Оглавление

ЧАСТЬ II. VIM

ЧАСТЬ III. VIM В БОЛЕЕ ШИРОКОЙ СРЕДЕ

ПРИЛОЖЕНИЯ

ГЛАВА 1

Знакомство с vi и Vim

Одна из наиболее важных повседневных задач на компьютере — работа с текстом: составление нового текста, редактирование и компоновка имеющегося текста, удаление или изменение неправильного и устаревшего текста. Все это выполняется в специальных текстовых процессорах, например в Microsoft Word. Будучи программистом, вы тоже работаете с текстом: с файлами исходного кода программ и вспомогательными файлами, необходимыми для разработки. Текстовые редакторы обрабатывают содержимое любых подобных файлов, независимо от того, содержат ли они данные, исходный код или естественный текст.

Книга посвящена обработке текста с помощью двух связанных редакторов: `vi` и Vim. Первый традиционно используется в качестве стандартного текстового редактора Unix[1]. Второй же основан на консольном режиме `vi` и командном языке, благодаря чему обладает бóльшим количеством полезных возможностей, чем исходная программа.

Текстовые редакторы и редактирование текста

Текстовые редакторы

С течением времени текстовые редакторы Unix улучшались и эволюционировали. Первоначально существовали *линейные редакторы*, такие как `ed` и `ex`, предназначенные для использования на последовательных терминалах, которые печатали на бумаге с непрерывной подачей (да, люди действительно программировали такие

[1] Сегодня термин Unix включает в себя как коммерческие системы, производные от оригинальной кодовой базы Unix (например, Solaris, AIX и HP-UX), так и системы Unix с открытым исходным кодом (например, GNU/Linux и производные от BSD системы). Под это понятие подпадает терминальная среда macOS, подсистема Windows для Linux (WSL) в MS-Windows, а также Cygwin и другие аналогичные среды для Windows. Если не указано иное, то описанное в книге применимо ко всем этим системам.

вещи, включая одного из авторов данной книги!). Линейные редакторы были так названы, потому что работали над программой по одной или нескольким строкам за раз.

С появлением электронно-лучевых терминалов (ЭЛТ) с адресацией курсора линейные редакторы превратились в *экранные редакторы*, такие как `vi` и Emacs. Экранные редакторы позволяют работать с несколькими файлами одновременно и легко перемещаться по строкам текста на экране.

С появлением сред графического пользовательского интерфейса (GUI) экранные редакторы превратились в графические текстовые редакторы, в которых используется мышь для прокрутки видимой части файла, перехода к определенной точке и выделения текста. Примерами таких текстовых редакторов, основанных на X Window System, являются `gedit` в системах на базе Gnome и Notepad++ в Windows. Но существуют и другие.

Особый интерес для нас представляет то, что популярные экранные редакторы эволюционировали в графические: GNU Emacs предоставляет несколько окон X, как и Vim через свою версию `gvim`. Графические редакторы работают идентично своим оригинальным экранным версиям, что сглаживает переход к программам с GUI.

Из всех *стандартных* редакторов в системе Unix `vi` наиболее полезен[1]. По сравнению с Emacs он доступен в практически идентичной форме на всех современных Unix-системах, обеспечивая, таким образом, своего рода общее средство редактирования текста[2]. То же самое можно сказать и о редакторах `ed` и `ex`, но GUI-редакторы намного удобнее в использовании (настолько, что линейные редакторы больше не используются).

vi — сокращение от visual editor и произносится как «ви». Это наглядно показано на рис. 1.1.

Для большинства новичков интерфейс `vi` выглядит громоздким и непонятным: вместо того чтобы позволить вам просто печатать текст, здесь почти все клавиши клавиатуры используются для выполнения каких-либо команд. Чтобы набирать текст на экране, вы должны перейти из *командного режима*, в котором каждая клавиша отвечает за определенное действие, в специальный *режим ввода*. Кроме того, на первый взгляд кажется, что команд в этой программе огромное множество.

---

[1] Если у вас не установлен ни vi, ни Vim, см. приложение Г.

[2] GNU Emacs стала универсальной версией Emacs. Единственная проблема заключается в том, что она не входит в стандартную комплектацию большинства систем, ее нужно установить самостоятельно, в том числе в некоторых системах GNU/Linux.

