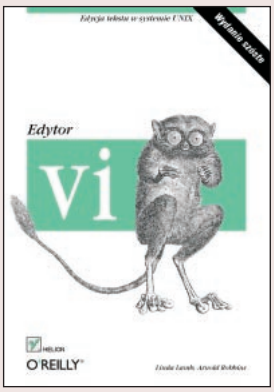
Edytor vi.

Autorzy: Linda Lamb, Arnold Robbins
Tłumaczenie: Tomasz Żmijewski
ISBN: 83-7197-539-2
Tytuł oryginału: Learning the vi Editor
Format: B5, stron: 304



Edytowanie tekstu to jedno z najbardziej typowych zadań realizowanych na komputerze, zaś vi jest jednym z najbardziej użytecznych, standardowych edytorów. Za jego pomocą można tworzyć w systemie UNIX nowe pliki tekstowe lub edytować istniejące.
Książka ta składa się z dwunastu rozdziałów oraz pięciu dodatków. Wszystkie je pogrupowano w trzy części.
W dwóch pierwszych rozdziałach, vi – edytor tekstu oraz Typowe zadania edycyjne, omówiono podstawowe polecenia edytora. Materiał ten należy ćwiczyć tak długo, aż przedstawione i opisane polecenia będzie można wykonywać niemal automatycznie („same będą wchodzić na klawiatu-rę").
W rozdziałach 3. i 4. (Biegiem po dokumencie i Dla bardziej zaawansowanych) skoncentrowano się na upraszczaniu własnej pracy.
W rozdziałach 5., 6. i 7. – kolejno Wprowadzenie do edytora ex, Podstawienia globalne i Zaawansowane techniki edycji – przedstawiono narzędzia umożliwiające w większym stopniu obciążenie komputera edycją tekstu, a nie użytkownika. Zaprezentowano edytor wierszowy ex, który leży u podstaw vi; pokazano też, jak w edytorze vi uruchamiać polecenia ex.
W części drugiej opisano rozszerzenia „standardowego" vi, dostępne w większości lub wszystkich jego klonach.
W rozdziale 8. (Zestawienie cech klonów edytora vi) omówiono edycję wielookienkową, interfejsy graficzne (GUI), rozszerzone wyrażenia regularne, ułatwienia edycji i inne.
W rozdziałach od 9. do 12. omawiano kolejno poszczególne klony edytora vi: nvi, elvis, vim i vile. Pokazano, jak używać zaimplementowanych w nich rozszerzeń i scharakteryzowano ich specyficzne cechy.
W części trzeciej zawierającej dodatki znajdują się przydatne zestawienia. W dodatku A zestawiono wszystkie polecenia vi i ex, uszeregowane według realizowanej funkcji.
Dodatek B zawiera alfabetyczną listę wszystkich poleceń ex.
W dodatku C zestawiono opcje używane w poleceniu set.
W dodatku D zestawiono opis problemów uwzględnionych w niniejszej książce.
Dodatek E opisuje miejsce edytora vi w środowisku UNIX i środowisku internetowym.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

5

Wprowadzenie do edytora ex

Po co w podręczniku edytora *vi* znajduje się rozdział dotyczący innego edytora? Otóż właściwie *ex* nie jest innym edytorem, to *vi* jest wizualną wersją ogólniejszego edytora wierszowego — właśnie *ex*. Niektóre polecenia *ex* mogą być przydatne podczas pracy z *vi*, gdyż pozwalają zaoszczędzić wiele czasu. Większość tych poleceń można wydawać, w ogóle nie opuszczając *vi*[1].

Wiadomo już, że pliki traktować można jako ciąg ponumerowanych wierszy. *ex* umożliwia stosowanie poleceń edycji bardziej dynamicznie. Łatwo jest poruszać się miedzy różnymi plikami i kopiować dane z jednego pliku do innego, robiąc to na różne sposoby. Można szybko operować na blokach tekstu większych niż jeden ekran, zaś podstawienia globalne umożliwiają podmienianie wszystkich wystąpień wzorca w pliku.

W tym rozdziale omówiono edytor *ex* i jego polecenia, czyli jak:

- poruszać się po pliku, korzystając z numerów wierszy,
- używać poleceń *ex* do kopiowania, przenoszenia i usuwania bloków tekstu,
- zapisywać pliki i ich części,
- pracować jednocześnie z wieloma plikami (odczytując dane i polecenia, poruszając się między plikami).

Polecenia ex

Na długo przed tym, zanim wymyślono *vi* lub w ogóle edytor ekranowy, ludzie komunikowali się z komputerem za pośrednictwem terminali drukujących, a nie monitorów. Numery wierszy były sposobem szybkiego wskazania potrzebnej części pliku, powstały wtedy edytory wierszowe służące do edycji tych plików. Programista czy inny użytkownik komputera zazwyczaj drukował odpowiedni wiersz (lub wiersze), wydawał polecenia edycyjne, po czym ponownie drukował zmodyfikowany tekst.

