Работа с текстом на максимальной скорости и мощности

7 издание

7 новых глав о Vim

Изучаем vi и Vim *редакторы*

СИМВОЛ®

O'REILLY®

Арнольд Роббинс,
Элберт Ханна и Линда Лэмб

Learning the vi and Vim Editors

Seventh Edition

Arnold Robbins, Elbert Hannah and Linda Lamb

O'REILLY®

Изучаем редакторы vi и Vim

Седьмое издание

Арнольд Роббинс, Элберт Ханна
и Линда Лэмб

Санкт-Петербург — Москва
2013

Арнольд Роббинс, Элберт Ханна и Линда Лэмб

Изучаем редакторы vi и Vim, 7-е издание

Перевод И. Аввакумова

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*В. Синицын*
Редактор	*Ю. Бочина*
Корректор	*С. Беляева*
Верстка	*Д. Орлова*

Роббинс А., Ханна Э., Лэмб Л.

Изучаем редакторы vi и Vim, 7-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2013. – 512 с., ил.

ISBN 978-5-93286-200-1

На протяжении 30 лет vi оставался стандартом для UNIX и Linux, а эта книга была главным пособием по vi. Однако сейчас UNIX уже не тот, что был 30 лет назад, и книга тоже не стоит на месте. Седьмое издание существенно расширено и включает подробную информацию о Vim – самом популярном клоне vi. Доступный стиль изложения сделал эту книгу классикой. Она незаменима, поскольку знание vi или Vim – обязательное условие, если вы работаете в Linux или UNIX.

Вы познакомитесь как с основами, так и с продвинутыми средствами, такими как интерактивные макросы и скрипты, расширяющие возможности редактора. Вы научитесь быстро перемещаться в vi, использовать буферы, применять глобальную функцию поиска и замены vi, настраивать vi и запускать команды UNIX, использовать расширенные текстовые объекты Vim и мощные регулярные выражения, редактировать в нескольких окнах и писать скрипты в Vim, использовать все возможности графической версии Vim (gvim), применять такие усовершенствования Vim, как подсветка синтаксиса и расширенные теги. Помимо Vim рассматриваются и другие клоны vi: nvi, elvis и vile.

ISBN 978-5-93286-200-1
ISBN 978-0-596-52983-3 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 380-5007, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Подписано в печать 12.12.2012. Формат 70×100 1/16.
Печать офсетная. Объем 32 печ. л.

Посвящается моей жене Мириам
за любовь, терпение и поддержку.

Арнольд Роббинс
(шестое и седьмое издания)

Оглавление

Предисловие

Редактирование текстов – одна из наиболее востребованных задач в любой компьютерной системе, а vi – один из наиболее полезных стандартных текстовых редакторов. С помощью vi можно создавать новые текстовые файлы или редактировать имеющиеся.

Как и многие классические программы, разработанные во времена становления UNIX, vi имеет репутацию сложной в управлении программы. Создавая улучшенный клон vi под названием Vim (от «vi improved»), Брам Моленар (Bram Moolenaar) сделал многое, чтобы устранить причины такого впечатления. Vim содержит многочисленные усовершенствования, визуальные подсказки и справочную систему. Он стал, вероятно, самой популярной версией vi, поэтому в седьмом издании этой книги ему посвящено семь новых глав в части II «Vim». Однако существует множество других клонов vi, три из которых мы рассмотрим в части III «Другие клоны vi».

План книги

Книга разбита на 4 части и состоит из 18 глав и 4 приложений.

Часть I «Базовый и продвинутый vi» поможет быстро начать работу с vi, а также получить углубленные навыки, позволяющие использовать его более эффективно.

В главе 1 «Текстовый редактор vi» описываются некоторые простые команды vi, с которых можно начать знакомство с программой. Попрактикуйтесь в них, пока не освоите достаточно хорошо. Глава 2 «Простое редактирование» познакомит с некоторыми элементарными инструментами редактирования.

Однако функциональные возможности vi выходят далеко за рамки обычной обработки текста. Большое разнообразие команд и опций позволит сократить существенную часть рутинной работы. В главе 3 «Быстрое перемещение» и главе 4 «За рамками основ» уделяется внимание более простым способам выполнения задач. При первом чтении вы получите, по крайней мере, представление о возможностях vi и о том, какие команды можно приспособить под ваши нужды. Впоследствии можно вернуться к этим главам для более детального изучения.

Глава 5 «Введение в редактор ex», глава 6 «Глобальная замена» и глава 7 «Продвинутое редактирование» посвящены средствам, позволяющим переложить часть бремени редактирования на плечи компьютера. Вы познакомитесь со строковым редактором `ex`, лежащим в основе `vi`, и узнаете, как из `vi` обращаться к командам `ex`.

Глава 8 «Представляем клоны vi» знакомит с расширениями, доступными в четырех клонах `vi`. Здесь описываются многооконное редактирование, графические интерфейсы, расширенные регулярные выражения, функции, облегчающие редактирование, и некоторые другие особенности, тем самым показывая план оставшейся части книги. Кроме того, в этой главе есть ссылка на исходный код первоначального `vi`, который может быть легко скомпилирован на современных UNIX-системах (включая GNU/Linux).

Часть II «Vim» описывает Vim – наиболее популярный на сегодняшний день клон `vi`.

В главе 9 «Vim (vi Improved): введение» дается общая информация о Vim, в том числе, где взять бинарные версии для наиболее популярных операционных систем и каковы различные варианты применения Vim.

В главе 10 «Главные улучшения Vim по сравнению с vi» описываются наиболее существенные улучшения в Vim по сравнению с `vi`, такие как встроенная справка, управление инициализацией, дополнительные команды перемещения и расширенные регулярные выражения.

Глава 11 «Многооконность в Vim» уделяет внимание многооконному редактированию, которое, возможно, является наиболее значимым дополнением к стандартному `vi`. В главе рассматриваются все подробности создания и использования нескольких окон.

В главе 12 «Скрипты Vim» рассматривается язык команд Vim, который позволит вам писать скрипты, чтобы приспособить Vim под ваши нужды. Простота использования Vim «из коробки» во многом объясняется огромным количеством скриптов, написанных другими пользователями и включенных в дистрибутив Vim.

В главе 13 «Графический Vim (gvim)» рассматривается Vim в современных графических окружениях, например тех, которые являются стандартными на современных коммерческих UNIX-системах, в GNU/Linux и других UNIX-системах, а также в MS Windows.

Глава 14 «Улучшения Vim для программистов» сосредоточена на использовании Vim в качестве редактора для программистов, оставляя за рамками его возможности обычного редактирования текста. Особенно ценными являются функции сворачивания кода и редактирования планов-схем, умные отступы, подсветка синтаксиса и ускорение цикла «редактирование-компиляция-отладка».

Глава 15 «Другие полезности в Vim» является отчасти собирательной, так как в ней охватывается множество интересных вопросов, не вошедших в предыдущие главы.

Часть III «Другие клоны vi» посвящена трем другим популярным клонам vi: nvi, elvis и vile.

Глава 16 «nvi: новый vi», глава 17 «elvis» и глава 18 «vile: vi как Emacs» охватывают различные клоны vi: nvi, elvis и vile. В главах обсуждается, как использовать их расширения, и описываются особенности каждого из них.

Часть IV «Приложения» содержит полезные справочные материалы.

В приложении A «Редакторы vi, ex и Vim» перечисляются все команды vi и ex, отсортированные по функциям. Кроме того, приводится список команд ex в алфавитном порядке, а также некоторые команды vi и ex из Vim.

Приложение B «Установка опций» содержит список опций команды set для vi и всех четырех его клонов.

