

Spis treści

1	Wstęp	2
1.1	Po co w ogóle pisać ten poradnik? (czyli podejście)	2
1.2	Jak korzystać z tutoriala? (czyli instrukcja obsługi)	3
1.3	Co mogę z tym zrobić? (czyli licencja)	3
2	Tutorial (czyli część zasadnicza)	3
2.1	Tryby pracy	4
2.2	Poznaj schemat komend VIM-a	5
2.3	WYJDŹ!	6
2.4	Mnemonika	6
2.5	Wywołanie	7
2.6	Kontekst jest ważniejszy niż pozycja	8
2.7	Cytowanie metaznaków w wyrażeniach regularnych	10
2.8	Nie panikuj, masz historię poleceń	10
2.9	Wielkie litery i ŚMIERĆ PRZEZ CAPSLOCK	11
2.10	Wstaw, nadpisz, zmień	11
2.11	NIE POZOSTAWAJ W TRYBIE WSTAWIANIA	13

1 Wstęp

Jest wiele edytorów tekstu. Niektóre z nich są bez wątpienia świetne i nic nie stoi na przeszkodzie, żebyś ich używał. Jednakże są pewne powody, dla których warto używać właśnie VIM-a. Nawet pomimo tego, że powody te nie są unikalnymi cechami tegoż edytora.

- Dzięki wzrostowi zainteresowania platformami Uniksowymi (głównie Linux i Mac OS X), VIM jest powszechnie używanym edytorem
- VIM potrzebuje niewielkich ilości pamięci RAM i mocy procesora. Nawet średniej klasy komputer jest w stanie udźwignąć wiele instancji edytora jednocześnie
- VIM ma wiele “supermocy”, które sprawiają, że edycja plików przebiega dużo sprawniej
- VIM wytwarza wokół siebie aurę “geekowości”
- VIM ma bardzo aktywną społeczność użytkowników i deweloperów. Zawsze ją miał.

1.1 Po co w ogóle pisać ten poradnik? (czyli podejście)

Są inne bardzo dobre tutoriale i Google na pewno pomoże Ci w znalezieniu ich. Zapewne kolejny tutorial do VIM-a nie jest artykułem pierwszej potrzeby, ale ten jest wyjątkowy – bo mój.

Przyjąłem nieco odmienne podejście niż inni autorzy. Uważam, że istnieje pewien sposób myślenia, który sprawia, że nauka VIM-a staje się dużo prostsza. Omawiam również nawyki, dzięki którym zaprzyjaźnisz się z tym edytorem. Szczerze mówiąc nie wiem, czy ktoś przedstawił to w ten sposób.

Długo układałem (i układałem i układałem. . .) ten tutorial tak, żeby materiał był w odpowiedniej kolejności. Cały czas stawiając sobie za cel napisanie tego poradnika tak, żeby czytelnik, po dobrnięciu do końca, umiał korzystać z VIM-a w jak najbardziej profesjonalny sposób. Prawdopodobnie lepiej niż niejeden ich bardziej doświadczony kolega.

Doradzam cierpliwość i poleganie na pamięci długoterminowej, ale jednocześnie uważam, że ten poradnik to jeden z najszybszych sposobów na usprawnienie Twojej pracy z VIM-em, jak również całkiem niezły na naukę od zera. Pisałem ten poradnik dla raczej niecierpliwego programisty.

Zdecydowałem, że nie chcę pisać i sprzedawać książki, ponieważ każdy powinien mieć możliwość używania VIM-a jak zawodowiec, w każdym miejscu i czasie, bez konieczności wnoszenia dodatkowych opłat.

1.2 Jak korzystać z tutoriala? (czyli instrukcja obsługi)

Traktuj każdy punkt poradnika jako osobną lekcję i poświęć chwilę czasu na wypróbowanie danych komend, zanim przejdziesz do następnej. Możesz poświęcić cały dzień na każdą lekcję, a może nawet kilka, jeżeli dany materiał okaże się wyjątkowo trudny do przyswojenia.

