7

Продвинутое редактирование

В этой главе рассматриваются некоторые более продвинутые возможности редакторов `vi` и `ex`. Перед началом работы с представленными здесь концепциями стоит ознакомиться с материалом предыдущих глав.

Мы разделили эту главу на пять частей. В первой рассказывается о некоторых способах настройки окружения редактора. Вы узнаете, как использовать команду `set` и как с помощью файлов `.exrc` создавать несколько различных окружений редактирования.

Вторая часть излагает, как выполнять команды UNIX непосредственно из `vi` и использовать его для фильтрации текста через команды UNIX.

В третьей говорится о разных способах сохранения длинных последовательностей команд, сводя их к аббревиатурам или даже к командам, выполняющимся по нажатию одной клавиши клавиатуры (это называется *отображение (mapping) клавиш*). Также в ней содержится раздел, посвященный @-функциям, позволяющим хранить последовательности команд в буфере.

Четвертая часть рассказывает об использовании скриптов `ex` из командной строки UNIX или из скриптов оболочки. Такие скрипты являются мощными инструментами для многократного внесения однородных правок.

В пятой части говорится о некоторых функциях `vi`, которые будут полезны программистам. В этом редакторе есть опции для управления отступами строк и опция отображения невидимых символов (например, табуляций и переносов строк). Некоторые команды поиска особенно полезны в блоках программного кода и функциях языка C.

Настройка vi

На разных терминалах vi ведет себя по-разному. В современных системах UNIX он получает рабочие инструкции о типе вашего терминала из базы данных терминалов terminfo (в старых системах vi использовал оригинальную базу данных termcap)¹.

Многие аспекты поведения vi регулируются опциями, которые задаются непосредственно в программе. Например, можно устанавливать правое поле, таким образом заставляя vi автоматически переносить строки, что позволит не нажимать на ENTER.

Вы можете менять эти опции из vi с помощью ex-команды :set. Кроме того, при запуске vi считывает файл .exrc, находящийся в домашнем каталоге, где могут содержаться различные инструкции. Прописав команды :set в этом файле, вы настроите стандартное поведение редактора под свои нужды.

Файлы .exrc можно располагать в разных каталогах, чтобы устанавливать разные опции для разных окружений. Например, для редактирования английских текстов вам потребуются одни установки, а для программного кода – совсем другие. Сначала будет обрабатываться файл .exrc из домашнего каталога, а затем – из текущего.

Наконец, при запуске vi выполняются все команды, хранящиеся в переменной окружения EXINIT. Установки в EXINIT имеют более высокий приоритет, чем установки, прописанные в файле .exrc.

Команда :set

Команда :set может менять опции двух типов: переключатели, принимающие значения on (включен) или off (выключен), и опции с числовыми и строковыми значениями (например, положение правого поля или имя файла).

По умолчанию опция-переключатель может быть в состоянии «on» или «off». Команда ее включения имеет вид:

```
:set option
```

Для выключения опции следует выполнить команду:

```
:set nooption
```

Например, чтобы включить игнорирование регистра при поиске по шаблонам, введите:

```
:set ic
```

¹ Расположение этих двух баз данных отличается у разных производителей. Попробуйте ввести команды `man terminfo` и `man termcap`, чтобы получить информацию, касающуюся вашей системы.

Чтобы снова сделать `vi` регистрозависимым при поиске, наберите команду:

```
:set noic
```

Некоторые опции требуют присвоения им определенных параметров. Например, опция `window` устанавливает количество строк, отображаемых в «окне» экрана. Значения этих параметров устанавливаются знаком равенства (=):

```
:set window=20
```

Во время работы в `vi` можно проверить, какие именно установки он использует. Команда:

```
:set all
```

отображает полный список опций, как заданных вами, так и «выбранных» редактором по умолчанию.

Список должен выглядеть примерно так:¹

<code>autoindent</code>	<code>nomodelines</code>	<code>noshowmode</code>
<code>autoprint</code>	<code>nonumber</code>	<code>noslowopen</code>
<code>noautowrite</code>	<code>nonovice</code>	<code>tabstop=8</code>
<code>beautify</code>	<code>nooptimize</code>	<code>taglength=0</code>
<code>directory=/var/tmp</code>	<code>paragraphs=IPLPPPQPP LIpplpipnbp</code>	<code>tags=tags /usr/lib/tags</code>
<code>noedcompatible</code>	<code>prompt</code>	<code>tagstack</code>
<code>errorbells</code>	<code>noreadonly</code>	<code>term=vt102</code>
<code>noexrc</code>	<code>redraw</code>	<code>noterse</code>
<code>flash</code>	<code>remap</code>	<code>timeout</code>
<code>hardtabs=8</code>	<code>report=5</code>	<code>ttytype=vt102</code>
<code>noignorecase</code>	<code>scroll=11</code>	<code>warn</code>
<code>nolisp</code>	<code>sections=NHSHH HUuhsh+c</code>	<code>window=23</code>
<code>nolist</code>	<code>shell=/bin/ksh</code>	<code>wrapscan</code>
<code>magic</code>	<code>shiftwidth=8</code>	<code>wrapmargin=0</code>
<code>nomesg</code>	<code>showmatch</code>	<code>nowriteany</code>