В приложении C «Возможные проблемы» обсуждаются возможные проблемы при работе с vi и его клонами, а также способы их устранения.

В приложении D «vi и Интернет» рассказывается о месте, которое занимает vi в более широкой культуре UNIX и Интернета.

Способ представления материала

Наша задача – дать хороший обзор материала, который поможет новичкам изучить vi. Освоение нового редактора, особенно редактора со всеми возможностями vi, может показаться непреодолимой задачей. Мы сделали попытку представить основные концепции и команды в логичной и удобочитаемой форме.

После изложения общих основ vi, применимых везде, мы переходим к более глубокому рассмотрению Vim. Картину завершает обзор nvi, elvis и vile. Последующие разделы описывают условные обозначения, используемые в этой книге.

Обсуждение команд vi

Здесь вы найдете краткое описание основной идеи, предшествующее узкоспециализированным разделам. Затем приводятся примеры применения этой команды в каждом конкретном случае наряду с ее описанием и синтаксисом использования.

Условные обозначения

В описании синтаксиса и в примерах данные для ввода набраны шрифтом MonoCondensed. То же касается названий команд, имен файлов и опций. Переменные (то есть то, что не будет вводиться буквально, а будет заменяться при вводе команды на нужное значение) набраны *курсивным MonoCondensed*. Квадратные скобки означают, что переменная является необязательной. Например, в строке с синтаксисом:

```
vi [filename]
```

filename будет заменено на реальное имя файла. Скобки говорят о том, что команда vi может вызываться без указания имени файла. Сами скобки вводить не надо.

Некоторые примеры показывают результат работы команд, вводимых в командной строке UNIX. В таких примерах то, что вы реально вводите, набрано шрифтом **MonoCondensed Bold**, чтобы отличать это от отклика системы. Например:

```
$ ls
ch01.xml ch02.xml ch03.xml ch04.xml
```

В примерах кода *курсив* обозначает комментарий, который вводить не надо. В основном тексте ***курсивом*** выделены специальные термины либо то, на что следует обратить внимание.

Следуя общепринятым соглашениям по документации UNIX, ссылки вида *printf(3)* указывают на электронное справочное руководство (которое можно получить посредством команды man). Этот пример ссылается на страницу функции printf() в разделе 3 этого руководства (в большинстве систем нужно ввести man 3 printf, чтобы увидеть ее).

Клавиши

На протяжении всей книги вы встретите таблицы, содержащие команды vi и результаты их работы:

Клавиши	Результаты
ZZ	"practice" (New file) 6 lines, 320 characters Введите команду выхода с сохранением – ZZ. Ваш файл будет сохранен как обычный файл UNIX.

В этом примере команда ZZ приведена в левом столбце. В рамке справа содержится строка (или несколько строк) экрана, показывающая результат выполнения команды. Положение курсора показано инверсией фона и цвета символов. В этом случае, поскольку ZZ сохраняет файл и выходит из программы, после записи файла вы увидите строку состояния; положение курсора не показано. Под рамкой расположено объяснение команды и ее результата.

Иногда к командам vi обращаются при помощи одновременного нажатия клавиши CTRL с другой клавишей. В основном тексте такая комбинация клавиш обычно записывается так: CTRL-G. В примерах кода в таких случаях перед названием клавиши ставят знак вставки (^), например ^G означает, что при нажатии на G нужно удерживать нажатой клавишу CTRL.

Возможные проблемы

В тех разделах, где у вас могут возникнуть затруднения, содержится перечень возможных ошибок при выполнении тех или иных задач. Вы можете просмотреть эти ошибки и вернуться к ним, когда столкнетесь с подобной проблемой на практике. Чтобы упростить доступ к перечню возможных ошибок, они приведены также в приложении C.

Что нужно знать

Мы полагаем, что вы уже прочли «Learning the Unix Operating System» (O'Reilly) или какое-нибудь другое введение в UNIX. Вы должны знать, как:

- осуществлять вход в систему и выход из нее;
- вводить команды UNIX;
- менять каталоги;
- выводить список файлов в каталоге;
- создавать, копировать и удалять файлы.

Знакомство с `grep` (global search program, программа глобального поиска) и символами подстановки также будет полезным.

Замечания и вопросы

Свои замечания и вопросы по этой книге отправляйте, пожалуйста, издателю:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

Технические вопросы и замечания о книге присылайте на электронный адрес:

bookquestions@oreilly.com

Веб-сайт этой книги содержит примеры, список ошибок и планы будущих изданий. Страница располагается по адресу:

http://www.oreilly.com/catalog/9780596529833

Для получения дополнительной информации о наших книгах, конференциях, программных продуктах, информационных центрах и о сети O'Reilly Network посетите наш веб-сайт:

http://www.oreilly.com

Safari® Books Online

Если вы видите значок Safari® Books Online на вашей любимой технической книге, это означает, что книга доступна онлайн посредством O'Reilly Network Safari Bookshelf.

Safari предлагает лучшее решение, нежели электронные книги. Это виртуальная библиотека, где вы можете легко найти любую из тысяч технических книг, копировать и вставлять фрагменты кода, скачивать отдельные главы и быстро находить ответы, когда вам нужна самая точная и актуальная информация. Попробуйте это бесплатно на *http://safari.oreilly.com*.

О предыдущих изданиях

В пятом издании книги под названием «Learning the vi Editor» команды редактора `ex` рассматривались более подробно. В главах 5, 6 и 7 с помощью большого количества примеров разъяснялись сложные функции `ex` и `vi` в таких темах, как синтаксис регулярных выражений, глобальные замены, файлы `.exrc`, сокращения слов, отображение клавиш и скрипты редактирования. Несколько примеров было взято из журнала «Unix World». Вальтер Зинц (Walter Zintz) написал учебник о `vi` из двух частей[1], рассказывающий о нескольких неизвестных нам вещах и содержащий множество грамотных примеров, которые иллюстрируют уже рассмотренные нами функции. Рэй Шварц (Ray Swartz) в одной из своих заметок также поделился полезными советами[2]. Мы благодарны им за идеи, изложенные в этих материалах.

Шестое издание «Learning the vi Editor» содержало ознакомительный обзор четырех доступных «клонов», то есть редакторов со схожими принципами работы. Многие из них содержали улучшения по сравнению с `vi`. Следовательно, можно говорить о существовании «семейства» редакторов `vi`. Издание в равной степени уделяло внимание `nvi`, `Vim`, `elvis` и `vile` с целью познакомить читателя с этими клонами.

Также в шестом издании было добавлено следующее:

- В основной текст внесены многочисленные исправления и дополнения.
- В конце каждой главы приведена сводка соответствующих команд.

---

[1] Две статьи Вальтера Зинца: «`vi` Tips for Power Users», *Unix World*, апрель 1990 и «Using `vi` to Automate Complex Edits», *Unix World*, май 1990. (В приложении D указаны веб-адреса этих статей.)

[2] «Answers to Unix», *Unix World*, август 1990.

- Новые главы, посвященные каждому из клонов vi, функции и/или расширения, общие для двух или более клонов, и многооконное редактирование.
- Главы, рассказывающие немного об истории, целях, уникальных особенностях, способах установки каждого из клонов vi.
- Новое приложение, где говорится о месте vi в более широкой культуре UNIX и Интернета.

Предисловие к седьмому изданию

Седьмое издание «Learning the vi and Vim Editors» содержит все лучшее от шестого. Время показало, что именно Vim является самым популярным клоном vi, так что в этом издании обзор данного редактора существенно расширен (ему даже отведено место в названии книги). Но чтобы книга оставалась полезной для как можно большего числа читателей, мы оставили и обновили материалы о nvi, elvis и vile.

