Poznajemy Vima

Opowieść, pierwsze kroki i tryby.

- Vim: edytor, z którego nie da się wyjść
- Pierwsze kroki: otwarcie, zapis, wyjście
- Tryby: dlaczego klawisze robią różne rzeczy

Vim: edytor, z którego nie da się wyjść

Poziom: podstawy **Dotyczy: Vim 9.x** **Czas: ~10 min**

Ta strona otwiera książkę o Vimie: skąd wziął się edytor, który każdemu początkującemu wydaje się złośliwy, dlaczego Zielona nie może się bez niego obejść i co czeka na kolejnych stronach.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera

Skąd się wzięło

> **Pojęcie: Vim.** Edytor tekstu uruchamiany w terminalu, bez myszki i bez okienek, w którym te same klawisze robią różne rzeczy w zależności od trybu pracy. Nazwa to skrót od Vi IMproved, czyli ulepszony `vi`.

Historia zaczyna się dużo wcześniej niż sam Vim. W 1969 roku Ken Thompson w laboratoriach Bell Labs napisał `ed`, pierwszy edytor tekstu dla systemu Unix. Nie miał ekranu w dzisiejszym rozumieniu, bo terminale tamtych czasów potrafiły tylko wypisywać kolejne linie na taśmie albo powoli odświeżanym ekranie, więc `ed` pracował linia po linii, bez podglądu całego pliku naraz.

W 1976 roku Bill Joy, student na Uniwersytecie Kalifornijskim w Berkeley, rozbudował `ed` do `ex`, a z niego wydzielił tryb pełnoekranowy `vi` (visual). Pracował na terminalu ADM-3A, urządzeniu bez strzałek kursora: klawiatura miała tylko literki, a producent narysował groty strzałek wprost na klawiszach `h`, `j`, `k`, `l`, bo akurat tam były wolne miejsca obok liter. Bill Joy przejął ten sam układ do ruchu kursora w `vi`, i tak zostało do dziś. Ten sam terminal tłumaczy też, dlaczego klawisz <kbd>Esc</kbd> wraca zawsze do trybu poleceń: na ADM-3A leżał w lewym górnym rogu, tam gdzie dziś mamy Tab, więc był pod ręką częściej niż jakikolwiek inny klawisz specjalny.

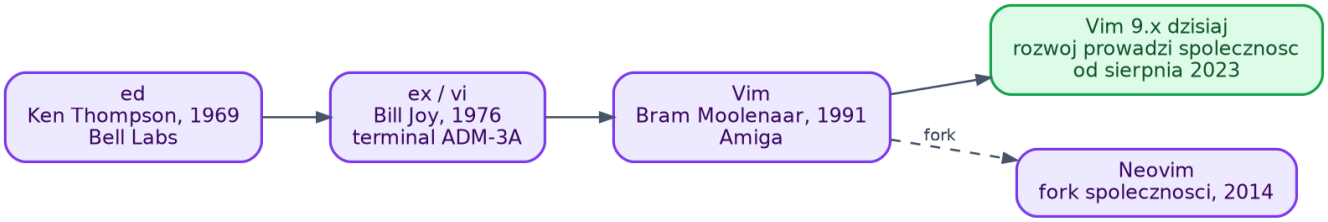
`vi` trafił do niemal każdego systemu uniksowego, ale każdy producent dopisywał własne rozszerzenia, więc wersje się rozjeżdżały. W 1988 roku Holender Bram Moolenaar zaczął poprawiać

`Stevie`, wcześniejszy klon `vi` na komputery Amiga, i w 1991 roku wydał pierwszą wersję własnego edytora pod nazwą Vim. Projekt rósł przez kolejne trzy dekady pod jego opieką, aż do sierpnia 2023 roku, gdy Bram Moolenaar zmarł. Rozwój Vima przejęła społeczność programistów z Christianem Brabandtem na czele, a wersja 9.1, wydana w styczniu 2024 roku, jest dedykowana jego pamięci. Dziś aktualna gałąź to Vim 9.x, rozwijana dalej w tym samym duchu: stabilność i zgodność wsteczna zamiast gwałtownych zmian.