Postaraj się przerabiać po kilka lekcji tygodniowo. Nie spiesz się. Nie przeciążaj swojego mózgu ucząc się ciągle nowych rzeczy, bo zapomnisz o tym, czego nauczyłeś się prędej. Tutorial ten będzie istniał tak długo, jak będziesz tego potrzebował. Możesz złapać oddech.

Nie nauczysz się VIM-a nie używając go, więc powinieneś przygotować sobie jakieś pliki (najlepiej kod jakiegoś wolnodostępnego programu), na których będziesz pracował. Jeszcze lepiej, jeżeli używasz VIM-a w pracy. Pomocne może być również przerabianie tego “kursu” we dwie osoby, żebyście z partnerem mogli się nawzajem wspierać.

1.3 Co mogę z tym zrobić? (czyli licencja)

Praca ta jest udostępniona na zasadach [licencji Creative Commons Attribution 3.0](#). Kopiuj, dziel się, udostępnij na swojej stronie. Nie udawaj, że to Twój tekst i dodaj informację o autorze. W ramach uprzejmości – jeśli uznasz, że jest to warte zachodu – nie będę miał nic przeciwko poinformowaniu mnie o tym.

2 Tutorial (czyli część zasadnicza)

Nikt nie zna całego VIM-a. Jest tyle różnych poleceń, że wystarczy dla tysięcy istnień ludzkich, z których każde może używać VIM-a na swój własny sposób. Na szczęście nie musisz znać ich wszystkich. Wystarczy Ci tyle, ile potrzebujesz, żeby wykonać swoje zadanie.

Z tego również powodu, przez cały czas używania VIM-a będziesz poznawał nowe sztuczki. Sekret polega na tym, żeby nie poprzestawać na jednym konkretnym (kiepskim) sposobie wykonania danej czynności.

VIM ma uzupełnianie składni, historię operacji, skróty klawiszowe, personalizację ustawień klawiatury, makra i skrypty. Możesz go przemienić w **swój** edytor, przystosowany do **twojego** środowiska. Brzmi świetnie, ale pewnie ucieszysz się z faktu, że możesz być dużo wydajniejszy nawet bez dotykania tych zaawansowanych funkcji.

Jak mówi *Bram Moolenaar* (główny autor VIM-a), najlepszym sposobem na naukę VIM-a jest używanie go i zadawanie pytań. Ten poradnik pełen jest pytań, o których mogłeś nawet nie pomyśleć, żeby je zadać. To najbardziej wartościowa

rzecz, którą mogę Ci przekazać.

VIM ma wbudowany tutorial. Możesz go wypróbować, zwłaszcza jeśli nie spodoba Ci się mój. Żeby z niego skorzystać, wystarczy wywołać polecenie vimtutor z poziomu wiersza poleceń Twojego systemu. Jest on bardzo przystępny i raczej kompletny (w przeciwieństwie do mojego, który jest raczej przyjemny, ale na pewno nie kompletny).

Proponuję również rozważyć GVIM-a. Sprawi on, że będzie Ci się pracować dużo przyjemniej. Jeżeli masz pod ręką tylko VIM-a, to oczywiście nadal możesz korzystać z tego poradnika, ale GVIM wygląda dużo lepiej, pozwala na używanie myszy i ma menu oraz ikony dla tych z Was, którzy przywykli do takich rzeczy.

2.1 Tryby pracy

Oryginalny VI powstał w czasach, gdy czarno-zielone ekrany były szczytem osiągnięć techniki (zapytaj dziadka o terminale ASCII). Nie było wtedy tak wielu klawiszy modyfikujących (shift, alt, ctrl, super, fn), jak również urządzeń wskazujących. Przyjmijmy, że mieliśmy do dyspozycji tylko klawisze shift i ctrl (nawet jeśli to nieprawda).

Programując (tak jak i robiąc cokolwiek innego na komputerze) mieliśmy oczy cały czas skierowane na ekran, a dłonie położone na klawiaturze. VI sprawił, że pisanie programów odbywało się **szybko**, ponieważ VI jest trochę jak gra komputerowa, w której każde naciśnięcie klawisza sprawia, że coś się dzieje.

