С ЮНИКСОМ НА vi

СЕРГЕЙ СУПРУНОВ

Так уж исторически сложилось, что операционные системы UNIX считаются очень сложными и недружественными по отношению к пользователям. И одним из олицетворений этого часто называют редактор vi. По мере того, как набирает обороты Linux, позиционируемый как система более дружественная, среди его пользователей все большую популярность приобретают более привычные редакторы, особенно тот, который встроен в Midnight Commander (поскольку всегда под рукой), и старый добрый vi начинает забываться. Тем не менее, этот редактор обладает непревзойденной на сегодняшний день мощью, функциональностью, универсальностью и удобством работы, а кроме того, разработан в полном соответствии с идеологией UNIX. Чтобы не быть голословным, приведу несколько примеров (синтаксис команд редактора, которые в них встретятся, будет подробнее рассмотрен ниже):

- Задача: в середину некоторой статьи нужно вставить вывод команды «ipfw show» (или любой другой). Попробуйте сделать это, используя ваш любимый редактор, а затем посмотрите, как это делается в vi.
  Конечно же, будучи квалифицированным пользователем, вы не стали вбивать результат вручную, а перенаправили вывод в файл, который объединили с файлом статьи:

```
# ipfw show > buffer
# cat article buffer > temp
# mv temp article
```

  после чего, воспользовавшись функциями редактирования блоков (ваш же редактор позволяет это?), перенесли блок в нужное место. Конечно, с помощью Midnight Commander можно сделать то же самое проще:

```
# ipfw show > buffer
# mcedit article
```

  Идем в точку, в которую нужно вставить блок, затем жмем F9 и выбираем в меню «Файл» пункт «Вставить файл…», указываем путь к файлу buffer.
  Ну а с помощью vi это выполняется одной командой:

```
!ipfw show
```

- Предположим, что у вас есть пронумерованный список из 12 пунктов. Вам нужно вставить два пункта, начиная с третьей позиции, и соответственно поправить номера всех нижележащих пунктов.
  Наверняка решать эту задачу вам придется руками. Vi позволяет сократить число операций до минимума. Для первого пункта списка, номер которого нужно поменять, ставим курсор на его номер и даем команду «2#+», после чего на номере каждого последующего пункта просто давим символ «.».
- Двадцать четыре строки в тексте нужно сдвинуть вправо символом табуляции.
  Конечно, не проблема минутку пощелкать «Tab – Стрелка вниз – Home», но vi предлагает более элегантный способ: «23>j» над первой строкой. И все.

Надеюсь, мне удалось убедить вас в том, что vi – один из лучших текстовых редакторов для систем UNIX. Ниже мы более подробно рассмотрим его особенности, синтаксис некоторых команд, коротко коснемся режимов ex и view. Чтобы учесть интересы читателей с разным уровнем подготовки, данная статья будет разбита на следующие подразделы:

- основы редактора vi;
- команды перемещения и поиска;
- команды редактирования и форматирования;
- работа с блоками и буфером;
- команды режима редактирования;
- прочие команды режима visual;
- команды режима ex;
- установки редактора;
- заключение.

Мне не хочется перегружать статью снимками экранов, поэтому, если вы не очень хорошо знакомы с vi, настоятельно рекомендую вам читать ее перед монитором и сразу пробовать те примеры, которые будут описаны ниже, чтобы «почувствовать» этот редактор.

Основы редактора vi

Итак, vi – текстовый редактор, входящий в поставку практически всех операционных систем семейства UNIX. Запуск его осуществляется командой:

```
# vi [filename]
```

Помимо основного (visual) режима он может быть запущен в командном режиме, называемом ex (от англ. execute), ориентированном на работу со строками, и в режиме view (только для чтения). Запуск этих режимов осуществляется с ключами –e и –R, или командами ex и view соответственно:

```
# {vi -e | ex} [filename]
# {vi -R | view} [filename]
```

Если файл с таким именем существует в текущей директории, то он будет открыт для редактирования. Ну а если его нет, то будет создан новый файл. Если команда vi дана без параметров, будет запущено редактирование нового файла, создаваемого во временном каталоге, заданном переменной окружения TMPDIR (обычно это папка /tmp) с именем типа vi.L14259 (часть после точки создается случайным образом, чтобы обеспечить уникальность создаваемого файла). В дальнейшем вы сможете сохранить набранный файл под любым именем.