Nie wszyscy zgodzili się z tym tempem. W 2014 roku Thiago de Arruda założył fork o nazwie Neovim, po tym jak jego propozycja przebudowy Vima od środka nie znalazła poparcia u ówczesnych opiekunów projektu. Neovim ma inny silnik wewnętrzny i wtyczki pisane w Lua zamiast Vimscriptu, ale klawiatura działa tak samo: kto zna Vima, w Neovimie czuje się od razu jak u siebie.

Ciekawostka na koniec: pytanie „jak wyjść z Vima" jest według danych samego serwisu Stack Overflow jednym z najczęściej odwiedzanych pytań w historii tego serwisu, otwieranym przez ponad milion osób. Nie ma w tym nic wstydliwego: to pytanie zadał sobie prawie każdy, kto kiedykolwiek trafił do tego edytora bez wcześniejszego przygotowania.

Jak to działa w jednym obrazku



Rodowód Vima: od ed przez ex i vi do Vima oraz jego forka Neovim (kliknięcie otwiera pełny rozmiar)

Dlaczego Zielona

> **Z ŻYCIA WZIĘTE**
>
> Drugi tydzień pracy Tomka. Dostaje zadanie: poprawić literówkę w `/etc/fstab` na `docker01`, żeby dysk z danymi montował się pod właściwą nazwą. Loguje się przez SSH, wpisuje `sudo vim /etc/fstab`, poprawia literówkę, a potem przez piętnaście minut próbuje wyjść z edytora, naciskając na przemian <kbd>Ctrl</kbd>+<kbd>C</kbd>, <kbd>Ctrl</kbd>+<kbd>Z</kbd> i samo <kbd>Esc</kbd>, aż w końcu zamyka całe okno

> terminala, zostawiając niezapisaną poprawkę.

Vim nie jest wyborem stylistycznym administratorów Zielonej, tylko koniecznością. Na każdym serwerze linuksowym w firmie, od pełnej instalacji `docker01` po najbardziej okrojony kontener, coś z rodziny `vi` jest już zainstalowane i gotowe do pracy bez dokładania czegokolwiek. Na serwerach Rocky Linux 9 w pełnej instalacji stoi pakiet `vim-enhanced` z pełnym Vimem, na instalacjach minimalnych zostaje tylko okrojony `vi`, ale klawiatura działa tak samo w obu przypadkach.

Ola, prowadząca zespół programistów, korzysta z Vima od lat i ma go rozbudowanego wtyczkami: podświetla składnię niestandardowych formatów, podpowiada nazwy zmiennych i skacze między plikami projektu jednym skrótem. Tomek na razie potrzebuje dużo mniej: otworzyć plik konfiguracyjny, poprawić jedną linię i wyjść bez katastrofy. Ta książka prowadzi dokładnie tą drogą, od pierwszego bezpiecznego wyjścia do wygody, z jakiej korzysta Ola.

Mapa tej książki

Strona	Na jakie pytanie odpowiada
Vim: edytor, z którego nie da się wyjść	Skąd wziął się Vim i po co Zielona go używa
Pierwsze kroki: otwarcie, zapis, wyjście	Jak otworzyć plik, zapisać go i bezpiecznie wyjść
Tryby: dlaczego klawisze robią różne rzeczy	Czym jest tryb i jak poznać, w którym akurat jesteśmy
Poruszanie się po pliku	Jak przesuwać kursor bez myszki, szybciej niż strzałkami
Edycja: kasowanie, kopiowanie, wklejanie, cofanie	Jak zmieniać tekst gramatyką operator plus ruch
Szukanie i zamiana	Jak znaleźć fragment tekstu i podmienić go w całym pliku
Wiele plików: bufory, okna, karty	Jak pracować na kilku plikach naraz w jednym oknie terminala
Makra, rejestry, bloki i inne sztuczki	Jak zautomatyzować powtarzalną edycję
Konfiguracja: plik .vimrc	Jak dostosować Vima raz na zawsze, zamiast ustawiać go co sesję
Ściągawka do wydruku	Gdzie znaleźć wszystkie skróty na jednej stronie

Powiązane: Edytory tekstu w terminalu: vim, nano, mcedit · Wyrażenia regularne: BRE, ERE, PCRE