С помощью команды:

```
:set option?
```

можно вывести текущее значение любого параметра, а команда:

```
:set
```

показывает опции, заданные или измененные как посредством файла `.exrc`, так и во время текущего сеанса работы.

¹ Результат `:set all` сильно зависит от версии `vi`. Приведенный пример типичен только для `vi` в UNIX. Порядок следования опций – алфавитный по кодам без учета стоящих в начале `no`.

Например, ее вывод может иметь следующий вид:

```
number sect=AhBhChDh window=20 wrapmargin=10
```

Файл .exrc

Файл `.exrc`, управляющий окружением `vi`, расположен в домашнем каталоге (в который вы попадаете при входе в систему). Этот файл можно менять в редакторе, как и любой другой текстовый файл.

Если у вас нет такого файла, просто создайте его с помощью `vi`. Введите туда команды `set`, `ab` и `map`, которые вы хотите задействовать при работе в `vi` или `ex` (про команды `ab` и `map` рассказывается далее). Приведем пример файла `.exrc`:

```
set nowrapscan wrapmargin=7
set sections=SeAhBhChDh nomsg
map q :w~M:n~M
map v dwElp
ab ORA O'Reilly Media, Inc.
```

Поскольку он обрабатывается редактором `ex` перед переходом в визуальный режим (`vi`), начинать содержащиеся в `.exrc` команды с двоеточия не нужно.

Различные окружения

Вдобавок к автоматическому считыванию файла `.exrc`, расположенного в домашнем каталоге, можно разрешить `vi` считывать файл с именем `.exrc` из текущего каталога. Это позволяет задавать опции, подходящие для конкретного проекта.

Во всех современных версиях `vi` для считывания файла `.exrc` из текущего каталога нужно установить опцию `exrc` в файле `.exrc` домашнего каталога:

```
set exrc
```

Этот механизм предотвращает возможность помещения в рабочий каталог файлов `.exrc` других пользователей, команды в которых могут нарушить безопасность системы¹.

Например, в каталоге, предназначенном в основном для программирования, можно иметь следующий набор опций:

```
set number autoindent sw=4 terse
set tags=/usr/lib/tags
```

а в каталоге, используемом для редактирования текстов, — другой набор:

```
set wrapmargin=15 ignorecase
```

¹ Первоначальные версии `vi` автоматически считывали оба файла при их обнаружении. Опция `exrc` устранила эту потенциальную брешь в безопасности.

Обратите внимание, что в файле `.exrc` из домашнего каталога можно включить определенные опции, а в файле из локального каталога – отключить их.

Также можно определить альтернативные окружения `vi`, сохранив настройки в файле, отличном от `.exrc`, и считать этот файл при помощи команды `:so` (`so` – это сокращение слова `source`).

Например:

```
:so .progoptions
```

Локальные файлы `.exrc` также используются при определении аббревиатур и отображений клавиш (про которые рассказывается далее в этой главе). Например, когда мы пишем книгу, то сохраняем все используемые в ней аббревиатуры в файл `.exrc` каталога с этой книгой.

Некоторые полезные опции

Как показывает вывод команды `:set all`, в `vi` существует огромное количество опций. Одни из них предназначены для внутреннего использования в редакторе, и их значения обычно не изменяют, другие полезны в одних ситуациях, а в других – нет (например, `noredraw` и `window` могут пригодиться во время межконтинентального сеанса `ssh`). В табл. В.1 в разделе «Опции Solaris `vi`» на стр. 458 содержится краткое описание каждой опции. Мы рекомендуем уделить некоторое время на эксперименты с ними. Если вас заинтересовала какая-нибудь опция, попробуйте ее включить (или выключить) и посмотреть, что изменилось в редакторе. Таким образом можно обнаружить много полезного.