Jeżeli używasz vima i wciskanie klawiszy sprawia, że dzieją się albo rzeczy cudowne, albo straszne, to wiesz, że jesteś w trybie komend, który jest domyślnym trybem edytora. Komendy są przypisane do zwykłych klawiszy takich jak o, y czy g, zamiast kombinacji w stylu Ctrl+Alt+Shift+Esc.

VIM jest wyposażony w kombinacje klawiszy pozwalające wykorzystać specjalne moce, takie jak nawigacja między funkcjami w oddzielnych plikach, poprawianie formatowania całych list w samym środku dokumentu, uzupełnianie składni, skróty, szablony i inne takie, ale o tym później.

Musi istnieć również sposób na wprowadzanie tekstu, ale większość klawiszy ma już przypisane magiczne funkcje! Jedynym sensownym rozwiązaniem było stworzenie “trybu wprowadzania”, dzięki któremu wciśnięcie klawisza a sprawi, że w tekście pojawi się litera a. Zupełnie jak na maszynie do pisania (spytaj ojca co to). Nazywamy to “trybem wprowadzania”. Niewiele się w tym trybie dzieje, poza zwykły, starym, nudnym pisaniem. Powinieneś używać tego trybu tylko jeżeli chcesz coś napisać, bo fajne rzeczy dzieją się tylko w trybie zwykłym (komend).

Poznasz wiele wygodnych sposobów na przejście w tryb wprowadzania, ale na chwilę obecną powinieneś wiedzieć, że aby ten tryb opuścić i powrócić do trybu gry zręcznościowej, należy wcisnąć klawisz Escape. Gdy zrozumiesz, że w

VIM-ie istnieją dwa zasadnicze tryby pracy, twoja praca z nim przestanie wprawiać Cię w zakłopotanie i otworzy się przed Tobą droga do zostania guru tego edytora.

2.2 Poznaj schemat komend VIM-a

Przez większość czasu albo będziesz uzyskiwał rezultaty po pojedynczym wciśnięciu klawisza, albo będziesz wprowadzał komendy kończąc je komendą ruchu (często również powtarzając te same naciśnięcia: tzw. “podwójny skok”). Kiedy rozpoczniesz naukę innych rzeczy (rejstry, powtórzenia itd.) może Ci się wydać, że VIM nie jest spójny, ale tak nie jest. Schemat komend jest raczej spójny. Po prostu niektóre jego części są opcjonalne.

Rejestr (opcjonalnie, domyślnie rejestr “`”`)

Powtórzenia (opcjonalnie): np. 13

Operacja: `y` (Od angielskiego *yank* – *szarpać*. Możemy to rozumieć jako *wyszarpnięcie* tekstu)

Przesunięcie (w zależności od operacji): `yy` (*podwójny skok*, żeby wykonać operację na aktualnym wierszu, konwencja używana w VI)

Komendy VIM-a działają zgodnie z powyższym schematem. Są komendy, które nie używają rejestrów i takie, które nie przyjmują polecenia przesunięcia, ale w większości przypadków tak to właśnie wygląda.

Rejestr to w zasadzie po prostu schowek (dla operacji typu kopiuj/wklej). Większość edytorów udostępnia tylko jeden taki schowek. W VIM-ie masz ich aż nadto, ale nie musisz ich wszystkich używać, więc się nimi nie przejmuj aż nie dotrzesz do lekcji o rejestrach.

Powtórzenia to liczba oznaczająca ile razy chcesz daną operację wykonać. Jeżeli nie podasz żadnej liczby, domyślną wartością jest 1.

Operacja to klawisz, który rozkazuje VIM-owi coś zrobić. Najczęściej są to zwykłe naciśnięcia klawiszy i większość z nim nie wymaga klawiszy modyfikujących jak `Shift`, `Ctrl` czy `Alt`.

Przesunięcie to polecenie oznaczające ruch kursora. Jest ich całkiem sporo, ponieważ jest wiele sposobów na poruszanie się. Nie panikuj jednak – na początku możesz używać klawiszy strzałek, jeżeli koniecznie musisz. Poświęciłem w tym poradniku całą sekcję na polecenia przesunięcia.