Если вы последовали моей рекомендации и сразу проверяете все на практике, то первое, чему вы должны научиться, это выходить из редактора. Дело в том, что в vi вам не помогут ни <escape>, ни <Alt+X>, ни даже <Ctrl+C>. Нет в нем и так любимой народом верхней строчки, именуемой «Главное меню». Также он не будет отображать то, что вы попытаетесь набрать (хотя, если в панике начать беспорядочно стучать по клавиатуре, то может случиться «чудо» и ваш набор станет появляться на экране, однако выйти из редактора это вам все равно не помо-

жет). Так что абсолютно черный экран с одинокими тильдами по левому краю, никак не реагирующий на нажатия клавиш и не позволяющий вернуться в оболочку иначе, как «килянием» процесса из соседнего терминала (если вы, конечно, умеете пользоваться командой kill – иначе только Reset), может надолго отбить охоту запустить этот редактор еще раз. Поэтому сначала ознакомьтесь со следующей таблицей, которая поможет вам поверить в то, что vi – это действительно редактор:

Команда	Описание
:q!	Выход без сохранения
:w	Сохранение изменений
:w <filename>	Сохранение как <filename>
:wq	Выход с сохранением
:q	Выход, если файл не изменялся
i	Переход в режим вставки символов в позиции курсора
a	Переход в режим вставки символов в позиции после курсора
o	Вставка строки после текущей
O	Вставка строки над текущей
x	Удаление символа в позиции курсора
dd	Удаление текущей строки
u	Отмена последнего действия
<escape>	Возврат в режим команд

Команды, начинающиеся с символа «:», будут отображаться в нижней строке. Остальные выполняются «молча». Редактор vi имеет два режима работы – режим команд и режим редактирования.

Запускается он в командном режиме, так что все нажатия на клавиши трактуются как команды. Нажатие клавиш «i», «a», «o», «O» и ряд других переводят vi в режим вставки, когда набираемые символы трактуются как текст и отображаются на экране.

Возврат к режиму команд выполняется клавишей <escape> или в некоторых случаях автоматически, например, при попытке передвинуть курсор левее первого символа в строке (в редакторе vim, являющемся модернизированным вариантом vi и часто заменяющем его в Linux, в этом случае редактор остается в режиме вставки). Автоматический переход в командный режим обычно сопровождается звуковым сигналом, как и ошибочная команда.

Чтобы почувствовать все это, выполним небольшое практическое упражнение. Находясь в своем домашнем каталоге, запустите редактор командой:

```
# vi test.txt
```

Далее нажмите «i», чтобы перейти в режим вставки. Теперь все нажатия на клавиши будут трактоваться как ввод текста, и символы будут отображаться на экране с позиции курсора.

Наберите «Hello, world!». Постарайтесь не ошибаться, поскольку исправление ошибок, как и все остальное, имеет здесь свои особенности, о которых мы поговорим ниже. Нажмите <escape> для возврата в командный режим. Наберите «:wq» и нажмите <enter>. Убедитесь, что файл test.txt действительно создался. После этого небольшого задания вам будет проще представлять то, о чем пойдет речь далее.

Аналогично команде «i», в режим вставки можно перейти, нажав клавишу «a». Единственное отличие – текст будет вставляться не перед символом, на котором находится курсор, а после него. Кроме того, режим вставки может быть вызван командами «o» и «O». Первая из них добавляет пустую строку после, а вторая – перед текущей строкой, и дальнейший ввод символов трактуется как ввод текста.

Чтобы удалить символ, нужно перейти в режим команд и над удаляемым символом нажать клавишу «x». В режиме вставки удалить только что введенный ошибочно символ можно клавишей <backspace>, однако в vi таким образом может быть удалена только последняя непрерывно введенная последовательность символов. То есть если вы откроете для редактирования наш тестовый файл со строкой «Hello, world», и добавите между словами слово «my»: «Hello, my world», то, используя клавишу <backspace>, вы сможете удалить только что введенные символы « my», а вот запятую и последующие символы удалить таким образом уже не удастся. В этом случае придется использовать команду «x».

Удалить целиком строку, на которой находится курсор, можно командой «dd» (просто нажмите два раза клавишу <d>). Помните, что в vi строкой считается не экранная строка, а последовательность символов до перевода строки (\n). Если строка больше 80 символов (значение по умолчанию), то она переносится на новую линию (строку экрана). Используя «dd», вы удалите всю строку вне зависимости от того, на скольких экранных линиях она размещается.