Рис. 1.1. Как правильно произносить vi

Однако, как только вы начнете осваивать редактор, вы поймете, что он неплохо продуман. Вам понадобится лишь несколько нажатий клавиш, чтобы программа начала решать сложные задачи. По мере изучения `vi` вы узнаете сочетания клавиш, которые позволят компьютеру брать на себя большую часть работы.

`vi` и Vim (как и любые другие текстовые редакторы) не являются текстовыми процессорами класса «что видите, то и получаете». Если вы хотите создавать форматированные документы, вы должны ввести специальные инструкции (иногда называемые *кодами форматирования*), которые считываются отдельной программой форматирования для управления внешним видом выводимого документа. Например, если вы захотите сделать отступ в тексте абзаца, то потребуется ввести код с указанием, где начинается и заканчивается отступ. Коды форматирования позволяют экспериментировать с образом выводимого файла, и во многих отношениях они дают вам гораздо больше контроля над оформлением ваших документов, чем обычный текстовый процессор.

Коды форматирования — это специфические методы в так называемых *языках разметки*[1]. В последние годы популярность языков разметки резко возросла, а особую известность приобрели Markdown и AsciiDoc[2]. Возможно, наиболее популярным языком разметки сегодня является HTML, используемый при создании веб-страниц Интернета.

---

[1] От выделений красным маркером до «пометки» изменений в наборных гранках или пробах.

[2] Дополнительную информацию об этих языках можно найти на сайтах https://en.wikipedia.org/wiki/Markdown и https://asciidoc.org соответственно. Эта книга написана с помощью языка AsciiDoc.

Помимо вышеупомянутых языков разметки, Unix поддерживает пакет форматирования `troff`[1]. Популярны и широко распространены программы форматирования TeX (www.ctan.org) и LaTeX (www.latex-project.org). Чтобы использовать любой из этих языков, достаточно открыть текстовый редактор.

Редактор vi поддерживает некоторые простые механизмы форматирования. Например, вы можете дать ему команду автоматически переносить слова при переходе к концу строки или автоматически делать отступы в новых строках. Кроме того, Vim способен проверять орфографию.

Как и любой навык, редактирование в `vi` требует практики, и чем больше, тем лучше. Привыкнув ко всем возможностям программы, вы, возможно, не захотите возвращаться к какому-либо «более простому» редактору.

Редактирование текста

Что такое редактирование? Во-первых, *ввод* (если, например, забыли какое-то слово или пропустили целое предложение) и *удаление* текста (случайный символ или целый абзац). Также *изменение* букв и слов (исправление орфографических ошибок, опечаток). Возможно, вам понадобится *переместить* текст из одной части документа в другую. И иногда может потребоваться *скопировать* текст, чтобы продублировать его.

В отличие от многих текстовых процессоров в `vi` командный режим является режимом по умолчанию. Сложные интерактивные изменения могут быть выполнены всего несколькими нажатиями клавиш. Чтобы ввести текст, сначала необходимо задать любую команду ввода.

Для основных команд используются один или два символа. Например:

- `i` — ввод;
- `cw` — изменить слово.

Используя буквы в качестве команд, вы можете очень быстро редактировать файл. Не нужно запоминать сотни горячих клавиш или тянуться пальцами до клавиш

[1] troff используется в ПО для лазерных принтеров и наборных устройств. Его брат-близнец nroff предназначен для линейных принтеров и терминалов. Оба понимают один и тот же язык ввода. Следуя общепринятому соглашению Unix, под названием troff мы ссылаемся на оба пакета. В настоящее время все, кто использует troff, работают в GNU-версии под названием groff (https://www.gnu.org/software/groff).

в неудобных комбинациях. Вам не придется убирать руки с клавиатуры или возиться с многоуровневыми меню! Большинство команд можно запомнить по первым буквам, отражающим их названия на английском языке. Почти все команды следуют этому принципу и связаны друг с другом.

В целом команды `vi` и Vim:

- чувствительны к регистру (клавиши в верхнем и нижнем регистре означают разные действия; `I` отличается от `i`);
- не отображаются (не выводятся) на экране при вводе;
- не требуют нажатия Enter после ввода команды.