Spróbujmy wyjaśnić jak taki schemat działa. Jeżeli chcę skopiować 13 wierszy do mojego domyślnego rejestru, mogę pominąć rejestr, wpisać 13 jako liczbę powtórzeń, wcisnąć `y` jako komendę kopiowania i potem jeszcze raz `y` jako komendę ruchu (co oznacza aktualny wiersz). Takie polecenie (`13yy`) skopiuje 13 wierszy do domyślnego rejestru, poczynając od aktualnego. Jeżeli naciśnę teraz `p` (nie

precyzując rejestr, pomijając liczbę powtórzeń i wiedząc, że komenda `put` nie przyjmuje polecenia ruchu), to te wiersze zostaną wklejone zaraz za bieżącym. Znając ten schemat będziesz mógł wpływać na wszystko, czego się nauczysz o VIM-ie. Naucz się wszystkich sposobów poruszania się, a będziesz wiedział jak wycinać, wklejać, poprawiać formatowanie, zmieniać poziom wcięcia i wiele innych rzeczy, które chciałbyś zrobić.

2.3 WYJDŹ!

Po tym jak uruchomisz sesję VIM, pewnie chciałbyś z niej kiedyś wyjść. Jest na to kilka sposobów. Spróbuj tych:

Polecenie	Rezultat
<code>:q</code>	Zamyka aktualne okno (lub edytor, jeżeli to jest jedyne), jeżeli nie ma w nim niezapisanych zmian
<code>:q!</code>	Zamyka aktualne okno, nawet jeżeli były w nim niezapisane zmiany
<code>:qa</code>	Zamyka wszystkie okna, jeżeli nie ma w nich niezapisanych zmian
<code>:qa!</code>	Zamyka wszystkie okna, nawet jeżeli były w nich niezapisane zmiany
<code>:wq</code> lub <code>:x</code> lub <code>ZZ</code>	Zapisuje zmiany i zamyka aktualne okno

Kiedy wciśniesz dwukropek, kursor przeniesie się do lewego dolnego rogu ekranu. Później dowiesz się dlaczego. Na razie wystarczy, że będziesz wiedział, że tak powinno się dziać, a te polecenia działają. Zauważ, że przed `ZZ` nie ma dwukropka. *Od tłumacza: Po poleceniach rozpoczynających się dwukropkiem należy wcisnąć klawisz `Enter`, inaczej nie odniosą one skutku.*

Jeżeli nie możesz sobie poradzić z wyjściem z VIM-a, upewnij się, że Caps-Lock jest wyłączony i wciśnij Escape. Jeśli sprawi Ci to radość, możesz go nacisnąć kilkakrotnie. Jeżeli usłyszysz brzęczyk, będziesz wiedział, że wcisnąłeś go wystarczającą liczbę razy. Wtedy podane wyżej polecenia na pewno zadziałają.

2.4 Mnemonika

Nie wszystkie komendy są łatwe do zapamiętania. Twórcy VIM-a bardzo się starali, ale łatwo jest znaleźć więcej niż 26 czynności, które chciałbyś wykonywać korzystając z edytora, a poza tym tak się dziwnie składa, że niewiele jest słów zaczynających się na literę `q` i mających jakiś sens w programie do edycji tekstu. Jednakże wiele komend da się łatwo zapamiętać. Są to komendy do odnajdywania najbliższego znaku (ang. *Find* – `f`), przesunięcia o jedno słowo do tyłu (*Back one word* – `b`) lub do przodu (*Word* – `w`) itd.

Wiele poleceń jest mnemonicznych, pod warunkiem, że znasz żargon. Przykładowo: ponieważ zarówno *copy* (kopiuj) jak i *cut* (wytnij) zaczynają się na *c*, używamy slangowych *yank* (wyszarpnij – jako kopiuj), *delete* (usuń – jako wytnij) i *put* (połóż – jako wklej). Y, D, P. Może się to wydawać nieco dziwne, ale da się to zapamiętać. Pamiętaj, że z czasem przechodzi to do pamięci mięśniowej, ale autorzy VI i VIM-a starali się nie być całkowicie abstrakcyjni, gdy wszystko leżało w ich gestii. Czasem nie mieli jednak wielkiego wyboru.