Чтобы определить, где находится конец строки, нажмите клавишу «$» (вы, должно быть, уже заметили, что клавиши <Home>, <End> и т. п. тут не работают). При навигации по экрану (можно пользоваться стрелками, хотя есть и более «правильный» способ) курсор перемещается не по экранным линиям, а по строкам текста.

Если вы что-то сделали не так, то отменить последнюю операцию можно командой «u». Эта команда отменяет только последнее действие, то есть ее повторное применение отменит только что сделанную отмену. Конечно, отсутствие истории операций – один из серьезных недостатков vi, однако если работать вдумчиво и внимательно, то он почти незаметен. Собственно, идеология UNIX-систем такова, что они не особо балуют пользователя подсказками и возможностью отката выполненных операций, так что сразу настраивайтесь на ответственную работу.

Учтите, что по команде «u» отменяется вся команда целиком. Например, если вы дали команду «i», набили 3-й том «Войны и мира» и вернулись в режим команд, то «u» отменит весь этот ввод. Если ошибок получилось слишком много, можно выйти из редактора без сохранения сделанных изменений (команда «:q!»).

В принципе этих сведений достаточно, чтобы отредактировать файл. Однако вряд ли вы поверили, что vi – удобный редактор. Понимаю, как вам хочется нажать <F4> в своем любимом mc, но все же найдите в себе силы дочитать статью до конца и попрактиковаться с vi хотя бы пару недель. Думаю, ваше мнение изменится.

В дальнейшей части статьи некоторые команды и их применение будут рассмотрены более подробно. За описанием всех остальных команд отсылаю читателя к стра-

ницам справочного руководства «man vi», в полном соответствии с идеологией UNIX.

Команды перемещения и поиска

Эта часть статьи посвящена командам навигации, которые позволят вам быстро и эффективно перемещаться по редактируемому тексту. Как и все в vi, эти команды также имеют свою специфику, и их удобство и продуманность начинают осознаваться только через несколько месяцев работы. Хотя и новичок найдет среди них полезные и интересные.

В общем случае по тексту можно перемещаться, используя клавиши со стрелками. Но vi разработан таким образом, чтобы в процессе работы руки не покидали основной рабочей зоны клавиатуры. Те, кто владеет методом «слепой» печати, по достоинству оценят эту особенность. Кроме того, это позволяет не заботиться о совместимости клавиатур и терминалов – vi будет полностью функционален даже на самых тупых терминалах.

Команды навигации и поиска представлены ниже. Запись «№» означает число, которое набирается перед вводом команды (при этом на экране оно не отображается) и задает число повторений команды или в некоторых случаях номер строки, к которой команда должна быть применена. Если число не задано, команда выполняется один раз.

Команды перемещения курсора

- [№]h – перемещает курсор влево на № символов;
- [№]k – перемещает курсор на № символов вверх;
- [№]j – перемещает курсор вниз на № символов;
- [№]l, [№]<пробел> – перемещает курсор на № символов вправо;
- [№]$ – переход на последний символ строки, №-й после текущей;
- 0 (ноль) – переход на первый символ текущей строки;
- ^ – перемещает курсор на первый символ текущей строки, отличный от пробельного;
- [№]- – переход на первый непробельный символ №-й перед текущей строки;
- [№]+ – переход на первый непробельный символ №-й после текущей строки;
- [№]_ – переход на первый непробельный символ (№-1)-й после текущей строки.

Также существует масса команд для перемещения по словам, группам символов, предложениям, параграфам. При этом под «словом» будет пониматься последовательность символов, разделенных пробельными символами. Термином «группа символов» будем именовать последовательность символов, не разделенных специальными символами (такими как дефис, точка, запятая и т. д.). В терминах vi эти две единицы именуются «большим словом» (bigword) и «словом» (word) соответственно. Предложение – последовательность слов, ограниченная точкой или пустой строкой. Параграф – часть текста, обрамленная пустыми строками.