---

[1] *vile* różni się od pozostałych klonów tym właśnie, że wiele poleceń *ex* po prostu w nim nie działa. Nie będziemy tego za każdym razem przypominać, po informacje szczegółowe należy sięgnąć do rozdziału 12., gdzie omawiany jest edytor *vile*.

Obecnie już nikt nie używa terminali drukujących, ale niektóre polecenia edytora wierszowego *ex* nadal są przydatne użytkownikom znacznie doskonalszych edytorów ekranowych zbudowanych na bazie *ex*. Wprawdzie edytowanie pliku zwykle jest prostsze w *vi*, ale operacje wierszowe *ex* są wygodniejsze, głównie w sytuacji dokonywania wielu zmian w licznych miejscach w pliku.

W przypadku wielu poleceń omawianych w tym rozdziale jako parametr używa się nazwy pliku. Wprawdzie stosowanie spacji w tych nazwach jest możliwe, ale to bardzo zły zwyczaj. *ex* będzie miał problemy z interpretacją całego polecenia, zaś ewentualne korzyści nie są tego warte. Do rozdzielania poszczególnych słów można użyć podkreśleń, kresek czy kropek — całość będzie równie czytelna, a przysporzy znacznie mniej kłopotów.

Zanim użytkownik zacznie uczyć się na pamięć poleceń *ex* albo, co gorsza, postanowi ich w ogóle nie używać, można przyjrzeć się nieco samym edytorom wierszowym. Zobaczenie jak działa bezpośrednio wywołany *ex* może być pomocne w zrozumieniu składni bardziej skomplikowanych poleceń.

Można spróbować otworzyć znany już plik i wypróbować kilka poleceń *ex*. Tak jak wywołuje się *vi* od razu z nazwą pliku, tak samo można wywołać *ex*. Edytor ten na początku pokazuje informację o liczbie wierszy w pliku oraz wyświetla znak zachęty — dwukropek, np.:

```
$ ex practice
"practice" 6 lines, 320 characters
:
```

Trudno zobaczyć choć jeden wiersz z pliku, póki edytorowi *ex* nie zostanie wydane polecenie wyświetlenia tych wierszy.

Polecenia *ex* składają się z adresu wiersza (może to być po prostu numer wiersza) i samego polecenia, kończy je znak powrotu karetki. Jednym z podstawowych poleceń jest `p` — wyświetlenie na ekranie. Jeśli zatem wpisze się np. `1p`, można zobaczyć pierwszy wiersz pliku:

```
:1p
Edytor ekranowy pozwala
:
```

Tak naprawdę nawet to `p` można pominąć, gdyż podanie samego numeru wiersza też spowoduje jego wyświetlenie. Aby wydrukować więcej wierszy, należy podać zakres odpowiednich numerów, np. `1,3` — liczby rozdziela się przecinkiem:

```
:1,3
Edytor ekranowy pozwala
przewijać strony, przemieszczać kursor,
usuwać wiersze, wstawiać znaki itd.,
```

Jeśli podane zostanie polecenie, ale bez numeru wiersza, zakłada się, że polecenie dotyczy wiersza bieżącego. Wobec tego np. polecenie zastępujące jedno słowo innym (`s`) można zapisać tak:

```
:1
Edytor ekranowy pozwala
:s/ekranowy/wierszowy/
Edytor wierszowy pozwala
```

Zwróć uwagę, że zmodyfikowany wiersz po wykonaniu polecenia jest wyświetlany. Można też ten sam, co poprzednio efekt uzyskać następująco:

```
:1s/ekranowy/wierszowy/
Edytor wierszowy pozwala
```

Nawet jeśli polecenia *ex* wywoływane są z *vi* i nie będą wykonywane bezpośrednio, warto poświęcić kilka minut na poznanie samego edytora *ex*. Dzięki temu można zdać sobie sprawę, jak przekazywać edytorowi wierszowemu adresy wierszy, których dotyczyć ma polecenie.

Kiedy już kilka poleceń *ex* zostało wykonanych na pliku *practice*, można otworzyć ten sam plik w *vi* i zobaczyć go w dobrze już znanym trybie pełnoekranowym. Aby z *ex* przejść do *vi*, wystarczy wydać polecenie `:vi`.

Aby z kolei w *vi* wywołać *ex*, należy użyć specjalnego znaku — dwukropka — po którym podaje się polecenie i naciska RETURN. Aby zatem w edytorze *ex* przejść do odpowiedniego wiersza, wystarczy po dwukropku podać jego numer. Aby przejść do wiersza 6., należy w edytorze *vi* wpisać:

```
:6
```

Naciśnij RETURN.