Существует также группа команд, которые отображаются в нижней строке экрана. Им предшествуют разные символы. Слеш (`/`) и знак вопроса (`?`) отвечают за команды поиска и обсуждаются в главе 3. Двоеточие (`:`) предваряет все команды `ex`. Команды `ex` — это те, которые используются линейным редактором `ex`. Редактор `ex` доступен вместе с любой версией `vi`, потому что `ex` — это базовый редактор, а `vi` — просто его «визуальный» режим. Бóльшая часть команд и концепций `ex` обсуждается в главе 5, а в этой мы познакомимся с командами `ex` для закрытия файла без сохранения изменений.

Небольшая историческая справка

Прежде чем погрузиться во все тонкости `vi` и Vim, важно понять роль `vi` в вашей рабочей среде. В частности, мы разберемся во множестве обескураживающих сообщений об ошибках `vi`, а также выясним, как Vim эволюционировал по сравнению с оригинальным `vi`.

Редактор `vi` появился, когда в ходу были ЭЛТ-терминалы, подключенные через последовательные интерфейсы к центральным миникомпьютерам. Сотни различных моделей терминалов разрабатывались и использовались по всему миру. Каждый из них выполнял одни и те же действия (очищал экран, перемещал курсор и т. д.), но команды, необходимые для реализации этих действий, были разными. Кроме того, система Unix позволяла выбирать символы, которые будут использоваться для возврата, генерации сигнала прерывания и других задач, таких как приостановка и возобновление вывода, выполняемых в последовательных терминалах. Эти действия управлялись (и до сих пор управляются) с помощью команды `stty`.

В оригинальной версии `vi` для Berkeley Unix информация об управлении терминалом была отделена от кода (который сложно изменять) и помещена в текстовую

базу данных свойств терминала (легко поддающуюся изменениям), управляемую библиотекой `termcap`. В начале 1980-х годов System V представила двоичную базу данных информации терминала и библиотеку `terminfo`. Эти две библиотеки в целом были эквивалентны по своему функционалу. Чтобы определить, какой у вас терминал, требовалось установить переменную окружения `TERM`. Обычно это выполнялось в файле запуска оболочки `.profile` или `.login`.

Библиотека `termcap` больше не используется. Системы GNU/Linux и BSD работают с библиотекой `ncurses`, которая предоставляет совместимое надмножество базы данных и возможностей библиотеки System V `terminfo`.

Сегодня популярны эмуляторы терминалов в графической среде (например, Gnome Terminal). Система почти всегда самостоятельно настраивает значение переменной `TERM`.

Конечно, вы также можете использовать в Windows консольную программу Vim без GUI. Это пригодится, к примеру, при восстановлении системы в однопользовательском режиме. Однако сейчас мало кто пользуется данным приложением на постоянной основе.

Для повседневного использования лучше всего подойдет версия `vi` с графическим интерфейсом, например `gvim`. В системе Microsoft Windows и macOS она, вероятно, будет приложением по умолчанию. Однако при запуске `vi` (или какого-либо другого экранного редактора того же типа) внутри эмулятора терминала редактор по-прежнему будет обращаться к значению переменной `TERM`, библиотеке `terminfo` и к параметрам команды `stty`. Использование редактора `vi` внутри эмулятора терминала — простой способ изучить `vi` и Vim.

Следует также знать, что программа `vi` была разработана во времена, когда системы Unix были значительно менее стабильными, чем сегодня. Пользователь `vi` тех лет должен был быть готов к сбою системы в любой момент, и поэтому в `vi` реализована функция восстановления файлов, которые редактировались в момент выхода системы из строя[1]. Итак, при изучении программ `vi` и Vim и при чтении описаний различных вероятных проблем учитывайте эти исторические события.

[1] К счастью, сейчас такого рода проблемы встречаются гораздо реже, хотя системы все еще могут выходить из строя из-за внешних обстоятельств, таких как отключение электроэнергии. Чтобы решить и эту проблему, обзаведитесь источником бесперебойного питания для настольной системы или емким аккумулятором на ноутбуке.

Открытие и закрытие файлов

Редактор `vi` можно использовать для редактирования любых текстовых файлов. Программа копирует файл в *буфер* (область памяти, выделяемая временно), отображает содержимое буфера (за раз вы видите только то, что помещается на вашем экране) и позволяет добавлять, удалять и изменять текст. Когда вы сохраняете внесенные изменения, программа копирует отредактированное содержимое буфера обратно в существующий файл, заменяя его под тем же именем. Помните, что вы всегда работаете с *копией* вашего файла в буфере и что вносимые изменения не влияют на исходный файл, пока вы не примените их, перезаписав содержимым из буфера. Сохранение изменений также называется записью буфера или чаще всего записью файла.