Чтобы вы знали, что искать, просто перечислю эти команды: B, W, E, b, w, e, (,), {, }, <Ctrl-F>, <Ctrl-B>, <Ctrl-D>, <Ctrl-U>, <Ctrl-E>, <Ctrl-Y>. Познакомиться с ними можно на страницах «man vi» или методом «научного тыка». Например, команда «5<Ctrl-Y>» прокрутит экран на 5 строк вниз, не перемещая курсор (очень полезна, если вам нужно увидеть несколько строк выше и затем продолжить редактирование текущей строки).

Еще несколько полезных команд навигации:

- z. – прокручивает текст так, что текущая строка становится в центре экрана;
- [№]G – перемещает курсор на №-ю строку от начала файла, если № не задано – на конец файла;
- [№]H – перемещает курсор на №-ю сверху строку, видимую на экране;
- [№]L – перемещает курсор на №-ю снизу строку, видимую на экране;
- M – перемещает курсор на строку, расположенную в центре экрана.

С помощью следующих двух команд вы сможете расставлять «маркеры» в тексте, и затем быстро переходить на эти метки:

- m<симв.> – запоминает текущую позицию курсора как символ <симв.>;
- `<симв.> – возвращает курсор на позицию, запомненную как <симв.>.

Например, запомнив начало второго параграфа как «m2», вы в дальнейшем сможете возвращаться к нему командой «`2».

Команды поиска

С поиском вроде все понятно, ничего пояснять не буду:

- /<образец поиска> – поиск в тексте по образцу;
- / – повторный поиск по предыдущему образцу (найти далее);
- ?<образец поиска> – поиск по образцу в обратном направлении;
- ? – повтор поиска по предыдущему образцу в обратном направлении.

Предыдущие четыре команды должны обязательно завершаться нажатием клавиши <Enter>.

- [№]<Ctrl-A> – поиск №-го слова, совпадающего с тем, на котором стоит курсор, начиная с позиции курсора в сторону конца документа;
- [№]f<char> – перемещает курсор на №-й после курсора символ <char> в строке;
- [№]t<char> – перемещает курсор на символ, стоящий перед №-м после курсора символом <char> в строке;
- [№]F<char> – перемещает курсор на №-й перед курсором символ <char> в строке;
- [№]T<char> – перемещает курсор на символ, стоящий после №-го перед курсором символа <char> в строке.

Команды редактирования и форматирования

Помимо рассмотренных выше команд «i» и «a», полезны будут и следующие:

- [№]I – включает режим вставки текста. Текст будет вводиться с начала строки;

- [№]A – включает режим вставки текста. Текст будет вводиться после последнего символа в текущей строке;
- [№]o – вставка новой строки после текущей. Текст будет вводиться с начала новой строки;
- [№]O – вставка новой строки выше текущей. Текст будет вводиться с начала новой строки.

Рассматривая две предыдущие команды, необходимо указать на одну особенность.

Использование оных со счетчиком № не добавляет № пустых строк, как следовало бы ожидать. Добавляется одна, редактор переходит в режим ввода, а затем, после возврата в командный режим, введенный блок текста будет повторен № раз. Кстати, и команды вставки ведут себя аналогичным образом – сначала вы получаете возможность вставить или добавить текст, начиная с соответствующей позиции курсора, после чего, в момент возврата в режим команд, введенный вами текст будет продублирован № раз.

Выполните команду «5i», введите текст «echo », вернитесь в режим команд (<escape>) и посмотрите, что из этого получится. Данный комментарий относится и к ряду команд, перечисленных ниже.

- [№]r<char> – заменяет символ в точке нахождения курсора (и №-1 последующих символов) символом <char>;
- [№]~ – заменяет текущий и следующие №-1 символов этими же символами в другом регистре;
- [№]s – заменяет № символов в строке, начиная с текущего, вводимым далее текстом. Вводимые символы после №-го добавляются после замененных. Граница области замены отмечается символом «$».

Поясним эту команду. Пусть у нас есть строка с текстом «This is a big string».

Мы хотим слово «big» заменить более справедливым «small». Поставим курсор на букву «b» (естественно, используя команду «fb» – мы же уже не маленькие, чтобы стрелочками по тексту скакать). Заменить нам нужно три символа, поэтому: «3s». Теперь просто вводим наше «small» – первые три символа введутся в режиме замены, последующие добавятся, не затирая то, что нам нужно. Теперь <escape>, и любуемся на дело наших рук, не забывая при этом громко восхищаться редактором.