Что нового

В этом издании появились следующие новые материалы:

- Внесены исправления в основном тексте.
- Семь новых глав, в которых всесторонне рассматривается Vim.
- Материал про современное состояние nvi, elvis и vile.
- Два приложения из предыдущего издания, содержащие справку по ex и vi, были объединены в одно, которое теперь содержит еще и дополнительные материалы по Vim.
- Обновлены другие приложения.

Версии

При тестировании различных функций vi использовались следующие программы:

- Версия vi для Solaris как наиболее близкая к версии vi в UNIX.
- Версия программы nvi 1.79 Кейта Бостича (Keith Bostic).
- Версия программы elvis 2.2 Стива Киркендалля (Steve Kirkendall).
- Версия Vim 7.1 Брама Моленара (Bram Moolenaar).
- Версия vile 9.6 Кевина Бейттнера (Kevin Buettner), Тома Дики (Tom Dickey) и Пола Фокса (Paul Fox).

Благодарности для шестого издания

В первую очередь благодарю свою жену Мириам за заботу о детях, пока я работал над книгой, в особенности во время «волшебных часов» непосредственно перед обедом. Я должен ей огромное количество тишины и мороженого.

Пол Манно (Paul Manno) из Технического колледжа компьютерных технологий Джорджии (Georgia Tech College of Computing) оказал неоценимую помощь в усмирении моих программ печати. Лен Мюлнер (Len Muellner) и Эрик Рэй (Erik Ray) из O'Reilly & Associates помогали с программами для SGML. Макрос `vi`, написанный Джерри Пиком (Jerry Peek), оказался бесценным.

Хотя при подготовке нового и исправления старого материала использовались все упомянутые программы, большая часть редактирования осуществлялась в Vim версий 4.5 и 5.0 под GNU/Linux (Red Hat 4.2).

Я благодарен Кейту Бостичу (Keith Bostic), Стиву Киркендаллю (Steve Kirkendall), Браму Моленару (Bram Moolenaar), Полу Фоксу (Paul Fox), Тому Дики (Tom Dickey) и Кевину Бейттнеру (Kevin Buettner), проверявшим книгу и снабдившим меня важными материалами для глав с 8 по 12 (номера этих глав соответствуют шестому изданию).

Без электричества, вырабатываемого энергетической компанией, работать на компьютере невозможно. Однако когда электричество есть в розетке, вы перестаете думать о нем. Точно так же и при написании книги – без редактора у вас ничего не получится, однако когда он делает свою работу, о нем легко забыть. Гиги Эстабрук (Gigi Estabrook) из O'Reilly – это просто жемчужина. Работать с ней одно удовольствие. Я высоко ценю все, что она делала и продолжает делать для меня.

И наконец, много благодарностей команде O'Reilly & Associates.

Арнольд Роббинс (Arnold Robbins)
Ra'anana, Израиль, июнь 1998

Благодарности для седьмого издания

И снова Арнольд благодарит свою жену Мириам за любовь и поддержку. Размер долга в виде тишины и мороженого продолжает расти. Кроме того, он благодарен Дж.Д. «Илиаду» Фрейзеру (J.D. «Illiad» Frazer) за прекрасные комиксы *User Friendly*[1].

Элберт хотел бы поблагодарить Анну, Келли, Бобби и своих родителей за проявленный интерес к его работе в нелегкое время. Их энтузиазм был заразительным и бесценным.

Благодарим Кейта Бостича (Keith Bostic) и Стива Киркендалля (Steve Kirkendall) за вклад в доработку глав об их редакторах. Том Дики (Tom Dickey) внес значительный вклад в подготовку главы о `vile` и таблицы опций команды `set` в приложении B. Брам Моленар (Bram Moolenaar), автор Vim, в этот раз также выполнил проверку всей книги. Роберт П. Дж. Дэй (Robert P.J. Day), Мэтт Фрай (Matt Frye), Юдит Майерсон (Judith Myerson) и Стивен Фиггинс (Stephen Figgins) дали ценные замечания по всему тексту.

Арнольд и Элберт хотят поблагодарить Энди Ора (Andy Ora) и Изабель Кункле (Isabel Kunkle) за редакторскую работу, а также всех сотрудников O'Reilly Media.

Арнольд Роббинс (Arnold Robbins)
Ноф Айалон, Израиль, 2008

Элберт Ханна (Elbert Hannah)
Килдир, Иллинойс, США, 2008

---

[1] Если вы ничего не слышали о *User Friendly*, зайдите на *http://www.user-friendly.org*.

I

Базовый и продвинутый vi

Часть I поможет быстро начать работу с `vi`, а также получить углубленные навыки, позволяющие использовать `vi` более эффективно. Материал охватывает оригинальный базовый `vi`, а рассматриваемые команды можно использовать в любой его версии; последующие главы посвящены популярным клонам `vi`. Часть I состоит из следующих глав:

- Глава 1 «Текстовый редактор vi»
- Глава 2 «Простое редактирование»
- Глава 3 «Быстрое перемещение»
- Глава 4 «За рамками основ»
- Глава 5 «Введение в редактор ex»
- Глава 6 «Глобальная замена»
- Глава 7 «Продвинутое редактирование»
- Глава 8 «Представляем клоны vi»

1

Текстовый редактор vi

UNIX[1] содержит множество редакторов, которые могут обрабатывать текстовые файлы, будь то файлы, содержащие данные, исходный код или обычный текст. Таковыми являются, например, строковые редакторы `ed` и `ex`, отображающие на экране лишь одну строку из файла. Кроме того, есть экранные редакторы, например `vi` и Emacs, у которых на экране терминала отображается часть файла. Текстовые редакторы, основанные на X Window System, также широко доступны и становятся все популярнее. Как в GNU Emacs, так и в его потомке XEmacs допускается использование нескольких X-окон; двумя другими интересными вариантами являются редакторы `sam` и Acme от Bell Labs. В Vim также доступен интерфейс, основанный на X.

`vi` – это наиболее полезный стандартный текстовый редактор в вашей системе. (`vi` – это сокращение от «**vi**sual editor», то есть визуальный редактор; произносится как «ви-ай». Это хорошо проиллюстрировано на рис. 1.1.) В отличие от Emacs, он доступен практически в неизменном виде на любой современной системе UNIX, тем самым являясь подобием *лингва-франка*[2] текстового редактирования. То же можно сказать

---

[1] В настоящее время термин «UNIX» включает как коммерческие системы, выведенные из оригинальной кодовой базы UNIX, так и UNIX-подобные системы с доступным исходным кодом. Примерами первых являются Solaris (хотя проект OpenSolaris придал ему некоторое «промежуточное» положение в такой схеме классификации. – *Примеч. науч. ред.*), AIX и HP-UX, а вторых представляют GNU/Linux и разнообразные системы, основанные на BSD. Сказанное в этой книге применимо ко всем системам такого типа, если нет специальной оговорки.

[2] GNU Emacs стал универсальной версией Emacs. Единственная проблема в том, что он не является стандартной частью большинства коммерческих UNIX-систем, поэтому его следует найти и установить самостоятельно.

про ed и ex, однако пользоваться экранными редакторами намного удобнее (настолько удобнее, что строковые редакторы сейчас практически не используются). В экранном редакторе можно пролистывать страницы, перемещать курсор, удалять строки, вставлять символы и многое другое, при этом вы сразу видите результат своих действий. Экранные редакторы стали популярными благодаря возможности вносить изменения при чтении файла, как если бы вы редактировали распечатанный экземпляр, только быстрее.

