Vim учебник

Продвинутый текстовый редактор – кошмар для случайного пользователя. Если его удается в конце концов закрыть, то вздыхаешь с облегчением и твердой уверенностью, что больше не будешь запускать эту программу ни разу в жизни.

Пожалуй, Vim нельзя просто запустить и начать работать – требуется определенная подготовка. Зато стоит раз испытать его в деле и поражаешься тому, как можно было без него обходиться.

Для быстрого ознакомления с функциональностью Vim запустите команду vimtutor.

Содержание

Начнем с примера

Типичная сессия работы с Vim выглядит так:

```
#vim httpd.conf
```

В открывшемся редакторе:

```
/SSLVerif
nwr1ZZ
```

В данном случае задачей было изменить значение определенного параметра конфигурации apache. В обычном редакторе подобный результат достигается следующими действиями:

- Открыть файл «httpd.conf»
- Открыть диалог поиска
- Ввести «SSLVerif».
- С помощью кнопки "Далее", перейти ко второму по счету вхождению искомой строки.
- Поставить курсор в нужном месте
- Удалить "0" и напечатать "1"
- Сохранить изменения
- Закрыть файл

Объяснение команд VIM:

- / — Перейти в режим поиска
- Клавиша **Enter** — Искать'.
- n — Перейти к следующему вхождению искомой строки
- w — Переместить курсор к следующему слову
- r — Заменить знак под курсором
- ZZ — Сохранить изменения и выйти
- ZQ — Выйти без сохранения изменений

Введение

Здесь я делюсь опытом работы в Vim'е. Предполагается, читатель имеет установленные vi и vim, а также знаком с основами работы с UNIX-подобной системой.

Помнится, когда товарищ установил FreeBSD на мою домашнюю ЭВМ, некоторые файлы открывались непонятным мне образом. Я быстро старался избавиться от этого несчастья. Через некоторое время я понял, что происходит: в системе по умолчанию, в переменной окружения EDITOR, установлен редактор vi. Я сразу поменял его на 'ee' (easy editor). Это действительно простой редактор, затруднений с ним возникнуть просто не может, однако и особой свободы в нем не испытать. Вскоре его стало не хватать…

Описание работы с редактором я привожу в хронологическом порядке – от простого к сложному. Так как Vim на 99 % совместим с Vi, сначала я дам краткое описание работы с vi.

Легенда сокращений и специальных символов:

```
Строки, начинающиеся с символа "#" - командная строка;
<ESC> - нажатие клавишу Escape (Ctrl-[ или Ctrl-c);
<CR> - ... Enter;
<SHIFT> - ... Shift
^x - ... Ctrl-x
```

Vi

Главное отличие Vi от других редакторов в том, что у него несколько режимов работы. При этом привычный нам режим ввода текста – не основной. Вместо этого, в обычном режиме vi воспринимает каждое нажатие на клавишу, как команду. Это позволяет мгновенно перемещаться по файлу и быстро осуществлять редактирование.

Насколько Vi быстрее других редакторов? – Давайте сравним. Например, вот действия, которые необходимо выполнить [«продвинутому»] пользователю, чтобы поменять местами две буквы в обычном редакторе и в vi:

Обычный редактор:

```
<Shift-Стрелка вправо>
<Ctrl-x>
<Стрелка влево>
<Ctrl-v>
```

Vi:

```
xp
```

Переключение режимов

- **«Обычный режим»** — перемещение по файлу, стирание текста и другие редактирующие функции. Переход в него из любого другого режима:

```
<ESC>, иногда 2 раза;
<Ctrl-[>
<Ctrl-c>
```

- **«Режим ввода»** — ввод текста. Заметьте, что стирание и ввод текста происходит в двух разных режимах. Переход в него из обычного режима:

```
i
<Insert>
```

- **«Командный режим»** — Команды (операции с файлом, поиск и замена, настройка редактора…). Переход в него из обычного режима:

```
:
```