- [№]R – включает режим замены (вводимые символы будут замещать текущие до конца строки, затем символы будут добавляться);
- [№]S – очищает строку и переходит в режим вставки текста (аналогичный результат достигается последовательным выполнением двух команд «[№]dd» и «O»). Обратите особое внимание, что № в данном случае относится к удалению, то есть № строк будут удалены, и вместо них можно будет ввести один блок текста.
- [№]C – удаляет символы от текущей позиции курсора до конца строки и переходит в режим вставки текста. № также относится к удалению.
- [№]c<направление> – удаляет символы в указанном направлении, которое задается командами перемещения курсора (№ отдельных символов для направлений «влево» и «вправо» и № строк для «вверх» и «вниз») и переводит редактор в режим вставки текста.

Все вышеприведенные команды переводят vi в режим вставки текста, то есть весь последующий ввод будет отображаться на экране начиная с указанной позиции. Для возврата в командный режим используется комбинация <Ctrl-C> или клавиша <escape>.

- [№]x – удаляет № символов после курсора (начиная с позиции курсора);
- [№]X – удаляет № символов перед курсором;
- [№]d <направление> – удаляет № символов/строк относительно курсора в указанном направлении. Направление задается командами управления курсором «h», «j», «k», «l» или стрелками. Если выбрано «влево» или «вправо», то удаляются № символов соответственно перед курсором или после него. Направление, заданное как «вверх» или «вниз», позволяет удалить № строк соответственно выше или ниже текущей строки (включая текущую);
- [№]D – удаляет символы начиная с позиции курсора до конца строки;
- [№]J – объединяет текущую строку с № следующими в одну;
- [№]> <направление> – сдвигает № строк вправо на символ табуляции;
- [№]< <направление> – сдвигает № строк влево на символ табуляции.

В предыдущих двух командах (впрочем, как и в остальных) направление задается командами управления курсором «h», «j», «k», «l» или стрелками. Так, если выбрано «вправо» или «влево», то команда сдвига действует только на текущую строку, № игнорируется. Если «вверх» или «вниз», то сдвигается данная строка и № предыдущих или последующих соответственно.

- [№]#{#|+|-} – инкремент/декремент числа. Если дать эту команду, когда курсор стоит под числом, то это число будет увеличено на № (если задан параметр «#» или «+») или уменьшено на № (если задан параметр «-»). Например, чтобы увеличить число 52 на 7, ставим курсор на символ «5» или «2» (он может быть установлен на любую цифру числа), после чего набираем последовательно «7#+». В результате «52» изменится на «59». В vim эта функция не поддерживается.
- [№]!<направление><команда shell> – заменяет в документе начиная с позиции курсора в заданном направлении № строк выводом команды оболочки. Если вам нужно вставить вывод, можно сначала создать две пустые строки, а затем выполнить данную команду, например: «2o», <escape>, «!kwho».

Работа с блоками и буфером

Vi предоставляет пользователю весьма мощные функции по работе с блоками.

- [№]yy – копирует в буфер № строк, начиная с текущей;
- [№]Y – копирует № строк в буфер (аналогична «yy»);
- [№]y<направление> – копирует текст в буфер в указанном направлении. № трактуется в зависимости от направления (впрочем, как и в остальных подобных командах): количество копируемых символов для «вправо» и «влево» и число строк (дополнительно к текущей) для «вверх» и «вниз». Например, «2yk» скопирует в буфер текущую строку и еще две, расположенные выше (итого – три). Если вам нужно скопировать только текущую строку, следует использовать команду «yy» или «Y»;
- [№]p – вставляет текст из буфера № раз после курсора;
- [№]P – вставляет текст из буфера № раз перед курсором.

Используя команды расширенного (ex) режима, можно работать с несколькими блоками. Где изучить эти возможности, вы, думаю, уже знаете. Конечно же, в «man vi».

Команды режима редактирования

Находясь в режиме редактирования, редактор все вводимые символы будет отображать на экране как часть текста. Однако существует несколько последовательностей, которые трактуются как специальные команды и для которых выполняется автоматическая замена введенных символов последовательности результатом выполнения команды. Наиболее полезные из них следующие:

- <escape> – завершает режим ввода текста и переводит редактор в командный режим;
- <Ctrl-C> – также возвращает редактор в режим команд (при «слепой» печати эта команда более удобна);
- <backspace> – удаление только что введенного символа;
- <Ctrl-W> – удаление только что введенного слова;
- <Ctrl-X>[0-9A-Fa-f] – вставка символа, имеющего код, выраженный шестнадцатеричным числом (например, «<Ctrl-X>30» отобразится на экране как «^X30» и автоматически заменится символом «0», ASCII-код которого – 30h).