Źródła: en.wikipedia.org/wiki/Vim_(text_editor) · en.wikipedia.org/wiki/Bram_Moolenaar · en.wikipedia.org/wiki/Ed_(text_editor) · en.wikipedia.org/wiki/Neovim · catonmat.net, „Here is why vim uses hjkl keys as arrow keys" · thenewstack.io, „Vim After Bram: A Core Maintainer on How They've Kept It Going" · stackoverflow.blog, „Stack Overflow: Helping One Million Developers Exit Vim"

Pierwsze kroki: otwarcie, zapis, wyjście

Poziom: podstawy **Dotyczy: Vim 9.x** **Czas: ~15 min**

To jest najważniejsza strona całej książki. Zanim nauczymy się czegokolwiek więcej, musimy umieć bezpiecznie otworzyć plik, zapisać zmiany i wyjść, bez zamykania całego okna terminala jak ostatniej deski ratunku.

Fiszka

TYLKO ODCZYT

Otwarcie pliku: vim

```
vim /etc/fstab
```

`vim SciezkaDoPliku` otwiera plik w trybie normalnym. W tym trybie klawisze nie wpisują tekstu, tylko sterują edytorem, więc samo otwarcie pliku niczego nie zmienia, nawet gdy naciśniemy kilka niewłaściwych klawiszy.

Przykład:

```
"/etc/fstab" 12L, 512B
```

vim = Vi IMproved: ulepszony edytor vi

TYLKO ODCZYT

Wejście w tryb wstawiania: i

```
i
```

`i` przełącza z trybu normalnego w tryb wstawiania dokładnie w miejscu kursora, dolny pasek pokazuje wtedy `-- INSERT --`. Warianty `a` (wstawiamy za kursorem), `o` (otwiera nową,

pustą linię poniżej) i `O` (otwiera linię powyżej) robią to samo, tylko zaczynają pisanie w innym miejscu.

□□ TYLKO ODCZYT

Wyjście z trybu wstawiania: Esc

```
Esc
```

<kbd>Esc</kbd> zawsze i bezwarunkowo wraca do trybu normalnego, niezależnie od tego, w jakim trybie akurat się znajdujemy. To jest pierwszy odruch w każdej sytuacji, w której nie wiadomo, co się dzieje na ekranie.

□□ TYLKO ODCZYT

Zapis bez wychodzenia: :w

```
:w
```

`:w` zapisuje bufor na dysk i zostawia nas w edytorze, gotowych do dalszej pracy. Wpisujemy to poleceniami z dwukropkiem z trybu normalnego, nigdy z trybu wstawiania.

□□ TYLKO ODCZYT

Zapis i wyjście: :wq, :x, ZZ

```
:wq
:x
ZZ
```

Wszystkie trzy zapisują plik i zamykają edytor, ale nie identycznie. `:wq` zapisuje zawsze, nawet gdy nic nie zmieniliśmy. `:x` i `ZZ` (dwa wielkie Z naciśnięte w trybie normalnym, bez dwukropka) zapisują tylko wtedy, gdy plik faktycznie się zmienił, co jest odrobinę szybsze i nie odświeża niepotrzebnie znacznika czasu pliku.

□□ TYLKO ODCZYT

Wyjście bez zapisywania: :q!

```
:q!
```

`:q!` wymusza wyjście z porzuceniem wszystkich niezapisanych zmian, bez żadnego pytania. To jest komenda, której szukamy, gdy coś w pliku poszło nie tak i wolimy zacząć od nowa niż ratować bieżący stan.

TYLKO ODCZYT

Wyjście ze wszystkich okien: :qa!

```
:qa!
```

`:qa!` różni się od `:q!` zakresem: `:q!` zamyka jedno, bieżące okno albo kartę, `:qa!` (quit all) zamyka naraz wszystkie otwarte okna i karty tej samej sesji Vima, porzucając zmiany we wszystkich z nich. Gdy pracujemy tylko na jednym pliku, oba polecenia dają ten sam efekt, różnica wychodzi na jaw dopiero po otwarciu kilku plików jednocześnie, co pokazuje strona Wiele plików: bufory, okna, karty.

ZMIENIA SYSTEM

Zapis pliku otwartego bez sudo: :w !sudo tee %

```
:w !sudo tee % > /dev/null
```

Ratunek, gdy otworzymy plik systemowy zwykłym `vim plik` bez `sudo`, wprowadzimy zmiany, i dopiero przy `:w` dostaniemy błąd braku uprawnień. `%` to symbol bieżącego pliku, cała komenda przepuszcza zawartość bufora przez `sudo tee`, zapisując ją z podwyższonymi uprawnieniami, bez zamykania edytora i bez utraty pracy.