W dalszym ciągu ćwiczeń będzie omawiane uruchamianie poleceń *ex* jedynie z poziomu *vi*.

Ćwiczenie — edytor ex

W wierszu poleceń UNIX wywołaj edytor *ex*, otwierając w nim plik *practice*:	`ex practice`
Pojawia się komunikat:	`"practice" 6 lines, 320 characters`
Przejdź do pierwszego wiersza i wyświetl go:	`:1`
Wyświetl wiersze od 1 do 3:	`:1,3`
Podmień słowo „ekranowy” w pierwszym wierszu:	`:1s/ekranowy/wierszowy`
Wywołaj edytor *vi*:	`:vi`
Przejdź do pierwszego wiersza:	`:1`

Zestaw pytań

✓ *Podczas edycji pliku w vi przypadkowo przeszedłeś do edytora ex.*

Polecenie `Q` w trybie poleceń *vi* wywołuje *ex*. Kiedy jest się w *ex*, można przejść do *vi*, wydając polecenie `vi`.

Edycja danych w ex

Wiele poleceń *ex* realizujących zwykłe operacje edycyjne ma swoje odpowiedniki w *vi* pozwalające te same edycje robić prościej. Oczywistym jest, że do usuwania słowa lub całego wiersza

używa się raczej `dw` czy `dd`, a nie `delete` z *ex*. Kiedy jednak modyfikacje mają dotyczyć wielu wierszy, wygodniejsze w użyciu będą polecenia *ex*. Umożliwiają one modyfikowanie dużych fragmentów tekstu jednym tylko poleceniem.

Poniżej zestawiono polecenia *ex* wraz z ich zapisem skrótowym. Pamiętaj, że w *vi* polecenia *ex* trzeba poprzedzać dwukropkiem. Można używać albo pełnego brzmienia polecenia, albo skrótu — co jest łatwiejsze do zapamiętania.

`delete`	`d`	usuwanie wierszy
`move`	`m`	przenoszenie wierszy
`copy`	`co`	kopiowanie wierszy
	`t`	kopiowanie wierszy (synonim `co`)

Jeśli wydawać by się mogło, że polecenia *ex* będą czytelniejsze, gdy ich części zostaną porozdzielane spacjami, można spacje stosować i np. rozdzielać nimi adresy, wzorce i polecenia. Nie można jednak użyć spacji wewnątrz wzorca lub na końcu polecenia podstawiania.

Adresy wierszy

W przypadku wszystkich poleceń edycyjnych *ex* konieczne jest podanie numerów modyfikowanych wierszy. Jeśli chodzi o polecenia `move` i `copy`, należy podać także miejsca docelowego przeniesienia czy kopiowania tekstu.

Adresy wierszy można wskazywać na kilka sposobów:

- jawnie podawać numery wierszy,
- korzystać z symboli opisujących położenie względem położenia bieżącego,
- korzystać ze wzorców wyszukiwania jako adresów.

Można zobaczyć teraz kilka przykładów.

Definiowanie zakresu wierszy

Za pomocą numerów wierszy można jawnie określić wiersz lub zakres wierszy. Adresy takie są nazywane adresami *bezwzględnymi*, np.:

`:3,18d`	Usuwa wiersze od 3. do 18.
`:160,224m23`	Przenosi wiersze od 160. do 224. za wiersz 23. (Działa jak połączenie w *vi* poleceń `delete` i `put`).
`:23,29co100`	Kopiuje wiersze 23. do 29. i wstawia za wiersz 100. (Jak `yank` i `put` w *vi*).

Aby ułatwić sobie edycję przy użyciu numerów wierszy, można je wyświetlać po lewej stronie ekranu. Polecenie:

```
:set number
```

lub jego skrócona postać:

```
:set nu
```

wyświetlają numery wierszy. Plik *practice* będzie miał wtedy następującą postać:

```
1  Edytor ekranowy pozwala
2  przewijać strony, przemieszczać kursor,
3  usuwać wiersze, wstawiać znaki
4  itd.,
```

Wyświetlone numery nie są zapisywane w pliku, nie są też drukowane. Wyświetlane są tylko do zakończenia pracy z *vi* lub do ich wyłączenia odpowiednią opcją `set`:

```
:set nonumber
```

lub:

```
:set nonu
```

Aby tymczasowo wyświetlić numery wierszy dla jakiejś ich grupy, można użyć znaku `#`, np.:

```
:1,10#
```

wyświetli numery wierszy od pierwszego do dziesiątego.

Jak to opisano już w rozdziale 3., do wyświetlenia numeru bieżącego wiersza można użyć polecenia CTRL–G. W ten sposób, wykorzystując odpowiednie polecenia ruchu, można odczytać numery wierszy początkowego i końcowego bloku tekstu.