Прочие команды режима visual

Осталось рассмотреть еще несколько команд, которые трудно выделить в ту или иную категорию, но которые весьма полезны при работе.

- [№]. – повтор последней команды для текущей позиции курсора № раз;
- u – отменяет последнее действие;
- U – восстанавливает текущую строку, отменяя все изменения, сделанные в ней;
- Q – переключение в интерфейс ex (см. далее);
- <Ctrl-G> – выводит информацию о редактируемом файле (состояние – modified/unmodified, отражающее, есть ли в файле несохраненные изменения; номер текущей строки и общее количество строк в файле);
- <Ctrl-R> – перерисовывает экран (может пригодиться на плохом канале связи или для восстановления экрана после вывода на него системных сообщений, если вы работаете с физической консоли);
- <Ctrl-Z> – временное прерывание сессии редактирования и выход в командную оболочку. Чтобы вернуться назад в редактор, нужно воспользоваться системной командой «fg», которая переводит фоновый процесс в активный. Номер задачи (job), соответствующий отложенному процессу редактирования, который потребуется ввести как параметр команды «fg», можно уточнить командой «jobs». При попытке выйти из оболочки командой «exit» вы получите предупреждение, что у вас остались незавершенные задачи:

```
$ exit
You have stopped jobs.
$ jobs
[1] 34336 Suspended          vi test
$ fg 34336
```

Здесь test – имя редактируемого файла. Нужно заметить, что если вы вызываете vi из Midnight Commander, то в списке приостановленных заданий вместо «vi test» будет отображаться «midc» («mc» для Linux). После этих действий вы вернетесь назад в vi и сможете завершить редактирование. Если отложенное задание только одно, можно вернуться к нему командой fg без параметров.

- ZZ – выход из редактора с сохранением изменений (аналог команды «:wq»).

Команды режима ex

Редактор vi имеет помимо рассмотренного еще один интерфейс, или режим, – ex. Запустить редактор в этом режиме можно одной из следующих команд:

```
# vi -e
# ex
```

Кроме того, вы можете перейти в него из режима visual по команде «Q». Данный интерфейс предоставляет ряд расширенных возможностей для обработки текста, таких как работа с несколькими буферами, открытие других файлов в текущем сеансе, изменение настроек редактора и т. д. В данном разделе будут рассмотрены лишь наиболее полезные из них.

В отличие от интерфейса vi, ex ориентирован на командную работу со строками. Редактирование идет как бы вслепую. Редактируемый текст на экране по умолчанию не отображается, вам придется специально вызывать на экран требуемые строки. Почти все ex-команды можно исполнить из vi-интерфейса, предварив команду двоеточием. Наиболее характерный пример: команда – выход из редактора: «:q». Ниже приведено лишь несколько полезных команд:

- :!<команда> – выполняет команду оболочки (результат просто выводится на экран);
- :[<start>] # [№] – выводит на экран № строк начиная со <start> (в режиме ex);
- :s/<текст1>/<текст2> – замена в текущей строке <текст1> на <текст2>;
- :s – повтор предыдущей замены для текущей строки;
- :%s/<текст1>/<текст2> – замена во всем файле <текст1> на <текст2>;
- :next – переход к редактированию следующего файла из списка параметров, с которым был вызван редактор.
- :prev – возврат к редактированию предыдущего файла.

Например, если вы вызываете редактор командой «vi file1 file2», то в процессе редактирования вы сможете переходить между файлами указанными выше командами. При этом содержимое буферов обмена будет сохраняться, и вы сможете вставлять в файл фрагменты, скопированные из другого файла. Заметьте, что редактор не позволит вам перейти к другому файлу, пока изменения в текущем не будут сохранены или отменены (отменить все изменения можно командой «:e!», которая перечитает редактируемый файл с диска).

- :edit <file> – открыть для редактирования файл <file>;
- :w[<filename>] – уже знакомая вам команда сохранения редактируемого файла. Опциональный параметр <filename> превращает команду в команду «Сохранить как…» для записи изменений в файл с другим именем.