Przykład:

```
E212: Can't open file for writing
```

TYLKO ODCZYT

Interaktywny samouczek: vimtutor

```
vimtutor
```

Osobny program instalowany razem z pełnym pakietem Vima, kopiuje plik ćwiczeniowy do katalogu tymczasowego i prowadzi przez piętnaście do dwudziestu minut praktycznych ćwiczeń z wejściem, zapisem i wyjściem opisanymi wyżej. Najszybszy sposób, żeby powyższe komendy faktycznie weszły w palce, zamiast zostać wiedzą teoretyczną.

vimtutor = samouczek Vima: interaktywny program instalowany razem z Vimem, uczy podstaw w piętnaście do dwudziestu minut

Z życia wzięte

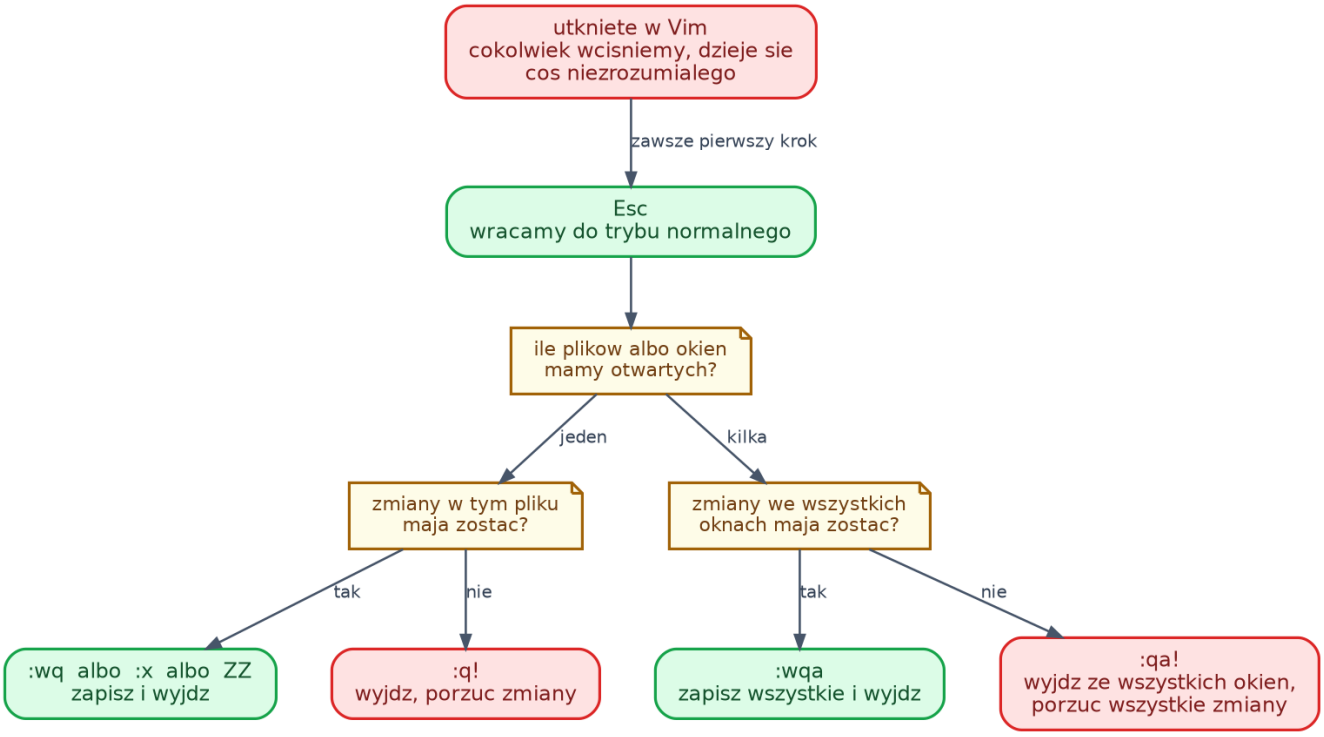
Z ŻYCIA WZIĘTE

Drugi tydzień pracy Tomka. Dostaje zadanie poprawienia wpisu w `/etc/fstab` na `docker01`, żeby dysk z danymi montował się pod właściwą nazwą. Wpisuje `sudo vim /etc/fstab`, poprawia literówkę klawiszem `i`, a potem przez piętnaście minut próbuje wyjść, naciskając na przemian <kbd>Ctrl</kbd>+<kbd>C</kbd> i samo <kbd>Esc</kbd>, aż w końcu zamyka całe okno terminala razem z niezapisaną poprawką.