2.5 Wywołanie

Gdy już wiesz jak wyjść z VIM-a, pora się nauczyć jak się do niego wejść. Zazwyczaj uruchamiamy VIM-a z wiersza poleceń, aczkolwiek całkiem możliwe, że masz do dyspozycji jakieś menu bądź inne sposoby na uruchamianie programów. Jest kilka sposobów na uruchomienie VIM-a:

Polecenie	Rezultat
<code>vim</code>	Uruchomienie z pustym oknem
<code>vim plik.txt</code>	Uruchomienie z załadowanym i gotowym do edycji plikiem <code>plik.txt</code>
<code>vim +N plik.txt</code>	Uruchomienie z załadowanym i gotowym do edycji plikiem <code>plik.txt</code> ; rozpoczyna edycję w linii <code>N</code>
<code>vimtutor</code>	Uruchomienie w trybie samouczka – całkiem niezły pomysł
<code>vimdiff plik1.txt plik2.txt</code>	Uruchomienie w trybie porównywania i łączenia dwóch plików
<code>vim .</code>	Uruchomienie w trybie przeglądarki plików

Oprócz tych wymienionych powyżej, jest jeszcze więcej. Na początek te powinny jednak wystarczyć. Wypróbuj `vimdiff` i `vimtutor`. Niektóre ze sposobów uruchamiania nie zadziałają dopóki nie skonfigurujesz swojego `.vimrc`, ale o tym później.

Jeżeli zamiast `vim` napiszesz `gvim`, to uruchomi się wybajerzona wersja graficzna VIM-a (oczywiście zakładając, że jest zainstalowana). Ma ona kilka dodatkowych możliwości. Zapewne spodoba Ci się bardziej niż czysty VIM, bo jest ona jak VIM z polewą czekoladową. Wszystko, co mówimy tutaj o VIM-ie działa również w `gVIM`-ie, więc możesz bez obaw używać tego tutoriala.

Nie jesteś skazany na edycję tylko jednego pliku jednocześnie. Możesz urucho-

nić (g)VIM-a z wieloma nazwami plików jako argumentami. Jeżeli tak zrobisz, to jest kilka parametrów, dzięki którym możesz uzyskać ciekawe efekty. Oczywiście ciekawsze będą jak nauczysz się obsługiwać podzielone okna, więc pewnie będziesz chciał zajrzeć do nich później.

Parametr	Rezultat
-o (mała litera "o")	Otwiera wiele plików w oknach ułożonych w poziomie
-O (wielka litera "O")	Otwiera wiele plików w oknach ułożonych w pionie
-p	Otwiera wiele plików w osobnych kartach. (Osobiście nie cierpię tego)

2.6 Kontekst jest ważniejszy niż pozycja

Biedna duszyczka, używająca VIM-a po raz pierwszy będzie wciskać klawisze strzałek, poruszając się w kodzie strasznie nieefektywnie. Będzie przewijać ekran o całą stronę w górę i w dół (przy okazji: to odpowiednio \hat{f} i \hat{b}) i przeczesywać kod swoimi zmęczonymi oczami. Ta biedna osoba jest powolna, jeszcze nie uświadomiona i zapewne bierze VIM-a za kiepską wersję windowsowego notatnika, zamiast za potężne narzędzie jakim jest.

Przy okazji – klawisze strzałek nie zawsze działają w VIM-ie. Nie obarczaj jednak winą za to VIM-a! Tak naprawdę jest to sposób w jaki Twój terminal obsługuje te klawisze. Przez to VIM nie potrafi rozpoznać, że Twoje strzałki to strzałki. Jeżeli masz ten problem, będziesz musiał poszukać rozwiązania na własną rękę.