Еще одно пояснение – очень часто бывает, что, отредактировав тот или иной файл (например, squid.conf), при попытке сохранить сделанные изменения вы получаете сообщение, что файл недоступен для записи. Причина понятна – или вы забыли войти как root, или файл имеет права «r--r--r--». Выходить без сохранения, менять права и редактировать снова – жалко… А вот сохранить файл под другим именем, а затем, обретя требуемые права, заменить им оригинал – как раз то решение, которое нас устраивает. Конечно, можно с другого терминала поменять права на сохраняемый файл и снова провести операцию записи. Но при удаленной работе открыть новую ssh-сессию не всегда проще, чем поступить описанным выше образом. Да и удаленный доступ к нескольким терминалам иногда запрещают из соображений безопасности. Кроме того, если вы внесли изменения в системный файл (скажем, /etc/crontab), будучи зарегистрированным как простой пользователь, то вряд ли безопасно менять к нему права доступа, пусть и кратковременно. Кроме того, статья рассматривает редактор vi, а потому примеры призваны прежде всего показать его пригодность для решения тех или иных задач, пусть и не всегда оптимальным образом.

- :g /<шаблон>/ <команда> – указанная команда, которой может быть одна из ex-команд, применяется к строкам, которые соответствуют шаблону. Например, команда «:g /qwerty/ delete» удалит все строки, в которых присутствует последовательность символов «qwerty»;
- :v /<шаблон>/ <команда> – работает аналогично предыдущей, но команда применяется к тем строкам, которые не соответствуют шаблону;
- :vi – переход в интерфейс visual.

Опции и параметры редактора

Было бы странно, если бы оказалось, что столь мощный редактор не позволяет настраивать себя под требования и предпочтения конкретного пользователя. Однако не следует думать, что настройкой можно превратить vi в редактор ее или что-то подобное. Следующие опции позволяют лишь добавить «удобства», не меняя общих принципов работы. Все они устанавливаются с помощью ex-команды «:set». «no» перед именем опции отключает ее.

- :set [no]list – включает режим отображения служебных символов (таких как конец строки, который будет отображаться символом «$», и т. д.);
- :set [no]nu – включает отображение номеров строк;
- :set [no]showmode – включает отображение режима (command, insert, append, replace) в нижнем правом углу экрана, в котором редактор находится в данный момент;
- :set [no]verbose – включает режим расширенных сообщений (на каждое ошибочное действие будет выдаваться соответствующее пояснение).

За остальными настройками, как всегда, – «man vi», соответствующий раздел.

Увековечить сделанные настройки можно в exrc-файлах. Данные файлы применяются в следующей последовательности:

- /etc/vi.exrc – глобальный файл настроек редактора;
- $HOME/.exrc – пользовательский файл настроек;
- .exrc – локальный файл настроек для текущей директории.

Действовать будут те настройки, которые применены последними. В эти файлы следует занести те ex-команды, которые должны быть исполнены при открытии редактора. Например, если вы хотите, чтобы редактор всегда работал с включенным отображением режима и расширенными сообщениями, то создайте в своем домашнем каталоге файл .exrc следующего содержания:

```
set showmode
set verbose
```

Теперь каждый раз при вызове редактора эти команды будут отрабатываться автоматически.

Заключение

Итак, вы в общих чертах познакомились с редактором vi. Приведу еще несколько преимуществ редактора vi, помимо его функциональности:

- способность работать практически на любых терминалах;
- обязательное наличие в любой UNIX-системе;
- надежная работа даже на самых плохих линиях;
- используется в некоторых системных командах (например, vipw), следовательно, умение в нем работать потребуется в любом случае;
- а сможете ли вы просмотреть файл, являющийся каталогом (в UNIX каталог – это специальный файл), с помощью «ee» или «mcedi»? Хотя редактировать такой файл вам и vi не позволит, но посмотреть для интереса – можно (например, «view /etc»).

Попытайтесь с ним подружиться!

P.S. Если вы все же отважились сделать vi вашим основным редактором, для этого установите значение переменной $EDITOR=vi (например, для оболочки sh нужно поправить файл .profile в вашем домашнем каталоге, для csh – это файл .cshrc). Также в настройках Midnight Commander отмените использование встроенного редактора (пункт меню «Настройки» – «Конфигурация…»), и по <F4> будет вызываться тот редактор, который задан в переменной $EDITOR.