Jak to działa

Pojęcie: tryb. Vim jest edytorem modalnym: te same klawisze robią różne rzeczy w zależności od tego, w jakim trybie akurat pracujemy. W trybie normalnym litera `i` przełącza tryb, w trybie wstawiania ta sama litera po prostu wpisuje znak „i" do tekstu. Pełne omówienie wszystkich trybów i tego, jak poznać, w którym jesteśmy, czeka na następnej stronie: Tryby: dlaczego klawisze robią różne rzeczy.

Gdy zupełnie nie wiadomo, co się dzieje, poniższy schemat prowadzi krok po kroku do bezpiecznego wyjścia z dowolnej sytuacji.



Jak wyjść z Vima: drzewo decyzyjne od utknięcia do zapisu albo porzucenia zmian (kliknięcie otwiera pełny rozmiar)

Częste błędy

Objaw	Przyczyna	Co robimy
Nie wiemy, w jakim jesteśmy trybie, klawisze robią coś dziwnego	zgubiliśmy się między trybem normalnym a wstawiania	<kbd>Esc</kbd>, sprawdzenie dolnego paska (`-- INSERT --` oznacza tryb wstawiania, jego brak, tryb normalny)
`:q` odmawia wyjścia komunikatem `E37: No write since last change`	są niezapisane zmiany, Vim celowo nie pozwala ich stracić przez przypadek	`:wq`, jeśli zmiany mają zostać, `:q!`, jeśli mają zostać porzucone
Wpisanie `qa!` w trybie normalnym nic nie robi albo zaczyna nagrywać makro	`qa!` bez dwukropka to nie jest polecenie wyjścia, w trybie normalnym `q` zaczyna nagrywanie makra do rejestru, a litera po nim wskazuje ten rejestr	poprawna komenda to `:q!`, z dwukropkiem na początku
`:w` kończy się `E212: Can't open file for writing`	plik systemowy otwarty bez `sudo`	`:w !sudo tee % > /dev/null`, bez zamykania edytora

□□ Ćwiczenia

1. Otwieramy dowolny plik tekstowy przez `vim`, dopisujemy jedną linię i chcemy ją zapisać, zostając w edytorze do dalszej pracy. Jakiej komendy używamy?

Odpowiedź

`:w`. Zapisuje bufor na dysk bez zamykania edytora, w przeciwieństwie do `:wq`, które od razu kończy sesję.

1. Wprowadziliśmy zmiany w pliku, uznaliśmy, że są błędne, i chcemy wrócić do wersji sprzed otwarcia edytora, bez zapisywania niczego. Jaka jest właściwa komenda i dlaczego stara notatka `qa!` nie zadziała?

Odpowiedź

Właściwa komenda to `:q!`. `qa!` bez dwukropka nie jest poleceniem trybu wiersza poleceń: w trybie normalnym Vim odczyta literę `q` jako początek nagrywania makra, `a` jako nazwę rejestru, do którego to makro ma trafić, a `!` jako zwykły znak do wpisania przy najbliższym wejściu w tryb wstawiania. Nic z tego nie zamyka edytora.

Powiązane: Tryby: dlaczego klawisze robią różne rzeczy · Wiele plików: bufory, okna, karty · Uprawnienia do plików i sudo · Konfiguracja: plik .vimrc · Ściągawka do wydruku

Źródła: `man 1 vim` · `:help quitting` w samym Vimie

Tryby: dlaczego klawisze robią różne rzeczy

Poziom: podstawy | Dotyczy: Vim 9.x | Czas: ~15 min

Cała trudność Vima dla początkującego sprowadza się do jednej rzeczy: nie wiadomo, w którym trybie akurat jest edytor. Ta strona nazywa to zjawisko i pokazuje, jak zawsze mieć nad nim kontrolę.