Żeby używać VIM-a porządnie, musisz się najpierw nauczyć porządnie poruszać. Nie przewijaj w poszukiwaniu tekstu. Nie używaj do tego swoich oczu. Nie wiem czy zauważyłeś, ale od jakiegoś czasu komputery robią to za nas. Masz tutaj garść najważniejszych poleceń ruchu. Najłatwiej jest się poruszać szukając:

Polecenie	Rezultat
/	Wyszukaj do przodu; zostaniesz zapytany o wzór
?	Wyszukaj do tyłu; zostaniesz zapytany o wzór
n	Powtórz ostatnie wyszukiwanie (tak jak kropka powtarza komendy)
N	Powtórz ostatnie wyszukiwanie, w przeciwnym kierunku
t x	Przesuń do (ang. <i>to</i>) litery x , zatrzymując się tuż przed nią – przydatne przy komendach usunięcia i zmiany
f x	Znajdź (ang. <i>find</i>) literę x , zatrzymując się na niej – również przydatne przy komendach usunięcia i zmiany

Jeżeli szukanie zbytnio Ci nie pomoże, rozważ skoki:

Polecenie	Rezultat
gg	Początek pliku
G	Koniec pliku
0 (cyfra zero)	Początek wiersza
w	Jedno słowo do przodu
W	Następne słowo rozdzielone spacjami (pomija znaki specjalne)
b	Jedno słowo do tyłu
B	Poprzednie słowo rozdzielone spacjami (pomija znaki specjalne)
e	Koniec aktualnego słowa, lub następnego, jeżeli już jesteśmy na końcu
E	Koniec aktualnego słowa rozdzielonego spacjami (pomija znaki specjalne)

Poniższe komendy są bardzo przydatne, a przy okazji łatwe do zapamiętania, jeżeli znasz wyrażenia regularne:

Polecenie	Rezultat
^	Początek tekstu w danej linii - zdecydowanie lepsze niż korzystanie z klawiszy strzałek bądź komendy h
\$	Koniec tekstu w danej linii. Zdecydowanie lepsze niż korzystanie z klawiszy strzałek bądź komendy l

Kilka ciekawszych komend:

Polecenie	Rezultat
%	Odpowiadający nawias, klamra itp.
{	Początek akapitu
}	Koniec akapitu (najbliższa pusta linia)
(Początek zdania (zdania są oddzielone kropką i spacją)
)	Koniec zdania
..	Miejsce ostatniej zmiany w aktualnym wierszu
]]	Następna funkcja/klasa (C, Java, C++, Python)
[[Poprzednia funkcja/klasa (C, Java, C++, Python)

Ostatecznie, jeżeli nie jesteś w stanie wykorzystać wyszukiwania, skoków itd., nadal możesz korzystać z klawiatury, więc odłóż mysz na bok:

Polecenie	Rezultat
h	W lewo o jeden znak
l	W prawo o jeden znak
k	W górę o jeden wiersz
j	W dół o jeden wiersz
^f	W dół o jeden ekran
^b	W górę o jeden ekran

Dobrym pomysłem jest ustawienie opcji `hls` (*highlight search* – podświetlaj wyszukiwane) w pliku `.vimrc`. Niedługo się dowiesz jak to zrobić. Tymczasowo możesz wpisać `:set hls` i wcisnąć `Enter`.

2.7 Cytowanie metaznaków w wyrażeniach regularnych

Jeżeli nie wiesz czym są wyrażenia regularne, pominię tę sekcję. Dla tych, którzy wiedzą i zdają sobie sprawę z tego, że komenda `/` przyjmuje wyrażenie regularne, a nie tylko zwykły tekst, będzie ona ważna. W pozostałych przypadkach wyda się pewnie kompletnie nie na miejscu i powinna zostać pominięta.

Powinieneś wiedzieć jak używać regeksów, bo kilka sztuczek z ich wykorzystaniem może sprawić, że będzie Ci się lepiej korzystało z Uniksa/Linuksa/Maca. Jest to zbyt rozległy temat, żeby go tutaj przedstawiać, ale możesz zerknąć na któryś ze świetnych tutoriali czy referencji dostępnych w sieci.

Najważniejszą sprawą do zapamiętania jest to, że VIM stawia na wygodę jeżeli chodzi o wyrażenia regularne. Ponieważ często szukasz, VIM zakłada, że wpisując `/+` chcesz znaleźć najbliższy znak `+`. W wyniku tego założenia wszystkie metaznaki muszą być poprzedzone wstecznym ukośnikiem (`\`). Czasem jest to problematyczne, ale jeżeli naprawdę chcesz znaleźć nawias poprzedzony plusem, jest to znaczne ułatwienie.