Открытие файла из командной строки

`vim` — это команда Unix, которая вызывает редактор Vim для обработки имеющегося или совершенно нового файла. Синтаксис команды `vim` выглядит так:

```
$ vim [имя_файла]
```

или

```
$ vi [имя_файла]
```

Чаще всего в современных системах команда `vi` — это просто ссылка на Vim. Квадратные скобки в приведенных выше командах указывают, что имя файла является необязательным. Скобки вводить не нужно. Знак `$` — это приглашение командной строки.

Если имя файла опущено, редактор открывает безымянный буфер. Вы можете назначить имя при записи содержимого буфера в файл. На данный момент давайте придерживаться именования файла в командной строке.

Имя файла внутри каталога должно быть уникальным (в некоторых операционных системах каталоги называют *папками*; по сути, это одно и то же).

В системах Unix имя файла может состоять из любых восьмибитных символов, за исключением слеша (`/`), который считается разделителем между файлами и каталогами в пути, и ASCII NUL — символа со всеми нулевыми битами. Чтобы использовать пробелы в имени файла, потребуется ввести обратный слеш (`\`) перед самим пробелом (в системах Windows запрещено использовать в именах файлов обратный слеш (`\`) и двоеточие (`:`)). Однако на практике имена файлов обычно состоят из любой комбинации прописных и строчных букв, цифр и символов точки (`.`) и подчеркивания (`_`). Помните, что Unix чувствителен к регистру: строчные

буквы отличаются от прописных. Также имейте в виду, что вы должны нажать клавишу Enter, чтобы сообщить командной оболочке о завершении выполнения команды.

Если вы хотите создать и тут же открыть новый файл, придумайте ему имя и введите его с помощью команды `vi`. Например, если требуется создать и открыть новый файл с именем `practice` в текущем каталоге, наберите:

```
$ vi practice
```

Поскольку это новый файл, буфер пуст, и экран выглядит следующим образом:

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

Тильды (~) в левом столбце экрана указывают, что в файле нет текста, даже пустых строк. Строка запроса (также называемая строкой состояния) в нижней части экрана отображает имя и статус файла.

Вы также можете отредактировать любой существующий текстовый файл в каталоге, указав его имя. Предположим, что существует Unix-файл по адресу /home/john/letter. Если вы уже находитесь в каталоге /home/john, используйте относительный путь. Например, команда ниже выводит на экран копию файла `letter`:

```
$ vi letter
```

Если вы находитесь в другом каталоге, укажите полный путь, чтобы начать редактировать файл:

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

Открытие файла из графического интерфейса

Хотя мы (настоятельно) рекомендуем научиться работать в командной строке, вы можете запустить Vim и открыть файл непосредственно из графического интерфейса. Как правило, необходимо щелкнуть правой кнопкой мыши на файле, а затем выбрать команду типа Открыть с помощью в контекстном меню. Если редактор Vim установлен правильно, это будет одним из доступных вариантов открытия файла.

Чтобы запустить Vim непосредственно из меню, необходимо с помощью ex-команды `:e` *`имя_файла`* указать редактору, какой файл редактировать.

Конкретную команду сложно привести, так как в настоящее время существует и используется множество различных графических сред.

Проблемы при открытии файлов

- *Появляется одно из следующих сообщений:*

```
Visual needs addressable cursor or upline capability
терминал: Unknown terminal type
Block device required
Not a typewriter
```

  Это значит, что ваш тип терминала не определен или что-то не так с записями `terminfo`. Введите `:q` для выхода. Чтобы исправить проблему и начать работу в упрощенном режиме, чаще всего достаточно присвоить параметру `$TERM` значение `vt100`. Дополнительная информация есть в Интернете или на популярном форуме по техническим вопросам, таком как StackOverflow (https://stackoverflow.com).

- *Вы предполагаете, что файл существует, но появляется сообщение [new file].*

  Убедитесь, что вы использовали правильный регистр букв в имени файла (системы Unix чувствительны к регистру в именах файлов). Если да, то вы, вероятно, находитесь не в том каталоге. Введите команду `:q` для выхода. Затем убедитесь, что вы открыли корректный каталог для этого файла (введите `pwd` в командной строке). Если путь верен, проверьте список файлов в каталоге (с помощью команды `ls`). Вероятно, файл записан под немного другим именем.