Рис. 1.1. *Произносите vi правильно*

Многим новичкам vi кажется непонятным и громоздким – вместо того чтобы использовать комбинации клавиш для обработки текста и позволить вам печатать обычным образом, в этом редакторе каждой клавише назначена своя команда. Когда ожидается вызов команды нажатием клавиши, говорят, что vi находится в *командном режиме* (*command mode*). Для того чтобы начать печатать собственно текст на экране, необходимо перейти в специальный *режим вставки* (*insert mode*). Следует отметить, что в vi огромное множество команд.

Однако начав освоение vi, вы обнаружите, что он хорошо продуман. Чтобы заставить его выполнить сложную работу, порой достаточно нескольких нажатий клавиш. По мере дальнейшего изучения vi вы узнаете комбинации клавиш, которые все больше и больше работы по редактированию будут передавать компьютеру, то есть туда, где ей и место.

vi (как и любой текстовый редактор) не является текстовым процессором типа «what you see is what you get» (что видишь, то и получишь). Если вам нужно создать отформатированный документ, то придется писать коды, понимаемые другой программой, которая и будет управлять видом печатаемой копии. Так, если у нескольких абзацев должен быть отступ, поместите специальный код там, где начинается и заканчивается отступ. Коды форматирования позволят вам экспериментировать или менять внешний вид печатаемых файлов. Во многих случаях они дают вам намного больше контроля над внешним видом документа,

нежели текстовый процессор. UNIX поддерживает пакет форматирования `troff`[1]. Популярными и широкодоступными альтернативами являются издательские системы TEX и LATEX[2].

(В действительности, `vi` поддерживает простые механизмы форматирования. Например, он может сам переносить слова при достижении конца строки или делать автоматические отступы у новых строк. Кроме того, в Vim версии 7 есть автоматическая проверка орфографии.[3])

Как и при любой деятельности, чем больше вы редактируете, тем быстрее освоите редактор и тем выше будет ваша производительность. А когда вы изучите все возможности `vi`, то вряд ли захотите вернуться к «более простым» редакторам.

В чем же заключается процесс редактирования? Во-первых, вам может понадобиться *вставить* (*insert*) текст (например, пропущенное или новое слово либо пропущенное предложение) или, наоборот, *удалить* (*delete*) текст (отдельный символ или целый абзац). Также должна быть возможность *менять* (*change*) буквы или слова (чтобы исправить опечатки или изменить термин). Возможно, вам придется *переносить* (*move*) текст из одной части файла в другую. Кроме того, порой требуется *копировать* (*copy*) текст, чтобы создать его дубликат в другой части файла.

В отличие от многих текстовых процессоров, изначальным режимом, или режимом «по умолчанию», в `vi` является командный режим, в котором сложные интерактивные правки можно выполнять нажатием всего лишь нескольких клавиш. (А для вставки неформатированного текста просто выполните любую из нескольких команд «вставки», после чего начинайте набор.)

В качестве базовых команд используются один или несколько символов. Например:

`i`

вставка (insert).

`cw`

изменить слово (change word).

---

[1] `troff` предназначен для лазерных принтеров и наборных машин и является «братом-близнецом» `nroff` – пакета форматирования для строчных принтеров и терминалов. Оба понимают один и тот же набор команд. Следуя общепринятому в UNIX соглашению, мы называем `troff` оба пакета. В настоящее время все, кто использует `troff`, работают с его GNU-версией, `groff`. За более подробной информацией обратитесь на сайт *http://www.gnu.org/software/groff/*.

[2] Для получения информации о TEX и LATEX посетите сайты *http://www.ctan.org* и *http://www.latex-project.org* соответственно.

[3] Vim «из коробки» также может делать выключку текста по левому, правому краю или по центру. – *Прим. науч. ред.*

Используя буквы в качестве команд, вы сможете редактировать файл с огромной скоростью. Вам необязательно запоминать все сочетания функциональных клавиш или растягивать пальцы, чтобы нажать неудобную комбинацию клавиш. Вам никогда не потребуется убирать руки с клавиатуры или путаться в многоуровневых меню! Многие команды можно запомнить по первым двум буквам их названий, и почти все команды следуют одинаковым правилам и связаны друг с другом.

Вообще говоря, команды `vi`:

- Зависят от регистра клавиши (прописная и строчная буквы соответствуют разным командам: `I` – не то же, что `i`).
- Не отображаются на экране, когда их вводят.
- Не требуют нажатия ENTER после ввода команды.

Также есть группа команд, которые отображаются в нижней строке экрана. Они начинаются со специальных символов. Косая черта (/) и знак вопроса (?) запускают команды поиска; об этом рассказано в главе 3. Все команды `ex` начинаются с двоеточия (:); они используются в строковом редакторе `ex`. Этот редактор доступен при работе в `vi`, поскольку `ex` является базовым редактором, а `vi` – это просто его «визуальный» режим. Команды и понятия `ex` обсуждаются в главе 5, но уже в этой главе вы узнаете о команде `ex` для выхода из файла без сохранения.

Краткая историческая справка

Перед погружением во все тонкости `vi` полезно понять, какими глазами он «смотрит» на ваше окружение. В частности, это поможет осмыслить кажущиеся туманными сообщения `vi` об ошибках, а также разобраться, насколько клоны `vi` развились по сравнению с оригиналом.

`vi` восходит к тем временам, когда пользователи работали за терминалами, которые последовательно подсоединялись к центральным компьютерам. По всему миру были распространены сотни разновидностей терминалов. Каждый из них выполнял одни и те же действия (очистка экрана, перемещение курсора и т. п.), однако команды для управления ими были различными. Кроме того, система UNIX позволяет выбирать, какие символы использовать для забоя, генерации сигнала прерывания и других команд, применяемых на последовательных терминалах, например подавления и возобновления вывода. Такие функции управлялись (и до сих пор управляются) командой `stty`.

Первоначальная версия `vi`, созданная в Калифорнийском университете в Беркли (University of California, Berkeley, UCB), абстрагировала информацию об управлении терминалом из кода (который было сложно изменить) в текстовую базу данных возможностей терминала (которую изменить было легко), поддерживаемую библиотекой `termcap` (от **term**inal **cap**abilities). В начале 80-х в System V были внедрены база данных, содержащая двоичную информацию о терминалах, и библиотека `terminfo`

(от **term**inal **info**rmation). Эти две библиотеки были в основном функционально эквивалентными. Чтобы сообщить vi, каким именно терминалом вы пользуетесь, необходимо было установить переменную окружения TERM. Обычно это проделывалось файлом запуска оболочки, таким как .profile или .login.

В настоящее время все пользуются эмуляторами терминалов в графическом окружении (например, xterm). Как правило, система сама заботится о задании переменной TERM. (Конечно, можно вызвать vi и в неграфической консоли вашего ПК. Это может очень сильно помочь при восстановлении системы в однопользовательском режиме. Хотя осталось немного людей, которые предпочли бы так работать на регулярной основе.) Скорее всего, для повседневного использования вы выберите графическую версию vi, например Vim или один из других клонов. В Microsoft Windows или Mac OS X он, возможно, будет запускаться по умолчанию. Однако когда вы запускаете vi (или какой-нибудь другой столь же винтажный экранный редактор) в эмуляторе терминала, он все еще использует TERM и данные termcap или terminfo, а также обращает внимание на установки stty. Запуск в эмуляторе терминала – такой же простой способ изучить vi, как и любой другой.

Другим важным для понимания vi фактом является то, что он развивался в то время, когда системы UNIX были намного менее стабильными, чем сейчас. Пользователи тех лет должны были быть готовыми к сбою в системе в любой момент, а в vi была предусмотрена поддержка восстановления тех файлов, которые редактировались в момент системного сбоя[1]. Так что если во время изучения vi вы увидите описание различных возникающих проблем, вспомните историю его развития.