Jeszcze innym sposobem identyfikowania wierszy jest użycie polecenia *ex* =:

`:=` wyświetla liczbę wierszy w pliku,

`:.=` wyświetla numer wiersza bieżącego,

`:/`*`wzorzec`*`/=`
wyświetla numer pierwszego wiersza zawierającego *`wzorzec`*.

Symbole adresowania wierszy

Można też użyć symboli oznaczających adresy wierszy. `.`(kropka) oznacza wiersz bieżący, zaś `$` — ostatni wiersz w pliku. `%` oznacza wszystkie wiersze z pliku, czyli równoważne jest zapisowi `1,$`. Symbole te można łączyć z adresami bezwzględnymi, np.:

`:.,$d` usuwa wiersze od bieżącego do końca pliku,

`:20,.m$`
przenosi wiersze od 20. do bieżącego na koniec pliku,

`:%d` usuwa wszystkie wiersze z pliku,

`:%t$` kopiuje wszystkie wiersze z pliku na jego koniec (w efekcie tworząc duplikat pliku).

Oprócz bezwzględnych adresów wierszy można też stosować adresy określane względem wiersza bieżącego. Symbole + i − działają jak zwykłe operatory arytmetyczne. Jeśli podane zostaną przed liczbą, spowodują dodanie lub odjęcie znajdującej się za nimi wartości, np.:

`:.,.+20d`
: usuwa wiersz bieżący i 20 następnych,

`:226,$m.-2`
: przenosi wiersze od 226. do końca pliku w miejsce dwa wiersze przed wierszem bieżącym,

`:.,+20#`
: wyświetla numery wierszy dla wiersza bieżącego i 20 następnych wierszy.

Zresztą w przypadku korzystania z operatorów + i − nie jest konieczne używanie kropki, gdyż właśnie od wiersza bieżącego zaczyna się zliczanie.

Jeśli za + lub − nie zostanie podana żadna liczba, zinterpretowane to zostanie jako `+1` i `−1`[2]. Analogicznie zapis ++ i −− odpowiednio rozszerzają zakres o kolejny wiersz itd. Operatorów + i − można też użyć w połączeniu z wzorcami wyszukiwania, co opisano w następnym punkcie.

Liczba `0` oznacza początek pliku i jest odpowiednikiem zapisu `1-`, pozwala przenosić lub kopiować dane na sam początek pliku przed pierwszym wierszem zawierającym tekst, np. polecenie:

`:-,+t0` skopiuje trzy wiersze (wiersz nad kursorem, wiersz z kursorem i jeszcze jeden następny wiersz) i wstawi je na początek pliku.

Wzorce wyszukiwania

Innym sposobem adresowania wierszy w *ex* jest użycie wzorców wyszukiwania, np.:

`:/`*`wzorzec`*`/d`
: usuwa następny wiersz zawierający *`wzorzec`*.

`:/`*`wzorzec`*`/+d`
: usuwa wiersz znajdujący się *pod* wierszem zawierającym *`wzorzec`* (zamiast samego + można byłoby zapisać `+1`).

`:/`*`wzorzec1`*`/,/`*`wzorzec2`*`/d`
: usuwa wszystkie wiersze od pierwszego, zawierającego *`wzorzec1`* do pierwszego, zawierającego *`wzorzec2`*.

`:.,/`*`wzorzec`*`/m23`
: pobiera tekst od bieżącego wiersza (`.`) do pierwszego wiersza zawierającego *`wzorzec`* i przenosi go za wiersz 23.

Zwróć uwagę na to, że *przed* i *za* wzorcami znajdują się ukośniki.

---

[2] W przypadku adresowania względnego nie należy oddzielać symboli plusa i minusa od znajdującej się za nimi liczby, np. `+10` oznacza „10 kolejnych wierszy”, ale `+ 10` oznacza już „11 kolejnych wierszy (bo `1 + 10`)”, a przecież nie o to chodziło.

Jeśli tekst jest usuwany według wzorca w *vi* i w *ex*, edytory te działają nieco inaczej. Zakładając, że plik *practice* zawiera następujące dane:

```
Edytor ekranowy pozwala przewijać strony,
przemieszczać kursor, usuwać wiersze,
wstawiać znaki itd., wyniki tych
operacji widoczne są od razu na ekranie.
```

Klawisze	**Wyniki**

`d/wyniki`	`Edytor ekranowy pozwala przewijać strony,`
	`przemieszczać kursor, wyniki tych`
	`operacji widoczne są od razu na ekranie.`

Edytor *vi* usuwa wszystkie dane od bieżącego położenia kursora do słowa *wyniki*, jednak reszta obu wierszy pozostaje bez zmian.

`:.,/wyniki/d`	`Edytor ekranowy pozwala przewijać strony,` `operacji widoczne są od razu na ekranie.`

Edytor *ex* usuwa cały wskazany zakres wierszy. W tym przypadku usunięty został wiersz bieżący oraz wiersz ze wzorcem[3] — oba zostały usunięte w całości.