Fiszka

TYLKO ODCZYT

Tryb normalny: startowy tryb Vima

```
Esc
```

Tryb, w którym Vim startuje po otwarciu pliku i do którego zawsze wraca po naciśnięciu <kbd>Esc</kbd>. Klawisze są tu komendami: ruchem, kasowaniem, kopiowaniem, wejściem w inne tryby, nigdy zwykłym pisaniem tekstu.

TYLKO ODCZYT

Tryb wstawiania: pisanie tekstu

```
i    a    o    O
```

Wchodzimy przez `i` (przed kursorem), `a` (za kursorem), `o` (nowa linia poniżej) albo `O` (nowa linia powyżej). Tu klawiatura działa jak w zwykłym edytorze tekstu, każdy znak trafia do pliku.

TYLKO ODCZYT

Tryb wizualny: zaznaczanie przed działaniem

```
v    V    Ctrl+v
```

`v` zaznacza dowolny ciągły fragment tekstu, `V` zawsze całe linie od brzegu do brzegu, `Ctrl+v` prostokątny blok kolumn niezależny od długości poszczególnych linii. Po zaznaczeniu kolejna komenda (`d`, `y`, `>`) działa na całym zaznaczeniu naraz.

□□ TYLKO ODCZYT

Tryb wiersza poleceń: polecenia tekstowe

```
:
```

Wchodzimy przez dwukropek naciśnięty w trybie normalnym. Tu wpisujemy polecenia, których nie da się zmieścić w pojedynczym klawiszu: zapis, wyjście, podstawienia, ustawienia. Polecenie kończymy klawiszem Enter, a samo `:` bez niczego więcej opuszczamy klawiszem <kbd>Esc</kbd>.

□□ TYLKO ODCZYT

Tryb zastępowania: nadpisywanie znaków

```
R
```

`R` (wielkie r) działa jak tryb wstawiania, tylko każdy wpisywany znak nadpisuje ten, który stał pod kursorem, zamiast przesuwać resztę linii w prawo. Przydatne przy poprawianiu pojedynczych znaków w tabelach czy liczbach o stałej szerokości kolumn.

□□ ZMIENIA SYSTEM

Sprawdzenie bieżącego trybu: :set showmode

```
vim -c 'set showmode' /tmp/przyklad.txt
```

`:set showmode` (włączone domyślnie w większości dystrybucji) pokazuje nazwę aktywnego trybu w lewym dolnym rogu ekranu, na przykład `-- INSERT --` albo `-- VISUAL --`. Brak takiego napisu oznacza tryb normalny.

Przykład:

```
-- INSERT --
```

vim = Vi IMproved: ulepszony edytor vi

Z życia wzięte

Z ŻYCIA WZIĘTE

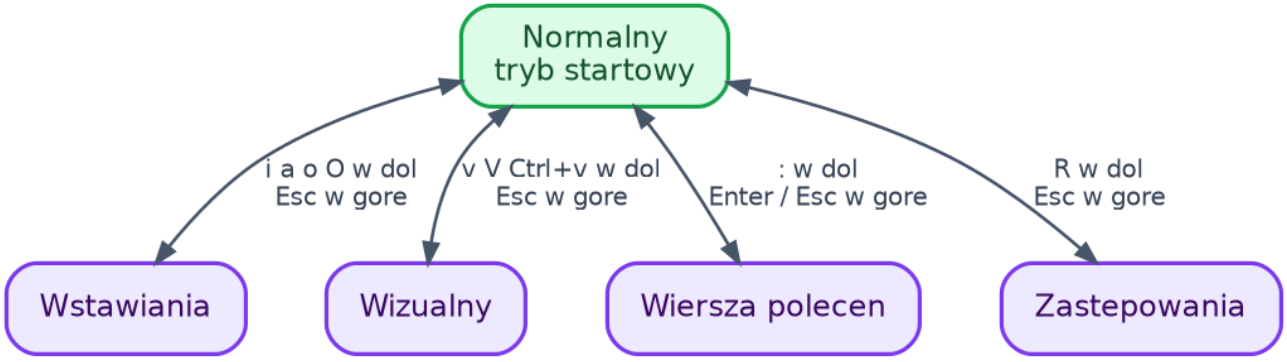
Ola tłumaczy Tomkowi różnicę między trybami na przykładzie z jego własnej wpadki sprzed tygodnia: gdy w `/etc/fstab` naciskał losowe litery myśląc, że wpisuje tekst, Vim był w trybie normalnym i czytał każdą literę jako komendę. Fragmenty linii znikały, bo `d` w tym trybie zaczyna kasowanie, a nie pisanie znaku „d".