2.8 Nie panikuj, masz historię poleceń

Komendą cofającą zmiany jest `u` (ang. *undo*). Niezbyt trudne do zapamiętania, prawda? Jak już prędeż wspomniałem, wiele poleceń VIM-a jest mnemonicznych.

Komendą powtórzenie cofniętych zmian powinna być litera `r` (ang. *repeat*), ale jest już zajęta przez polecenie zamiany (*replace*); powiemy o tym nieco później. Pozostaje nam `Ctrl-R` (`^r`). Cóż... nie można mieć wszystkiego.

Jest jeszcze trochę odnośnie cofania i powtarzania, ale na razie tyle powinno wystarczyć. Ciesz się, że już możesz cofać i potwarzać zmiany. VIM nie jest już taki

nieudolny i niewybaczający, jak mogło Ci się wydawać, chociaż wciąż nie musisz go lubić. Poczekaj tylko, aż do gry wejdzie pamięć mięśniowa.

Jeżeli narobisz naprawdę niezłego bałaganu, to po prostu wyłącz edytor bez zapisywania (: qa!).

Jeżeli bardzo się obawiasz o swoje pliki, albo jesteś po prostu bardzo ostrożny, powinieneś używać jakiegoś systemu kontroli wersji. Zalecam, żebyś tak czy siak zaczynał naukę na bezwartościowych plikach, w jakimś tymczasowym katalogu, ale gdy przejdziesz do prawdziwej pracy, nie powinieneś się obawiać wprowadzania zmian. Kontrola wersji jest dobrym zabezpieczeniem i może być z powodzeniem stosowana jako sposób na kopie bezpieczeństwa. Proponuję rozważyć [Gita](#) i [Mercuriala](#) – oba są jednocześnie proste i potężne.

2.9 Wielkie litery i ŚMIERĆ PRZEZ CAPSLOCK

W przypadku wielu poleceń `Shift` albo odwraca kierunek (`N` jest przeciwieństwem `n`; zobacz: następny punkt), albo całkowicie zmienia sposób działania. Poruszając się do przodu o jedno słowo (`w`) można wcisnąć `W`, żeby nadal przesuwac się o jedno słowo, ale traktując znaki specjalne jako jego część. To samo tyczy się przeciwnego kierunku – `b` i `B`.

Ponieważ jednak wielkie litery mogą mieć zupełnie inne znaczenie niż ich małe odpowiedniki, powinieneś uważać, żeby nie wcisnąć klawisza `CapsLock`! Czasem zdarzy się, że przez przypadek wcisniesz `CapsLock` i gdy będziesz chciał przejść do kolejnego wiersza (`j`), połączysz go z obecnym. Wiele innych niepożądanych zmian może się zdarzyć w takiej sytuacji, jeżeli Twoje palce będą szybko przemierzać klawiaturę w celu wykonania jakiegoś skomplikowanego ciągu komend. To okropne.

Kiedy spotka Cię ŚMIERĆ PRZEZ CAPSLOCK, powinieneś od razu go wyłączyć i wciskać `u` tak długo, aż cofniesz niechciane zmiany. Jeżeli uważasz jednak, że to gra nie warta świeczki, wpisz po prostu `:e!` i zatwierdź enterem, a VIM wczyta plik ponownie z dysku. Trochę to kłopotliwe, ale na pewno kiedyś Ci się to zdarzy, więc warto, żebyś o tym wiedział. Niektórzy z tego powodu wyłączają `CapsLock` na stałe.

2.10 Wstaw, nadpisz, zmień

W VIM-ie jest kilka różnych sposobów na rozpoczęcie wprowadzania tekstu, jak już zostało wspomniane w sekcji *Tryby pracy*.

Twoim standardowym trybem pracy powinien być tryb komend. Po wciśnięciu niektórych klawiszy przechodzisz w tryb wstawiania lub nadpisywania. W trybie wstawiania, wszystko co piszesz zostaje wstawione w miejsce kursora, a to co było za nim jest spychane do prawej bądź do kolejnego wiersza.