Zmiana wiersza bieżącego

Czasami użycie w poleceniu adresu względnego może zaowocować niespodziewanymi wynikami. Można założyć np., że kursor jest w pierwszym wierszu i użytkownik chce wydrukować wiersz 100. i pięć następnych. Jeśli zostanie wpisane polecenie:

```
:100,+5 p
```

otrzyma się następujący komunikat o błędzie: „First address exceeds second”[4]. Polecenie nie zostało wykonane, gdyż drugi adres wyliczany jest względem bieżącego położenia kursora (czyli 1), więc polecenie to jest interpretowane następująco:

```
:100,6 p
```

Tymczasem chodzi o nakazanie użycia wiersza 100. jako wiersza bieżącego.

Aby *ex* tak zrozumiał nasze polecenie, zamiast przecinka należy użyć średnika; wtedy pierwszy adres zostanie potraktowany jako wiersz bieżący. Aby zrealizować omawiany przykład, należy wydać polecenie:

```
:100;+5 p
```

---

[3] Zakłada się, że operujemy na niezmienionym (przez poprzednie polecenie) tekście. Gdybyśmy kontynuowali pracę po usunięciu tekstu przez *vi*, tym razem usunięty zostałby tylko jeden wiersz — zawierający wzorzec, będący jednocześnie wierszem bieżącym — *przyp. tłum.*

[4] Pierwszy adres znajduje się dalej niż drugi — *przyp. tłum.*

Tym razem `+5` interpretowane będzie względem wiersza 100. Średnik jest też użyteczny w przypadku korzystania z adresowania za pomocą wzorca i adresowania bezwzględnego. Aby np. wyświetlić wiersz zawierający *wzorzec* wraz z 10 następnymi wierszami, należy użyć polecenia:

```
:/wzorzec/;+10 p
```

Wyszukiwanie globalne

Wiadomo już, jak używać w *vi* ukośnika do wyszukiwania wzorców. *ex* zawiera z kolei polecenie `g`, które umożliwia wyszukiwanie wzorca i wyświetlanie wierszy go zawierających. Polecenie `:g!` działa przeciwnie niż `:g`. `:g!` lub jego synonimu `:v` można użyć do wyświetlenia wierszy *nie* zawierających *`wzorca`*.

Polecenie `g` dotyczyć może wszystkich wierszy z pliku, można też ograniczyć zakres jego działania.

`:g/`*`wzorzec`*
: odnajduje ostatnie wystąpienie *`wzorca`* w pliku (i przenosi tam kursor),

`:g/`*`wzorzec`*`/p`
: odnajduje i wyświetla wszystkie wiersze zawierające *`wzorzec`*,

`:g!/`*`wzorzec`*`/nu`
: odnajduje i wyświetla wszystkie wiersze nie zawierające *`wzorca`*; wyświetla też numery odnalezionych wierszy,

`:60,124g/`*`wzorzec`*`/p`
: odnajduje i wyświetla wiersze od 60. do 124., które zawierają *`wzorzec`*.

Zgodnie z oczekiwaniami polecenia `g` można użyć do robienia podstawień globalnych. Temat ten zostanie omówiony w rozdziale 6.

Łączenie poleceń ex

Nie każde kolejne polecenie *ex* musi zaczynać się dwukropkiem — kreska pionowa („`|`”) jest separatorem poleceń, co pozwala wykonywać wiele poleceń w jednym wierszu *ex* (podobnie jak średnik pozwala łączyć w wierszu poleceń wiele instrukcji systemu UNIX). W przypadku używania `|` trzeba uważać na adresowanie. Jeśli jedno polecenie zmieni kolejność wierszy w pliku, następne będzie działało już na podstawie nowej numeracji. Oto przykłady:

`:1,3d | s/ihc/ich/`
: usuwa wiersze od 1. do 3. (wierszem bieżącym staje się początek pliku), następnie robi podstawienie w wierszu bieżącym (w chwili wydawania całego polecenia był to wiersz 4.),

`:1,5 m 10 | g/`*`wzorzec`*`/nu`
: przesuwa wiersze od 1. do 5. za wiersz 10., następnie wyświetla wszystkie wiersze zawierające *`wzorzec`* (jednocześnie numerując wyświetlane wiersze).

Pamiętaj, że spacje ułatwiają czytanie poleceń.