- *Вы запускаете vi, но получаете приглашение с двоеточием (указывающее, что вы находитесь в режиме строкового редактирования ex).*

  Вероятно, вы ввели команду прерывания (обычно это сочетание клавиш `Ctrl+C`), прежде чем программа `vi` отрисовала экран. Запустите `vi`, введя в приглашении `ex` (`:`) команду `vi`.

- *Появляется одно из следующих сообщений:*

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

  `Read only` (только для чтения) означает, что вы не можете сохранять внесенные изменения, только просматривать файл. Возможно, вы запустили `vi` в режиме просмотра (с помощью команды `view` или `vi -R`) или у вас нет разрешения на перезапись файла (см. подраздел «Открытие файла из командной строки» ранее в этой главе).

- *Появляется одно из следующих сообщений:*

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

Файл, который вы хотите отредактировать, не является допустимым текстовым файлом. Введите `:q!`, чтобы завершить работу, а затем проверьте этот файл с помощью команды `file`.

- *При попытке ввести команду :q из-за одной из ранее упомянутых проблем появляется сообщение:*

  ```
  E37: No write since last change (add ! to override)
  ```

  Вы случайно изменили файл. Введите команду `:q!`, чтобы закрыть редактор. Ваши изменения, внесенные в ходе этого сеанса, не будут сохранены.

Modus Operandi

Как упоминалось ранее, концепция текущего режима имеет основополагающее значение для способа работы `vi`. Существует два режима: *командный режим* и *режим ввода текста* (режим команды `ex` можно считать третьим по счету, но пока мы не будем его рассматривать). При запуске активизируется командный режим, где каждое нажатие клавиши представляет собой команду[1]. В режиме ввода все, что вы набираете, преобразовывается в текст.

Иногда можно случайно перейти в режим ввода или, наоборот, выйти из него. Так или иначе любые нажатия на клавиши, скорее всего, повлияют на содержимое ваших файлов непредвиденным образом.

Нажмите клавишу `Esc`, чтобы перевести редактор в командный режим. Если вы уже в нем, программа подаст звуковой сигнал (поэтому командный режим иногда называют сигнальным режимом).

Перейдя в командный режим, вы сможете приступить к исправлению любых случайных изменений, а затем продолжить редактирование текста (см. подразделы «Проблемы при удалении» и «Откат изменений» в главе 2).

Сохранение и закрытие файла

Чтобы в любой момент прекратить работу с файлом, сохранить внесенные изменения и вернуться в командную строку (если вы работаете в окне терминала), используйте команду `ZZ`. Обратите внимание, что `ZZ` пишется прописными буквами.

Предположим, что вы создали файл с именем `practice`, чтобы попрактиковаться с командами `vi`, и ввели шесть строк текста. Чтобы сохранить файл, сначала

[1] Обратите внимание, что программы vi и Vim имеют команды не для всех возможных клавиш. В командном режиме редактор ожидает нажатий кнопок, представляющих команды, а не текст в качестве содержимого файла. Мы воспользуемся преимуществами неиспользуемых клавиш позже, в разделе «Команда map» главы 7.

убедитесь, что вы находитесь в командном режиме, нажав клавишу Esc, а затем наберите ZZ.

Команда	Результат
ZZ	"practice" [New] 6L, 104C written При вводе команды записи ZZ ваш файл сохранится на диске как обычный файл
$ ls	ch01.asciidoc ch02.asciidoc ch03.asciidoc В списке файлов в каталоге будет файл с именем practice, который вы только что создали

Внесенные изменения можно сохранить и с помощью команд ex. Введите :w, чтобы сохранить (записать) ваш файл, не закрывая его. Если вы не вносили никаких изменений, просто введите :q, чтобы выйти. А чтобы сохранить ваши изменения и закрыть текущий файл, введите :wq (команда :wq эквивалентна ZZ). Мы подробно объясним, как использовать команды ex, в главе 5. Сейчас просто запомните эти несколько команд для записи и сохранения файлов.

Закрытие файла без сохранения изменений

Если вы только начали изучать Vim, особенно будучи бесстрашным экспериментатором, вам пригодятся две другие команды ex, которые позволят избавиться от созданной вами путаницы.