Открытие и закрытие файлов

vi можно использовать для редактирования произвольного текстового файла. vi копирует редактируемый файл в *буфер* (временно выделяемую область памяти), отображает буфер (хотя в каждый момент времени вы видите только ту часть, которая поместилась на экране) и позволяет вам добавлять, удалять или менять текст. При сохранении результатов редактирования vi копирует отредактированный буфер обратно в постоянный файл, замещая старый файл с тем же именем. Не забывайте, что вы всегда работаете с *копией* вашего файла, хранимой в буфере, поэтому все ваши правки не изменят первоначальный файл, пока вы не сохраните буфер. Сохранение изменений часто называют «сохранением буфера» или просто «сохранением файла».

[1] К счастью, ситуации такого рода случаются гораздо реже, хотя системы все еще могут аварийно завершить работу из-за внешних причин, например из-за прекращения подачи питания.

Открытие файла

vi – это команда UNIX, которая вызывает редактор vi для существующего или для совершенно нового файла. Синтаксис использования этой команды следующий:

```
$ vi [filename]
```

Скобки, показанные в этой строке, означают, что имя файла – необязательный параметр. Сами скобки набирать не надо. Знак $ – это приглашение командной строки UNIX. Если не указать имя файла, то vi откроет безымянный буфер. Имя можно указать при сохранении буфера в файл. А пока давайте остановимся на указании имени файла в командной строке.

Имя файла должно быть уникальным в пределах одного каталога. Оно может содержать любой из 8-битных символов, кроме знака косой черты (/), зарезервированного в качестве разделителя между файлами и каталогами в пути файла, и ASCII NUL – символа с нулевыми разрядами. В имени файла можно даже использовать пробелы; в этом случае перед пробелом следует поставить обратную косую черту (\). Тем не менее на практике имена файлов в основном содержат различные сочетания больших и маленьких букв, цифр, символов точки (.) и подчеркивания (_). Помните, что UNIX чувствителен к регистру: строчные буквы отличаются от прописных. Также не забывайте нажимать на ENTER, чтобы сообщить UNIX о том, что вы закончили ввод команды.

Если вы хотите создать в каталоге новый файл, задайте в команде vi новое имя файла. Например, чтобы в текущем каталоге открыть новый файл с именем practice, введите:

```
$ vi practice
```

Поскольку это новый файл, буфер будет пустым, и на экране вы увидите следующее:

```
~
~
~
"practice" [New file]
```

Тильды (~) в левом столбце экрана указывают, что в файле нет никакого текста, нет даже пустых строк. Строка приглашения (также называемая строкой состояния) внизу экрана отображает имя и состояние файла.

Если вы укажете имя любого из существующих в каталоге файлов, то сможете отредактировать его. Предположим, что существует файл с абсолютным путем /home/john/letter. Если вы уже находитесь в каталоге /home/john, используйте относительный путь к файлу. Например

```
$ vi letter
```

выдаст на экран файл letter.

Если вы находитесь в другом каталоге, введите полный путь к файлу, чтобы начать его редактирование:

```
$ vi /home/john/letter
```

Проблемы при открытии файлов

- *При запуске* `vi` *появляется сообщение* `[open mode]`

  Возможно, неправильно распознается тип вашего терминала. Немедленно выйдите из сеанса редактирования, введя команду `:q`. Проверьте переменную окружения `$TERM`. Ей нужно присвоить имя вашего терминала. Или можете попросить системного администратора дать вам правильное значение типа терминала.

- *Вы видите одно из следующих сообщений*:

  ```
  Visual needs addressable cursor or upline capability
  Bad termcap entry
  Termcap entry too long
  terminal: Unknown terminal type
  Block device required
  Not a typewriter
  ```

  Либо тип вашего терминала не опознан, либо что-то не так с его записью в вашем `terminfo` или `termcap`. Введите `:q`, чтобы выйти. Проверьте переменную окружения `$TERM` или попросите системного администратора выбрать тип терминала для вашего окружения.

- *Появляется сообщение* `[new file]`, *когда вы считаете, что файл уже существует*.

  Проверьте, правильный ли регистр символов вы использовали в имени файла (имена файлов в UNIX чувствительны к регистру). Если все верно, возможно, вы находитесь в другом каталоге. Введите `:q` для выхода. После этого проверьте, находитесь ли вы в том же каталоге, что и файл (введите `pwd` в командной строке UNIX). Если вы в нужном каталоге, выведите список содержащихся в нем файлов (с помощью `ls`) и проверьте, нет ли файла под немного другим именем.

- *Вы запустили* `vi`, *однако попали в приглашение с двоеточием (что говорит о том, что вы находитесь в режиме строкового редактирования* `ex`*)*.

  Возможно, вы ввели прерывание перед тем, как `vi` успел отрисовать экран. Войдите в `vi`, введя в приглашении `ex` (`:`) команду `vi`.

- *Появляется одно из следующих сообщений*:

  ```
  [Read only]
  File is read only
  Permission denied
  ```

  «Read only» означает, что вы можете только просматривать файл; ваши изменения не могут быть сохранены. Возможно, вы запустили `vi`

в режиме просмотра (либо через `view`, либо как `vi -R`) либо у вас нет прав на запись этого файла. Обратитесь к разделу «Проблемы при сохранении файлов» на стр. 30.

- *Появляется одно из следующих сообщений*:

  ```
  Bad file number
  Block special file
  Character special file
  Directory
  Executable
  Non-ascii file
  file non-ASCII
  ```

  Файл, который вы хотите отредактировать, не является обычным текстовым файлом. Введите `:q!` для выхода и проверьте этот файл, например командой `file`.

- *При вводе* `:q` *по одной из вышеназванных причин появляется сообщение*:

  ```
  No write since last change (:quit! overrides).
  ```

  Вы ненароком внесли изменение в файл. Для выхода из `vi` введите `:q!`. В этом случае изменения, сделанные во время сеанса, не будут сохранены.

Образ действия

Как упоминалось ранее, концепция текущего «режима» является фундаментальной в работе `vi`. Существуют два режима: *режим вставки* и *командный режим*. Сразу после запуска активен командный режим, в котором каждое нажатие клавиши вызывает команду. В режиме вставки все, что вы печатаете, становится содержимым вашего файла.

Иногда случайно можно попасть в режим вставки или, наоборот, ненароком выйти из него. В любом случае то, что вы введете, скорее всего, нежелательно отразится на содержимом файла.

Нажмите ESC, чтобы попасть в командный режим. Если вы уже в нем, `vi` даст звуковой сигнал (beep) при нажатии ESC. (Поэтому командный режим иногда называют сигнальным режимом.)

Благополучно перейдя в командный режим, вы можете исправить любые случайные изменения, после чего вернуться к редактированию вашего текста.

Сохранение файла и выход

В любой момент можно прекратить работу с файлом, сохранить правки и вернуться в приглашение командной строки UNIX. Команда `vi`, которая сохраняет изменения и прекращает работу редактора, называется `ZZ`. Обратите внимание, что `ZZ` пишется прописными буквами.

Предположим, вы создали файл под названием practice и ввели в нем шесть строчек текста. Чтобы сохранить файл, сначала нажатием ESC проверьте, что вы попали в командный режим, после чего введите ZZ.

Клавиши	Результат
ZZ	"practice" [New file] 6 lines, 320 characters Введена команда записи ZZ. Ваш файл сохранится как обычный файл UNIX.
ls	ch01 ch02 practice Вывод списка файлов в каталоге покажет, что вы создали новый файл practice.