Zapisywanie plików i kończenie pracy

Wiadomo już, że polecenie *vi* `ZZ`, zapisując plik, kończy pracę edytora. Często jednak kończy się pracę za pomocą poleceń *ex*, gdyż umożliwiają one pełniejszą kontrolę nad tym, co się dzieje. Wspomniano już o nich — teraz można zobaczyć je dokładniej:

`:w` zapisuje bufor w pliku, lecz nie kończy pracy. Polecenia `:w` można (i należy) używać podczas sesji, aby ustrzec się przed utratą danych w wyniku awarii systemu lub poważnego błędu podczas edycji,

`:q` kończy pracę edytora (wracając do wiersza poleceń systemu UNIX),

`:wq` zapisuje plik i kończy pracę edytora. Zapis robiony jest bezwarunkowo — nawet jeśli nie zmieniono nic w pliku,

`:x` zapisuje plik i kończy pracę edytora. Plik jest zapisywany tylko wtedy, gdy był zmodyfikowany[5].

vi stara się chronić zarówno istniejące już w systemie pliki, jak i poczynione w buforze zmiany. Jeśli np. próbuje się zapisać bufor w miejsce istniejącego już pliku, wyświetlane jest ostrzeżenie. Analogicznie, jeśli plik zostanie zmodyfikowany, by zakończyć sesję *nie zapisawszy* zmian, *vi* wygeneruje komunikat o błędzie:

```
No write since last change[6].
```

Ostrzeżenia te pozwalają uniknąć kosztownych pomyłek. Czasem jednak zdarza się, że chciałbyś mimo wszystko wykonać dane polecenie. W tym celu za poleceniem należy dodać wykrzyknik:

```
:w!
:q!
```

Polecenie `:w!` pozwala też zapisać plik, który został otwarty w trybie tylko do odczytu (poleceniem `vi -R` lub `view`) — jednak pod warunkiem, że użytkownik posiada uprawnienia do pisania w danym pliku.

Polecenie `:q!` jest bardzo ważnym narzędziem umożliwiającym zakończenie pracy bez wpływania na zawartość pliku, niezależnie od ewentualnych zmian wprowadzonych podczas sesji[7]. Wszystkie zmiany zawartości bufora są odrzucane.

Zmiana nazwy bufora

Polecenie `:w` pozwala też zapisać cały bufor (czyli kopię edytowanego pliku) pod nową nazwą.

---

[5] Różnica między `:wq` i `:x` jest istotna, jeśli edytujesz kod źródłowy i używasz programu *make*, który w swoim działaniu bazuje na czasach modyfikacji plików.

[6] Nie zapisano ostatnich modyfikacji pliku — *przyp. tłum.*

[7] Pod warunkiem jednak, że nie używałeś (wcześniej) polecenia `:w` — *przyp. tłum.*

Załóżmy, że mamy plik *practice* zawierający 600 wierszy. Po otwarciu pliku robi się w nim mnóstwo poprawek. Chcąc skończyć pracę i tak potrzebne będą obie wersje pliku: stara i nowa. Aby zmodyfikowaną zawartość bufora zapisać pod nazwą *practice.new*, trzeba wydać polecenie:

```
:w practice.new
```

Stara wersja tekstu w pliku *practice* pozostanie niezmieniona (o ile nie używałeś wcześniej `:w`). Można zakończyć działanie edytora, wydając polecenie `:q`.

Zapisywanie części pliku

Czasem podczas edycji warto część tekstu zapisać w osobnym pliku. Można np. wpisać kody formatujące, które mają być potem stosowane jako nagłówki jeszcze wielu innych plików.

Połączenie adresowania wierszy *ex* z poleceniem `w` pozwala zapisywać tylko część bufora. Jeśli np. jest edytowany plik *practice*, to chcąc zapisać jego część jako *nowyplik*, można użyć polecenia:

`:230,$w nowyplik`
: zapisuje w pliku *nowyplik* wiersze od 230. do końca bufora,

`:.,600w nowyplik`
: zapisuje wiersze od bieżącego do 600. w pliku *nowyplik*.

Dołączanie do pliku nowych danych

Użycie operatora przekierowania lub dopisywania systemu UNIX pozwala dane do pliku dopisywać lub zastąpić nimi zawartość dotychczasową. Jeśli np. zostanie wpisane:

```
:1,10w nowyplik
```

i potem:

```
:340,$w >>nowyplik
```

to *nowyplik* zawierał będzie wiersze od 1. do 10. oraz od 340. do końca bufora.

Kopiowanie jednego pliku do drugiego

Czasami trzeba wstawić tekst lub dane właśnie wprowadzone do systemu do edytowanego pliku. W *vi* zawartość innego pliku można odczytać poleceniem *ex*:

```
:read nazwapliku
```

lub w zapisie skróconym:

```
:r nazwapliku
```

Polecenie to wstawia zawartość pliku *nazwapliku* począwszy od wiersza bezpośrednio za wierszem zawierającym kursor. Jeśli dane mają być wstawione w innym miejscu, przed poleceniem `read` lub `r` podaje się adres wiersza.

Na przykład edytowany jest plik *practice*, do którego można wczytać plik *data* znajdujący się w katalogu */home/tim*. Kursor należy umieścić nad miejscem, w które nowy plik ma być wstawiony i wpisać:

```
:r /home/tim/data
```

Cała zawartość */home/tim/data* zostanie wstawiona do *practice*, poczynając od wiersza pod kursorem.