- **«Режим поиска»** — ввод поискового запроса. Переход в него из обычного режима

```
/
```

- **«Визуальный режим»** — режим выделения текста. Переход в него из обычного режима

```
v
```

Открыть/создать файл

Опишем операции, которые можно производить с файлом.

```
#vi mamapapa.txt           - открыть один файл
#vi mamapapa.txt ++11      - открыть файл и перейти к строке 11.
#vi mama.txt papa.txt      - открыть файл mama.txt, после выхода из него открыть файл papa.txt
#vi -o mama.txt papa.txt   - открыть сразу все файлы.
```

Файл открывается в обычном режиме с помощью команды vi. Здесь мы можем просмотреть файл, переместиться по его содержимому, стереть текст, но ввести текст в этом режиме нельзя.

Создание файла происходит при помощи той же команды. Собственно создание файла происходит в момент сохранения.

Для открытия или создания нового файла в обычном режиме набираем

```
:e filename
```

или

```
:new filename
```

Перед этим нужно сохранить предыдущий файл

```
:w            - сохраняет файл с существующим именем
:w filename   - сохраняет файл с новым именем
```

Перемещение по файлу

Самые часто используемые в vi клавиши – клавиши перемещения. Это знаменитые h j k l, соответствующие стрелкам влево, вниз, вверх и вправо. Стрелочки тоже работают (при правильном указании терминала). Можно использовать также быстрые клавиши:

- 0 ("ноль") — в начало текущей строки;
- ^ — в начало текущей строки (к первому непробельному символу);
- $ — в конец текущей строки
- w — на слово вправо
- b — на слово влево
- W — до пробела вправо
- B — до пробела влево

- } — абзац вниз
- { — абзац вверх

- <Ctrl-f> — на страницу (экран) вниз
- <Ctrl-b> — на страницу (экран) верх

- gg — перейти в начало файла
- G — перейти в конец файла
- <number>G — перейти на конкретную строку <number>

- /<text><CR> — перейти к <text>
- ?<text><CR> — то же самое, но искать назад
- n — повторить поиск
- N — повторить поиск назад
- [[— в начало функции
- “ — к месту выполнения команды [[

Редактирование файла

Предположим в нашем файле записан следующий текст:

> – Винни, тебе что намазать на хлеб, мёду или сгущённого молока?
>
> – И того, и другого, и можно без хлеба!
> – Ну, если вы больше ничего не хотите…
>
> – А что, что-то есть ещё?

Диалог симпатичный, но несколько несвязный. Давайте изменим его – для этого необходимо перейти в режим ввода. Самый простой способ – из обычного режима нажатие клавиши «i». После чего можно смело приступать к вводу текста. <ESC> вернет нас к обычному режиму. Другие важные команды:

- R — перейти в режим ввода с забиванием текста
- r — заменить один символ
- i — перейти в режим ввода с текущей позиции
- I — переместиться в начало строки и перейти в режим ввода
- a — перейти в режим ввода после курсора
- A — переместиться в конец строки и перейти в режим ввода
- o — перейти в режим ввода с новой строки под курсором

- O — перейти в режим ввода с новой строки над курсором
- x — удалить символ под курсором
- X — удалить символ влево (удалить символ перед курсором)
- d — используется совместно с командами перемещения. Удаляет символы с текущего положения курсора до положения после ввода команды перемещения. Пример: dw - удаляет символы с текущего до конца слова; diw - удаляет слово под курсором
- c — команда аналогичная d, но после удаление переходит в режим ввода. Очень удобная альтернатива команде R
- dd — удалить текущую строку (вырезать)
- d<число>d — стирание числа строк начиная с текущей
- yy — копирование текущей строки в неименованный буфер
- y<число>y — копирование числа строк начиная с текущей в неименованный буфер
- p — вставка содержимого неименнованного буфера под курсором
- P — вставка содержимого неименнованного буфера перед курсором
- J — слияние текущей строки со следующей
- u — отмена последней команды
- U — отмена всех последних изменений в строке.
- <Ctrl-r> или :redo вперед по изменениям.
- . — повтор последней команды

Конечно, все сразу запомнить их трудно, даже не надо (к тому же здесь перечислены не все). Уверяю вас, через некоторое время эти команды будут пользоваться спросом.