Z kolei w trybie nadpisywania, wszystkie naciśnięcia klawiszy są wpisywane podobnie jak w trybie wstawiania, z tym wyjątkiem, że tekst za kursorem zamiast zostać odepchniętym, jest zastępowany przez kolejne wpisywane znaki. Aby przejść do tego trybu, musisz wydać odpowiednie polecenie w trybie komend.

W trybie wykonywania (ang. *ex mode*) wpisujesz łańcuch znaków w pole u dołu ekranu. Przejdziemy do tego później, bo jest to potężna rzecz. Jest też nieco za-
wiła, więc poczekamy z nią trochę. Do tego trybu przechodzisz wciskając : w trybie komend.

Z każdego tych trybów możesz powrócić do trybu normalnego (komend) wciskając Escape. Bardzo przydatny klawisz.

Polecenie	Rezultat
i	Rozpocznij pisanie w miejscu kursora
I	Rozpocznij pisanie na początku wiersza. Zdecydowanie lepsze niż wciskanie ^, a następnie i
a	Rozpocznij pisanie za kursorem
A	Rozpocznij pisanie na końcu wiersza. Zdecydowanie lepsze niż wciskanie \\$, a następnie a
r	Zamień pojedynczy znak w miejscu kursora
R	Przejdź do trybu nadpisywania, w którym niszczyliście nadpisujecie wszystko, aż do naciśnięcia Escape
s	Zamień pojedynczy znak w miejscu kursora i przejdź do trybu wstawiania
c	Komenda zastąpienia. Wymaga komendy przesunięcia następującej po niej. Najczęściej używam cw i cc
C	Jak c, ale stosowane do całej linii
o	Wstawia pustą linię pod bieżącą
O	Wstawia pustą linię nad bieżącą
:	Przejdź w tryb wykonywania poleceń (dla zaawansowanych uczniów)
!	Przejdź w tryb filtrowania przy pomocy powłoki systemowej (dla bardzo zaawansowanych uczniów)

Zastanów się przez chwilę nad wagą komendy c. Jeżeli użyjesz jej w połączeniu z t lub f, staje się ona naprawdę potężnym narzędziem. Gdybyś miał kursor na literze Z na początku tego akapitu, mógłbyś wykonać komendę ct., żeby przepisać całe zdanie, zachowując kropkę. To samo tyczy się innych, jak na przykład d do usuwania. Polecenia ruchu znacząco zwiększają możliwości komendy c i dlatego tak ważne jest, abyś je dobrze znał.

2.11 NIE POZOSTAWAJ W TRYBIE WSTAWIANIA

VIM jest stworzony do tego, żeby więcej się przemieszczać po tekście, niż edytować. Nagradza Cię za dostosowanie się do takiego trybu pracy – głównie tryb komend, z krótkimi przerwami na wprowadzanie.

Jeżeli będziesz próbował używać VIM-a jako kiepskiego zamiennika notatnika, tryby i nawigacja sprawią, że nigdy nie będziesz wydajny. Jak chcesz żeglować, musisz wsiąść do łódki. Jak chcesz efektywnie korzystać z VIM-a, musisz zaprzyjaźnić się z trybem komend.

Jeśli więc zatrzymujesz się, żeby nad czymś pomyśleć, wciśnij `Escape`. Jeżeli nie jesteś w trakcie pisania, powinieneś być w trybie komend. Jak chcesz przejść do następnej linijki czy w jakiegokolwiek inne miejsce w pliku, wciśnij `Escape`. Jak chcesz odejść od klawiatury, wciśnij `Escape`. W przeciwnym wypadku, zaczniesz wprowadzać komendy i zanim zorientujesz się, że nie jesteś w trybie komend, stracisz możliwość sensownego używania `.` czy cofania. Natomiast kiedy chcesz w trakcie pisania wydać jedno polecenie możesz zrobić to przez wciśnięcie kombinacji `Ctrl+o`. Vim po wykonaniu go automatycznie powróci do trybu wstawiania.