Aby ten sam plik wczytać i wstawić za wierszem 185., trzeba wydać polecenie:

```
:185r /home/tim/data
```

Oto inne sposoby odczytywania plików:

`:$r /home/tim/data`
: wstawia plik na koniec pliku edytowanego,

`:0r /home/tim/data`
: wstawia plik na sam początek pliku edytowanego,

`:/wzorzec/r /home/tim/data`
: wstawia plik do pliku edytowanego za wierszem zawierającym *wzorzec*.

Edycja wielu plików

Polecenia *ex* umożliwiają przełączanie się między wieloma różnymi plikami. Zaletą takiego rozwiązania jest szybkość działania. Jeśli system używany jest przez wielu użytkowników, zakończenie pracy w *vi* i ponowne uruchomienie tego edytora wymaga czasu. Jeśli w ramach jednej sesji będzie edytowanych wiele kolejnych plików, to nie tylko będzie można szybciej pracować, ale zachowane zostaną wszystkie zdefiniowane skróty i sekwencje poleceń (szczegóły w rozdziale 7.), poza tym pozostanie zawartość buforów, dzięki czemu można przenosić tekst między plikami.

Wywoływanie vi z wieloma plikami

Przy pierwszym wywołaniu *vi* można podać nazwy więcej niż jednego pliku do edycji — wtedy polecenia *ex* pozwolą przełączać się między tymi plikami, np.:

```
$ vi plik1 plik2
```

powoduje edycję najpierw pliku *plik1*. Kiedy edycja tego pliku zostanie zakończona, polecenie `:w` spowoduje jego zapisanie, zaś `:n` udostępni do edycji kolejny plik — *plik2*.

Załóżmy, że mamy edytować dwa pliki: *practice* i *note*.

Klawisze

```
vi practice
note
```

Wyniki

```
Edytor ekranowy pozwala przewijać strony,
przemieszczać kursor, usuwać wiersze,
wstawiać znaki itd., wyniki tych
```

Otwarto dwa pliki, *practice* i *note*. Na ekranie pojawia się pierwszy z nich — powstają wszystkie potrzebne modyfikacje.

`:w`

```
"practice" 6 lines, 328 characters
```

Zapisano edytowany plik *practice*, wydając polecenie *ex* `w`, po czym naciska się RETURN.

`:n`

```
Szanowny Panie
Henshaw:
Dziękujemy za uprzejme...
```

Wywołano następny plik, czyli *note*, za pomocą polecenia *ex* `n`, wciśnięto RETURN — można przystąpić do edycji drugiego pliku.

`:x`

```
"note" 23 lines, 1343 characters
```

Zapisano drugi plik i ukończono sesję.

Użycie listy parametrów

Edytor *ex* umożliwia więcej niż tylko przechodzenie za pomocą `:n` do następnego pliku podanego w wierszu poleceń. Polecenie `:args` (skracane jako `:ar`) wyświetla pliki z wiersza poleceń, przy czym plik aktualnie edytowany ujęty jest w nawiasy kwadratowe.

Klawisze **Wyniki**

`vi practice note`

```
Edytor ekranowy pozwala przewijać strony,
przemieszczać kursor, usuwać wiersze,
wstawiać znaki itd., wyniki tych
```

Otwarto dwa pliki: *practice* i *note*. Na ekranie pojawia się pierwszy z nich.

`:args`

```
[practice] note
```

vi wyświetla listę parametrów z wiersza poleceń, bieżący plik ujęty jest w nawiasy kwadratowe.

Polecenie `:rewind` (lub `:rew`) powoduje, że plikiem bieżącym ponownie staje się pierwszy plik z listy. *elvis* i *vim* dysponują poleceniem `:last` powodującym przejście do ostatniego pliku podanego w wierszu poleceń.

Wczytywanie nowych plików

Nie trzeba od razu na początku pracy z edytorem podawać nazw wszystkich plików, które mają być poprawiane. W każdej chwili poleceniem `:e` można otworzyć inny plik. Chcąc w *vi* zacząć edycję innego pliku, najpierw trzeba zapisać plik bieżący (`:w`) i wydać polecenie:

`:e` *nazwapliku*

Załóżmy, że edytuje się plik *practice* i chce przejść do edycji *letter*, po czym wrócić do *practice*.

