{aktualny}{nowy}	polecenie wydane, gdy kursor stoi wewnątrz aktualnych znaczników zamieni znacznik na nowy. Np. dla tekstu "Lorem ipsum" polecenie *cs"'* zamieni go na 'Lorem ipsum'
ysaw{znacznik}	wklejenie *znacznika* w obiekt *aw*. Polecenie działa analogicznie dla innych obiektów/ruchów.
ds{znacznik}	usunięcie *znacznika* okalającego pozycję kursora
v{zaznaczenie}S<b>	po zaznaczeniu tekstu w trybie wizualnej selekcji kombinacja *S{znacznik}* wklei otwierający i zamykający znacznik przed i po zaznaczeniu

Inne zasoby o Vim

Główna strona projektu Vim	https://vim.org
Vim as language	https://www.youtube.com/watch?v=wlR5gYd6um0
You need to grok vi	https://stackoverflow.com/questions/1218390/what-is-your-most-productive-shortcut-with-vim/1220118#1220118
Definitive guide to text objects	https://blog.carbonfive.com/vim-text-objects-the-definitive-guide/