Результаты редактирования можно сохранить и с помощью команд ex. Чтобы сохранить (write) файл, не выходя из vi, введите :w. Если вы ничего не меняли в файле, выйти можно с помощью команды :q, а введя :wq, вы сохраните изменения и покинете vi. (:wq эквивалентно ZZ.) В главе 5 мы подробно расскажем об использовании команд ex. Сейчас просто запомните эти несколько команд для записи и сохранения файлов.

Выход без сохранения правок

При первом знакомстве с vi, особенно если вы бесстрашный экспериментатор, вам могут понадобиться две другие команды ex, чтобы избавиться от созданной вами путаницы.

Если вы захотите отменить все сделанные за сеанс изменения и вернуться к первоначальному файлу, то команда

```
:e! ENTER
```

вернет вас к последней сохраненной версии файла, и вы сможете начать все заново.

Если же вы хотите отказаться от изменений и выйти из vi, то команда

```
:q! ENTER
```

осуществит выход из редактируемого файла и возврат в приглашение UNIX. Обе эти команды приведут к потере всех изменений, сделанных в буфере со времени последнего сохранения. Обычно vi не позволяет отказаться от изменений. Восклицательный знак, добавленный к командам :e или :q, заставит vi отменить этот запрет и выполнить операцию, несмотря на то, что буфер был изменен.

Проблемы при сохранении файлов

- *Вы пытаетесь записать файл, но получаете одно из следующих сообщений*:

  ```
  File exists
  File file exists - use w!
  [Existing file]
  File is read only
  ```

  Введите `:w! file`, чтобы перезаписать существующий файл, или `:w newfile`, чтобы сохранить текущую редакцию в новом файле.

- *Вы хотите записать файл, но у вас нет разрешения на запись для него. Вам выдается «Permission denied.»*

  Используйте команду `:w newfile`, чтобы записать содержимое буфера в новый файл. При наличии прав на запись для этого каталога вы сможете с помощью команды `mv` заменить первоначальную версию новым файлом. Если у вас нет разрешения на запись для этого каталога, введите `:w pathname/file`, чтобы записать буфер в том каталоге, где у вас есть разрешение на запись (например, домашний каталог или `/tmp`).

- *Вы пытаетесь записать файл, но получаете сообщение о том, что файловая система переполнена.*

  Введите `:!rm junkfile`, чтобы удалить (большой) ненужный файл, тем самым освободив немного места. (Если команду `ex` начать с восклицательного знака, то вы получите доступ в UNIX.)

  Или введите `:!df`, чтобы посмотреть, есть ли свободное место в другой файловой системе. Если есть, выберите каталог в той системе и запишите файл туда, воспользовавшись командой `:w pathname`. (`df` — это команда UNIX, которая проверяет свободное место на дисках; название происходит от **d**isk **f**ree.)

- *Система переводит вас в открытый режим (open mode) и сообщает, что файловая система переполнена.*

  Диск переполнен временными файлами `vi`. Введите `:!ls /tmp`, чтобы посмотреть, есть ли файлы, которые можно удалить, дабы получить немного места на диске[1]. Если таковые имеются, создайте временную оболочку UNIX, из которой вы сможете удалить эти файлы, или обратитесь к другим командам UNIX. Оболочку можно создать, если ввести `:sh`; для выхода из оболочки и возврата в `vi` нажмите CTRL-D или введите команду `exit`. (В современных системах UNIX при использовании оболочки с управлением заданиями можно просто нажать CTRL-Z, чтобы приостановить `vi` и вернуться в командную

[1] `vi` может хранить временные файлы в `/usr/tmp`, `/var/tmp` или в вашем домашнем каталоге; возможно, вам придется немного покопаться, чтобы разузнать, что именно занимает столько места. Vim обычно держит свои временные файлы в том же каталоге, что и редактируемый файл.

строку UNIX; для возврата в vi введите fg.) После высвобождения места на диске сохраните ваш файл командой :w!.

- *Вы пытаетесь записать файл, но получаете сообщение о том, что достигнуты дисковые квоты.*

  Попробуйте заставить систему записать ваш буфер с помощью команды :pre (сокращение от :preserve). Если это не сработало, поищите файлы для удаления. Воспользуйтесь командой :sh (или CTRL-Z, если система поддерживает управление заданиями), чтобы выйти из vi и удалить файлы. Для возврата в vi нажмите CTRL-D (или введите fg). Затем запишите файл командой :w!.

Упражнения

Единственный способ выучить vi – это практика. Сейчас вы уже знаете достаточно, чтобы создать новый файл и вернуться в приглашение UNIX. Создайте файл с именем practice, внесите в него немного текста, затем сохраните файл и выйдите.

Открыть файл с именем practice в текущем каталоге:	vi practice
Вставьте текст:	i *любой текст*
Возврат в командный режим:	ESC
Выход из vi с сохранением правок:	ZZ

2

Простое редактирование

Эта глава, построенная в виде руководства, познакомит вас с редактированием в vi. Вы узнаете, как перемещать курсор и делать простые правки. Если вы еще ни разу не работали в vi, то лучше прочтите эту главу целиком.

Последующие главы призваны углубить ваши навыки, что позволит вам работать быстрее и эффективнее. Одно из главных *преимуществ* vi для опытного пользователя – огромный выбор опций (при этом один из главных *недостатков* для новичка – огромное число команд редактора).

Нельзя освоить vi, просто запомнив все команды. Начните с изучения простых команд, о которых рассказывается в этой главе, и обращайте внимание на общие шаблоны их использования.

Изучая vi, берите на заметку задачи, которые вы можете поручить редактору, а затем найдите команды, решающие их. В последующих главах вы узнаете о продвинутых свойствах vi, но прежде чем браться за сложное, нужно освоить азы.

В этой главе рассказывается о том, как:

- Перемещать курсор
- Добавлять и менять текст
- Удалять, перемещать и копировать текст
- Переходить в режим вставки разными способами

Команды vi

В vi есть два режима: командный и режим вставки. При входе в файл вы оказываетесь в командном режиме, и редактор ждет ввода команд. Они позволяют перемещаться на любое место в файле, производить правки или переходить в режим вставки, чтобы добавить новый текст.

Команды также нужны для выхода из файла (с сохранением изменений или без), чтобы вернуться в командную строку UNIX.

Оба режима работы можно рассматривать как две разные «клавиатуры». В режиме вставки ваша клавиатура работает подобно печатной машинке. В командном режиме каждая клавиша имеет свое значение или вызывает какую-либо инструкцию.

Есть несколько способов сообщить vi о переходе в режим вставки. Самый простой из них – нажать клавишу i. Сама буква i на экране не появится, но после ее нажатия все, что вы набираете, *возникнет* на экране и будет передаваться в буфер. При этом курсор отмечает место для вставки нового текста[1]. Для выхода из режима вставки нажмите ESC. Это действие переместит курсор на один символ назад (он встанет на последний введенный вами символ) и вернет vi в командный режим.

Например, вы открыли новый файл и хотите вставить туда слово «introduction». Если ввести iintroduction, то на экране появится:

```
introduction
```

При открытии нового файла vi начинает работу в командном режиме и понимает первую клавишу (i) как команду вставки. После этого все введенные символы рассматриваются им как текст, пока вы не нажмете ESC. Чтобы исправить ошибку в режиме вставки, вернитесь назад с помощью клавиши BACKSPACE и наберите символ заново. В зависимости от типа используемого вами терминала BACKSPACE может либо удалять набранный текст с экрана, либо перемещать курсор поверх него. В любом случае замещаемый текст будет удален. Обратите внимание, что вы не сможете использовать клавишу BACKSPACE дальше того места, где включили режим вставки. (Если в Vim выключить совместимость с vi, то там возможно перемещение курсора дальше места начала режима вставки.)