В разделе «Перемещение в строке» на стр. 35 мы уже говорили о том, что одна из этих опций – `wrapmargin` – особенно важна при редактировании текста, не являющегося программой. Она определяет размер правого поля, используемого для автоматического переноса текста (при этом можно не вводить возврат каретки вручную). Обычно этот параметр устанавливают в диапазоне от 7 до 15:

```
:set wrapmargin=10
```

Три другие опции определяют поведение `vi` при выполнении поиска. Обычно поиск различает прописные и строчные буквы (`foo` не совпадает с `Foo`), циклически возобновляется с начала файла (то есть вы можете начать поиск из любого места файла, но отыскать при этом все вхождения) и распознает метасимволы при поиске по шаблону. Этими параметрами управляют опции `noignorecase`, `wrapscan` и `magic` соответственно. Чтобы изменить любую из этих установок по умолчанию, можно включить противоположные опции, такие как `ignorecase`, `nowrapscan` и `nomagic`.

Программистов могут заинтересовать опции `autoindent`, `showmatch`, `tabstop`, `shiftwidth`, `number` и `list`, а также их антиподы.

Наконец, рассмотрим использование опции `autowrite`. Когда она установлена, `vi` автоматически записывает содержимое измененного буфера

при вызове команды `:n` (next), переключающей на следующий редактируемый файл, или при запуске команды оболочки через `!.`

Вызов команд UNIX

Во время редактирования в `vi` можно отображать или считывать результаты выполнения любой команды UNIX. Восклицательный знак (!) предписывает `ex` создать оболочку и воспринимать все дальнейшие символы как команду UNIX:

```
!:command
```

Так что если во время редактирования вам захочется посмотреть время или дату, не выходя из `vi`, то можете ввести:

```
!:date
```

На экране появятся текущие время и дата. Нажмите `ENTER` для продолжения редактирования файла с того места, где вы остановились.

Если нужно выполнить несколько команд UNIX друг за другом, не возвращаясь к редактированию в `vi` между ними, то можно создать оболочку командой `ex`:

```
:sh
```

Для выхода из оболочки и возврата в редактор нажмите `CTRL-D`.

Вызов UNIX можно сочетать с `:read`, чтобы результаты выполнения команды UNIX считались в ваш файл. Следующий простой пример:

```
:r !date
```

считывает системную информацию о дате в текст файла. Если перед командой `:r` поставить адрес строки, то результат ее выполнения будет вставлен в файл по указанному адресу (по умолчанию он появляется в текущей строке).

Предположим, вы редактируете файл и хотите вставить туда четыре телефонных номера из файла `phone` в алфавитном порядке. Содержимое файла `phone` таково:

```
Willing, Sue 333-4444  
Walsh, Linda 555-6666  
Quercia, Valerie 777-8888  
Dougherty, Nancy 999-0000
```

Команда:

```
:r !sort phone
```

считает это содержимое после применения к нему фильтра `sort`:

```
Dougherty, Nancy 999-0000  
Quercia, Valerie 777-8888  
Walsh, Linda 555-6666  
Willing, Sue 333-4444
```

Допустим, вы редактируете файл и хотите вставить текст из другого файла в том же каталоге, но не можете вспомнить имя этого файла. *Можно* поступить неэффективно: выйти из *vi*, выполнить команду *ls*, увидеть правильное имя файла, снова зайти в файл и найти нужное место.

Но то же самое можно сделать за меньшее число шагов:

Клавиши	Результат
<code>!ls</code>	<pre>file1 file2 letter newfile practice</pre> <p>Выводит список файлов в текущем каталоге и находит имя нужного файла. Нажмите ENTER для продолжения редактирования.</p>
<code>:r newfile</code>	<pre>"newfile" 35 lines, 949 characters</pre> <p>Считывает новый файл.</p>

Фильтрация текста с помощью команды UNIX

Можно послать фрагмент текста в качестве стандартного ввода в команду UNIX. Ее результат заменит соответствующий текстовый блок в буфере. Фильтровать текст таким образом можно как в *ex*, так и в *vi*. Главное отличие между ними состоит в том, что для *ex* вы выделяете текстовый блок с помощью адресов строк, а в *vi* — как текстовый объект (командами перемещения).

Фильтрация текста в *ex*

Первый пример показывает, как фильтровать текст в *ex*. Предположим, список имен в предыдущем примере содержится не в отдельном файле *phone*, а уже в текущем файле в строках с 96-й по 99-ю. Пользователю просто нужно ввести желаемые адреса строк, после чего поставить восклицательный знак и нужную команду UNIX.

Например, команда:

```
:96,99!sort
```

Пропустит строки с 96-й по 99-ю через фильтр *sort* и заменит их на результат команды *sort*.

Фильтрация текста в *vi*

В *vi* текст отфильтровывается через команду UNIX, если ввести восклицательный знак, после чего указать любые комбинации клавиш *vi*, задающие текстовый блок, а затем выполняемую команду UNIX. Например:

```
!)command
```

передаст *command* текст до конца предложения.