Если вы хотите стереть все изменения, внесенные вами во время сеанса работы, а затем снова открыть исходный файл, то команда:

:e! Enter

откроет последнюю сохраненную версию файла, тем самым позволив вам начать заново.

Если вы не хотите сохранять свои изменения, а затем планируете выйти из редактора, то используйте следующую команду, которая немедленно завершит работу с редактируемым файлом и вернет вас в командную строку:

:q! Enter

Данные команды откатывают назад все изменения, внесенные в буфер с момента последнего сохранения файла. Редактор обычно не позволяет удалять изменения. Восклицательный знак, добавленный к команде :e или :q, снимает этот запрет, и программа выполняет операцию, даже если содержимое буфера было изменено.

Далее мы не будем напоминать о необходимости нажимать клавишу Enter для команд в режиме ex, однако это требуется, чтобы редактор их выполнял.

Проблемы при сохранении файлов

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

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

  Введите `:w! имя_файла`, чтобы перезаписать существующий файл, или `:w имя_нового_файла`, чтобы сохранить отредактированную версию в новом файле.

- *Вы хотите записать файл, но у вас нет для этого разрешения. Выводится сообщение Permission denied (Отказано в доступе).*

  Используйте команду `:w имя_нового_файла` для записи содержимого буфера в новый файл. Если у вас есть разрешение на запись в каталог, вы можете использовать команду `mv`, чтобы заменить исходную версию своей копией. Если у вас нет разрешения на запись в каталог, введите команду `:w путь/имя_файла`, чтобы записать содержимое буфера в файл в каталоге, для которого у вас есть разрешение (например, ваш домашний каталог или `/tmp`). Будьте осторожны, чтобы не перезаписать какие-либо существующие файлы в данной директории.

- *Вы пытаетесь записать свой файл, но получаете сообщение, что хранилище заполнено.*

  Сегодня, когда 500-гигабайтный накопитель считается небольшим по емкости, подобные ошибки, как правило, встречаются редко. Тем не менее, если что-то подобное произойдет, у вас есть несколько вариантов решения такой проблемы. Во-первых, попробуйте записать свой файл на другой диск или в другой каталог (например, в `/tmp`), чтобы ваши данные были сохранены. Затем попробуйте сохранить содержимое буфера с помощью команды `ex :pre` (сокращенно от `:preserve`). Если и это не сработает, освободите пространство одним из следующих способов.

  - Откройте файловый менеджер с графическим интерфейсом (например, Nautilus в GNU/Linux) и поищите старые ненужные файлы, которые можно удалить.
  - Нажмите `Ctrl+Z`, чтобы приостановить работу `vi` и вернуться к командной строке. Затем стоит использовать команды Unix, чтобы найти большие файлы, которые являются кандидатами на удаление:
    - команда `df` показывает, сколько свободного места доступно на диске в данной файловой системе или в системе в целом;
    - команда `du` указывает, какой объем диска используется для данных файлов и каталогов. Команда `du -s * | sort -nr` выводит список файлов и каталогов, отсортированных по занимаемому ими пространству в порядке убывания.

  Когда закончите удалять файлы, используйте команду `fg`, чтобы вернуться к программе `vi`; затем вы можете сохранить свою работу в обычном режиме.

Помимо нажатия Ctrl+Z и управления задачами, допускается ввести `:sh`, чтобы запустить новую оболочку. Нажмите Ctrl+D или введите команду `exit`, чтобы завершить работу в оболочке и вернуться к программе `vi` (это работает и с `gvim`!).

Вы также можете ввести что-то вроде `:!du -s *`, чтобы запустить консольную команду из `vi`, а затем вернуться к редактированию, когда команда будет выполнена.

Упражнения

Единственный способ изучить программы `vi` и Vim — практиковаться. Теперь вы знаете достаточно, чтобы создать новый файл и вернуться в командную строку. Создайте файл с именем `practice`, введите текст, а затем сохраните и закройте файл.

1. Откройте файл с именем `practice` в текущем каталоге:

   `$ vi practice`

2. Перейдите в режим ввода:

   `i`

3. Введите текст:

   любой текст

4. Вернитесь в командный режим:

   Esc

5. Выйдите из `vi`, сохранив изменения:

   `ZZ`