Klawisze	Wyniki
`:w`	`"practice" 6 lines, 328 characters`
	Poleceniem `w` zapisuje się plik *practice*, naciska RETURN. *practice* jest zapisywany na dysku, nadal pozostaje na ekranie. Teraz można już jednak przełączyć się do innego pliku.
`:e letter`	`"letter" 23 lines, 1344 characters`
	Poleceniem `e` wywołuje się plik *letter*, naciska RETURN i można przystąpić do edycji nowego pliku.

vi „pamięta" dwie nazwy plików jako nazwę bieżącą i alternatywną. Można się do nich odwoływać za pomocą symboli: `%` (symbol pliku bieżącego) i `#` (symbol pliku alternatywnego). `#` jest szczególnie użyteczny w połączeniu z poleceniem `:e`, gdyż umożliwia przełączanie się między dwoma plikami. W powyższym przykładzie do pierwszego pliku (*practice*) można byłoby wrócić poleceniem `:e #`. Za pomocą polecenia `:r #` można też wczytać plik *practice* do pliku bieżącego.

Jeśli nie zostanie zapisany najpierw plik bieżący, *vi* nie pozwoli przełączyć się poleceniami `:e` czy `:n` — chyba że doda się do tych poleceń wykrzyknik nakazujący wykonanie polecenia niezależnie od konsekwencji.

Jeśli np. powstaną jakieś modyfikacje w *letter*, z których chce się zrezygnować, aby wrócić do *practice*, należy wydać polecenie `:e! #`.

Użyteczne jest też polecenie umożliwiające odrzucenie wszystkich zmian i przywracające postać bieżącego pliku w wersji z chwili ostatniego zapisu:

```
:e!
```

Z kolei symbol `%` użyteczny jest głównie przy zapisywaniu zawartości bufora do nowego pliku, np. kilka stron wcześniej, w punkcie *Zmiana nazwy bufora*, pokazano, jak zapisać drugą wersję pliku *practice* za pomocą polecenia:

```
:w practice.new
```

Ponieważ `%` oznacza nazwę pliku bieżącego, powyższy wiersz można zastąpić następującym:

```
:w %.new
```

Przełączanie się między plikami w vi

Jako że wracanie do pliku poprzedniego zdarza się dość często, w celu jego realizacji nie jest konieczne uciekanie się do wiersza poleceń *ex*. Polecenie *vi* `^^` (klawisz karetki naciśnięty wraz z CTRL) powoduje przełączenie się do pliku poprzedniego. Użycie tego polecenia daje efekt taki sam, jak `:e #`. Tak jak w przypadku polecenia `:e` *vi* nie pozwoli przełączyć się, jeśli bieżący bufor nie został zapisany.

Przenoszenie danych między plikami

Kiedy użyje się bufora roboczego z jednoliterową nazwą, można wygodnie przenosić tekst z jednego pliku do innego. Nazwane bufory nie są czyszczone podczas ładowania nowego pliku poleceniem `:e`. W ten sposób, kopiując lub usuwając tekst z jednego pliku do dowolnej liczby nazwanych buforów i następnie wywołując polecenie `:e` *`nowyplik`*, można wklejać tekst z jednego pliku do innego — dane są w ten sposób przenoszone lub kopiowane.

W poniższym przykładzie przedstawiono przenoszenie tekstu między plikami.

Klawisze **Wyniki**

`"f4yy`

```
Edytor ekranowy pozwala przewijać strony,
przemieszczać kursor, usuwać wiersze,
wstawiać znaki itd., wyniki tych
operacji widoczne są od razu na ekranie.
```

Skopiowano cztery wiersze do bufora roboczego `f`.

`:w`

```
"practice" 6 lines, 238 characters
```

Zapisano plik.

`:e letter`

```
Szanowny Panie
Henshaw:
Wydaje mi się, iż może
zainteresować Pana fakt, że:
Łączę wyrazy szacunku,
```

Poleceniem `:e` przechodzi się do pliku *letter*. Umieszcza się kursor w miejscu, za którym wstawiony ma być kopiowany tekst.

`"fp`

```
Szanowny Panie
Henshaw:
Wydaje mi się, iż może
zainteresować Pana fakt, że:
Edytor ekranowy pozwala przewijać strony,
przemieszczać kursor, usuwać wiersze,
wstawiać znaki itd., wyniki tych
operacji widoczne są od razu na ekranie.
Łączę wyrazy szacunku,
```

Umieszczono skopiowany tekst z bufora `f` zaraz pod kursorem.

Innym sposobem przenoszenia tekstu między plikami jest użycie poleceń *ex*: `:ya` (kopiowanie) i `:pu` (wstawianie). Polecenia te działają tak samo jak odpowiedniki z *vi*: `y` i `p`, ale umożliwiają użycie znanego z *ex* adresowania.

I tak np. polecenie:

```
:160,224ya a
```

skopiuje do bufora `a` wiersze od 160. do 224. Następnie poleceniem `:e` przechodzi się do nowego pliku, w którym skopiowany tekst ma być umieszczony. Kursor należy ustawić w miejscu, za którym mają się znaleźć wiersze, po czym wpisać:

```
:pu a
```

Spowoduje to wstawienie za bieżącym wierszem zawartości bufora `a`.