В vi есть опция, позволяющая определить правый отступ и выполняющая возврат каретки всякий раз при его достижении. Пока же во время вставки текста для перехода на новую строку используйте ENTER.

Иногда сложно определить, в каком из двух режимов вы находитесь. Если vi ведет себя не так, как ожидалось, нажмите ESC один или два раза, чтобы проверить, в каком вы режиме. Звуковой сигнал означает, что в командном.

Перемещение курсора

Скорее всего, во время сеансов редактирования вы будете уделять мало внимания вставке нового текста, так как большая часть времени уйдет на правку существующего.

---

[1] В некоторых версиях строка состояния показывает, что вы находитесь в режиме вставки.

В командном режиме можно перемещать курсор в любую часть файла. Поскольку основное редактирование (изменение, удаление и копирование текста) начинается с перемещения курсора в тот фрагмент, который вы хотите изменить, вам, вероятно, захочется переместить его туда как можно быстрее.

В `vi` есть команды, которые перемещают курсор:

- Вверх, вниз, влево или вправо на один *символ*.
- Вперед или назад на *текстовые* блоки, такие как слова, предложения или абзацы.
- Вперед или назад по файлу на один *экран*.

На рис. 2.1 знак подчеркивания указывает на текущее положение курсора. Кружки показывают, какие его перемещения относительно текущей позиции вызовут те или иные команды `vi`.

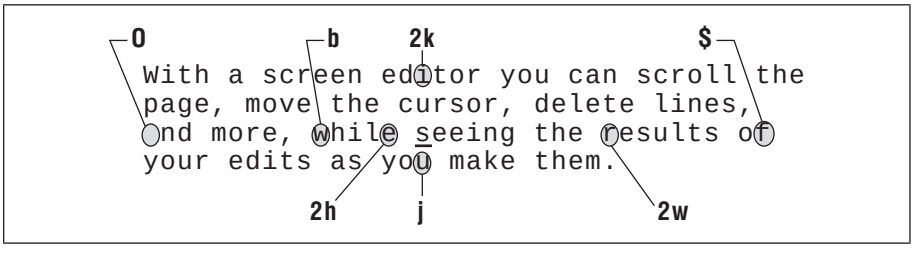

Рис. 2.1. *Примеры команд перемещения курсора*

Простые движения

Клавиши `h`, `j`, `k` и `l`, удобно расположенные под кончиками ваших пальцев, перемещают курсор:

`h`
: На один символ влево.

`j`
: На одну строку вниз.

`k`
: На одну строку вверх.

`l`
: На один символ вправо.

Также для перемещения вверх и вниз можно использовать курсорные клавиши (←, ↓, ↑, →), + и – либо ENTER и BACKSPACE, но они расположены в стороне. Поначалу может показаться, что пользоваться буквенными кнопками вместо курсорных неудобно, но потом вы поймете, что это одно из лучших качеств `vi`, позволяющее перемещаться по тексту, не отрывая пальцев от центра клавиатуры.

Перед перемещением курсора нажмите ESC, чтобы убедиться, что вы находитесь в командном режиме. Используйте h, j, k и l, чтобы двигаться вперед или назад относительно текущего положения курсора. Достигнув предела в каком-либо направлении, вы услышите звуковой сигнал, и курсор остановится. Например, находясь в начале или конце строки, нельзя воспользоваться h или l для возврата в конец (или начало) предыдущей (или следующей) строки, поэтому следует нажать j или k[1]. Аналогично нельзя ни подвинуть курсор дальше знака тильды (~), который обозначает строку без текста, ни переместить курсор выше первой строки в файле.

Числовые аргументы

Перед командами перемещения можно указывать число. Рисунок 2.2 показывает, как команда 4l переместит курсор на четыре символа вправо, как если бы вы нажали четыре раза на l (llll).

Способность дублировать команды дает больше возможностей и контроля над каждой изученной командой. Не забывайте об этом, когда будете знакомиться с другими командами.

```
        4l
   With a screen editor you can scroll the
```

Рис. 2.2. *Дублирование команд с помощью коэффициентов*

Перемещение в строке

Ранее при сохранении файла practice vi выдал сообщение о количестве строк в этом файле. *Строка* не обязательно имеет одинаковую длину со строкой на экране (часто ограниченной 80 символами). Строка – это любой текст между двумя переводами строк. (Символ *перевода строки* вставляется в файл всякий раз при нажатии ENTER в режиме вставки.) Если перед нажатием на ENTER вы введете 200 символов, vi воспримет все эти 200 символов как одну строку (несмотря на то, что визуально они займут несколько строк на экране).

Как мы уже упоминали в главе 1, vi имеет опцию, позволяющую установить расстояние от правого поля, при котором автоматически вставляется перевод строки. Эта опция называется wrapmargin (сокращается до wm). Можно установить wrapmargin на 10 символах:

```
:set wm=10
```

[1] Vim с установленной опцией nocompatible позволяет переместиться с конца строки на начало следующей с помощью клавиш l или пробела.

Данная команда не повлияет на уже набранные строки. В главе 7 мы поговорим о задании опций подробнее, но конкретно эту нельзя было отложить на потом!

Если вы не используете автоматический перенос `wrapmargin`, то следует завершать строки возвратом каретки, чтобы они имели приемлемый размер.

Для перемещения по строке есть две очень полезные команды:

`0` (цифра «ноль»)

Перемещает в начало строки.

`$`

Перемещает в конец строки.

В следующем примере отображаются номера строк. (В `vi` их можно вывести, если воспользоваться опцией `number`, которая включается при вводе `:set nu` в командном режиме. Эта операция описана в главе 7.)

```
1 With a screen editor you can scroll the page,
2 move the cursor, delete lines, insert characters,
  and more, while seeing the results of your edits
  as you make them.
3 Screen editors are very popular.
```

Количество логических строк (3) не соответствует тому, что вы видите на экране (5). Если ввести `$`, когда курсор расположен на букве *d* слова *delete*, то он перейдет на точку после слова *them*. При вводе `0` курсор переместится назад к букве *m* в слове *move* в самом начале строки 2.

Перемещение по текстовым блокам

Курсор можно перемещать и по текстовым блокам, таким как слова, предложения, абзацы и т. д. Команда `w` перемещает курсор вперед на одно слово, при этом символы и знаки препинания тоже считаются словами. Следующая строка демонстрирует перемещение курсора с помощью `w`:

```
cursor, delete lines, insert characters,
```

Командой `W` можно перемещаться без учета символов и знаков препинания (можете рассматривать это как «обобщенное» или «большое» слово – Word).

Передвижение курсора при помощи `W` выглядит так:

```
cursor, delete lines, insert characters,
```

Для движения по словам в обратном направлении используется `b`. Прописная `B` позволит делать то же самое, но без учета пунктуации.

Как упоминалось ранее, команды перемещения могут снабжаться числовыми аргументами, то есть как `b`, так и `w` могут умножаться на числа. `2w` передвинет курсор вперед на два слова, а `5B` – назад на пять слов без учета знаков препинания.

Для перехода на определенную строку используйте команду `G`. Просто `G` переместит вас в конец файла, `1G` – в его начало, а `42G` – на строку под номером 42. Подробнее об этом читайте в разделе «Команда G (Go To)» на стр. 64.

В главе 3 мы рассмотрим перемещение по предложениям и абзацам, а пока попрактикуйтесь в перемещении курсора уже известными вам командами в сочетании с числовыми указателями.

Простая правка текста

При наборе текста в файле он редко получается идеальным. Можно обнаружить опечатки или захотеть улучшить отдельные фразы. Также, если вы пишете программу, то она может содержать ошибки. После ввода текста полезно иметь возможность изменить, удалить, переместить или скопировать его. На рис. 2.3 показаны разновидности правок, которым может подвергнуться файл. На исправления указывают отметки корректора.