Jak to działa

Pojęcie: edytor modalny. Edytor, w którym te same klawisze mają różne znaczenie zależnie od trybu pracy, w odróżnieniu od edytorów typu Notatnik czy `nano`, gdzie klawiatura zawsze robi to samo, czyli wpisuje znaki. Model modalny powstał w latach 70., na terminalach bez klawiszy funkcyjnych i strzałek, żeby zmieścić bogaty zestaw operacji edycyjnych na zwykłych literach alfabetu.

Analogia: samochód ze skrzynią manualną. Kierowca, który zmienia bieg, wciska sprzęgło i przesuwa dźwignię, a ten sam gest w innym momencie jazdy oznaczałby coś zupełnie innego. Vim działa podobnie: naciśnięcie <kbd>Esc</kbd> to zawsze powrót do neutralnego biegu, czyli trybu normalnego, z którego dopiero świadomie wybieramy kolejny tryb, tak jak kierowca świadomie wybiera bieg do jazdy do przodu albo do tyłu. Kłopoty zaczynają się, gdy zapomnimy, na jakim biegu jesteśmy, i spróbujemy ruszyć, myśląc, że mamy włączony inny.

Poniższy schemat pokazuje wszystkie pięć trybów i klawisze, którymi się między nimi przechodzi.



Przejścia między trybami Vima: normalny, wstawiania, wizualny, wiersza poleceń i zastępowania (kliknięcie otwiera pełny rozmiar)

Częste błędy

Objaw	Przyczyna	Co robimy
Litery znikają z tekstu zamiast się pojawiać	jesteśmy w trybie normalnym, myśląc że to tryb wstawiania	<kbd>Esc</kbd>, sprawdzenie dolnego paska, dopiero potem świadome `i`
Wpisany tekst nadpisuje istniejące znaki zamiast ich rozsuwać	tryb zastępowania (`R`) włączony przez pomyłkę zamiast trybu wstawiania	<kbd>Esc</kbd>, ponowne wejście przez `i`
Po `:` nic się nie dzieje, dopóki nie naciśniemy czegoś jeszcze	jesteśmy dokładnie w trybie wiersza poleceń, czeka on na treść polecenia i Enter	dokańczamy polecenie albo wychodzimy z niego przez <kbd>Esc</kbd>
Ruch kursora w trybie wizualnym rozciąga zaznaczenie zamiast po prostu przesuwać kursor	to zamierzone działanie trybu wizualnego, nie błąd	<kbd>Esc</kbd> kończy zaznaczanie bez wykonywania żadnej operacji na nim

Ćwiczenia

1. Otwieramy plik, jesteśmy w trybie wstawiania i widzimy na dole ekranu napis `-- INSERT --`. Chcemy wykonać komendę `dd`, żeby skasować bieżącą linię. Co musimy zrobić najpierw?

Odpowiedź

Nacisnąć <kbd>Esc</kbd>, żeby wrócić do trybu normalnego. `dd` wpisane w trybie wstawiania po prostu doda litery „d" i „d" do tekstu, zamiast skasować linię.

1. Jaka jest różnica między trybem wizualnym liniowym (`V`) a trybem wizualnym blokowym (`Ctrl+v`) przy zaznaczaniu kilku linii z rzędu?

Odpowiedź

`V` zawsze zaznacza całe linie, od lewego do prawego brzegu, niezależnie od pozycji kursora w poziomie. `Ctrl+v` zaznacza prostokątny blok kolumn o stałej szerokości i wysokości, niezależny od faktycznej długości poszczególnych linii, co wykorzystuje strona Makra, rejestry, bloki i inne sztuczki do komentowania wielu linii naraz.

Powiązane: Pierwsze kroki: otwarcie, zapis, wyjście · Makra, rejestry, bloki i inne sztuczki

Źródła: `man 1 vim` · `:help vim-modes` w samym Vimie