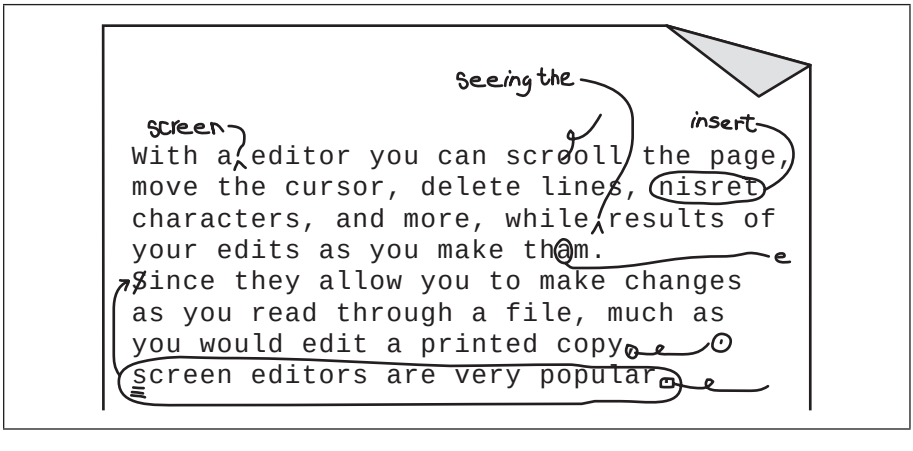

Рис. 2.3. *Правки с отметками корректора*

В `vi` можно выполнять любые из этих изменений с помощью всего нескольких клавиш: `i` – для вставки (вы это уже знаете), `a` (от *append*) – для добавления, `c` – для изменения (от *change*) и `d` – для удаления (*delete*). Чтобы переместить или скопировать текст, используйте последовательности команд. Например, текст перемещается путем его «удаления» командой `d` и последующей «вставки» командой `p` (*put*); при копировании текста следует сначала нажать `y` для «копирования» (*yank*), затем `p` – для

«вставки». В данной главе описаны все команды редактирования. На рис. 2.4 показаны команды vi, используемые в правках на рис. 2.3.

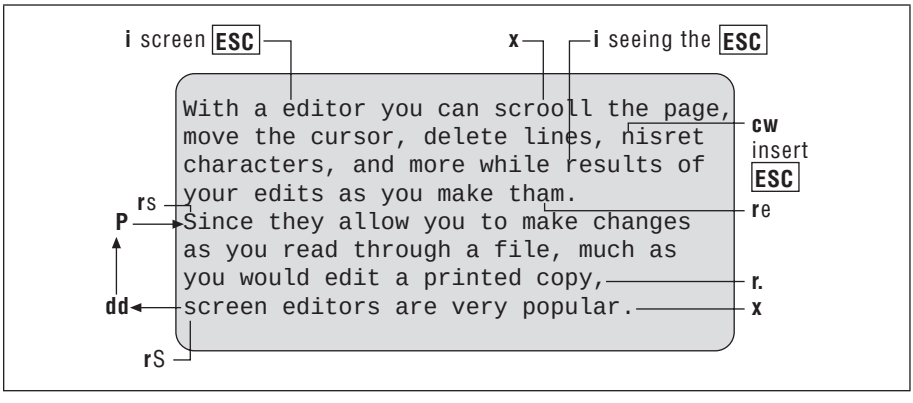

Рис. 2.4. Редактирование командами vi

Ввод нового текста

Вы уже знакомы с командой вставки, которая применяется для ввода текста в новом файле. Эта команда также используется при редактировании существующего текста и добавлении недостающих символов, слов и предложений. Предположим, в файле practice есть следующее предложение:

```
you can scroll
the page, move the cursor, delete
lines, and insert characters.
```

Здесь отмечено положение курсора. Чтобы вставить в начало предложения *With a screen editor*, введите следующее:

Клавиши	Результат

2k	`you can scroll`
	`the page, move the cursor, delete`
	`lines, and insert characters.`
	
	Командой k переместите курсор вверх на две строки, где нужно вставить новые слова.

iWith a	`With a you can scroll`
	`the page, move the cursor, delete`
	`lines, and insert characters.`
	
	Введите i для перехода в режим вставки и начните набирать текст.

Клавиши	Результат

screen editor
ESC
	`With a screen editor you can scroll`
	`the page, move the cursor, delete`
	`lines, and insert characters.`
	
	После ввода текста нажмите ESC, чтобы выйти из режима вставки в командный режим.

Добавление текста

С помощью команды добавления `a` (*append*) вы можете добавлять текст в любом месте файла. Эта команда работает аналогично `i`, но позволяет вставлять текст *после* курсора, а не *перед* ним. Возможно, вы заметили, что при нажатии на `i` для перехода в режим вставки курсор не двигается, пока вы не введете что-нибудь. Напротив, после нажатия на `a` и перехода в режим вставки курсор переместится на одну позицию вправо. Теперь при вводе текст появится после изначального положения курсора.

Изменение текста

Текст в файле можно заменять при помощи соответствующей команды `c` (*change*). Чтобы сообщить ей, сколько именно текста нужно поменять, сочетайте ее с командой перемещения. В этом случае команда перемещения служит как текстовый объект, на который влияет команда `c`. Например, `c` можно использовать для изменения текста, расположенного:

`cw`

До конца слова.

`c2b`

Назад на два слова.

`c$`

До конца строки.

`c0`

До начала строки.

После вызова команды изменения указанный отрезок текста можно заменить любым количеством нового, пустой строкой, одним словом или сотней строк. Подобно `i` и `a`, команда `c` оставляет пользователя в режиме вставки до нажатия ESC.

Если замена затрагивает только текущую строку, `vi` помечает конец изменяемого текста знаком `$`, и тогда видно, какая часть строки изменяется (см. пример к `cw` чуть ниже).

Слова

Чтобы изменить слово, используйте сочетание команд `c` (*change*) и `w`. Вы можете заменить слово (`cw`) на более длинное или короткое (или на любое количество текста). `cw` можно рассматривать как «удалить помеченное слово и добавлять новый текст, пока не будет нажата ESC».

Представим, что у нас есть следующая строка в файле `practice`:

```
With an editor you can scroll the page,
```

и мы хотим поменять *an* на *screen*. Нужно изменить всего одно слово:

Клавиши	Результат
`w`	`With an editor you can scroll the page,` С помощью `w` переместитесь на то место, где хотите начать изменение.
`cw`	`With a$ editor you can scroll the page,` Задайте команду замены слова. Конец изменяемого текста отмечается символом `$` (знак доллара).
`a screen`	`With a screen editor you can scroll the page,` Введите текст замены и нажмите ESC для возврата в командный режим.

`cw` также работает с фрагментом слова. Например, чтобы исправить *spelling* на *spelled*, подведите курсор к *i*, введите `cw`, затем – *ed* и завершите ввод, нажав на ESC.

Строки

Чтобы полностью заменить текущую строку, используйте команду изменения `cc`. Она меняет всю строку независимо от положения курсора в ней, подставляя вместо нее любое количество текста, введенного перед нажатием на ESC.

Работа команды типа `cw` отличается от команды типа `cc`. При вызове `cw` старый текст остается на месте, пока вы не начнете печатать поверх него. После этого старый текст стирается при нажатии ESC. А если воспользоваться командой `cc`, то старый текст удаляется сразу, оставляя пустое место для вставки нового.

Подход «печать поверх» применяется для всех команд изменения, которые затрагивают часть строки, тогда как подход «пустая строка» используется, если команда изменения влияет на целую строку или больше.