Наконец мы дошли к самому главному – командному режиму. Помним как в него перейти? «:». Шифт-ж если по-русски.

Запись/выход

Если вы в режиме ввода, то необходимо предварительно нажать <ESC>, чтобы перейти в обычный режим.

```
ZQ - выйти без сохранения
```

или

```
:q! - выйти без сохранения
```

```
ZZ - записать файл и выйти(Если файл не изменяли, то записываться он не будет)
```

или

```
:wq - записать файл и выйти
```

или

```
:x - записать файл и выйти
```

```
:w<CR> - записать файл
```

```
:sav filename<CR> - "Сохранить как"
```

или

```
:w filename<CR> - "Сохранить как"
```

```
:w!<CR> - записать файл.
```

Эта команда может помочь, если файл открыт в режиме "read-only" (с опцией -R), если файл заблокирован другим пользователем, либо отсутствуют такие привилегии, либо еще какая беда. При попытке записи без «!» будет выдано соответствующее предупреждение.

```
:w new_file<CR>
```

Создать новый файл «new_file» и записать в него текущее содержимое. Если файл существует, будет показано предупреждение. Дальше вы продолжаете работать со старым файлом – mamapapa.txt

```
:q<CR> - выйти из редактора.
```

Если файл был изменен, у вас ничего не получится. В таких случаях необходимо добавлять после команды «!»:

```
:q!<CR> - выйти из файла, не сохраняя изменения
:qa!<CR> - выйти из всех файлов, не сохраняя изменения
```

Коэффициент повторения

Почти все команды vi принимают так называемый коэффициент повторения. Попросту говоря, если перед командой в обычном режиме вы указываете число, то команда повторяется соответствующее количество раз. Например, команда «j» перемещает курсор на одну строку вниз – «3j» переместит курсор на 3 строки вниз. Результатом команды «iпривет<ESC>», как вы, наверное, догадались, будет вставка слова «привет» – «2iпривет<ESC>» вставит «приветпривет».

Дополнительные возможности

Есть несколько вещей, без которых редактирование похоже на пытку:

```
^G - показать информацию о файле
```

```
:<number> - перейти на строку с номером <number>
<number>G - перейти на строку с номером <number>
:set number - отобразить слева нумерацию строк (:set nonumber - спрятать нумерацию)
:set wrap - переносить длинные строки(:set nowrap - не переносить)
:sy[ntax] on/off - включить/выключить подсветку синтаксиса
:colorscheme <name> - задать цветовую тему (где <name> имя темы, TAB работает как авто-дополнение)
:h или :help - список возможной помощи(:viusage, :exusage)
привести концы строк в файле к виду dos или unix соответственно:
:set fileformat=dos
:set fileformat=unix
Задать размер табуляции в 4 пробела:
```

```
:set tabstop=4
:set expandtab
```

Конвертация кодировки файла:

```
:set fenc=cp1251<CR>
:set fenc=koi8-r<CR>
:set fenc=ibm866<CR>
:set fenc=utf-8<CR>
```

Смена кодировки (возможна до внесения изменений в редактируемый файл. т.е. сразу после открытия)

```
:e ++enc=cp1251<CR>
:e ++enc=koi8-r<CR>
:e ++enc=ibm866<CR>
:e ++enc=utf-8<CR>
```

Vim

Главные отличия от vi

- Неограниченное число undo/redo. (в 7-ой версии появились деревья)
- Многооконность.
- Поддержка множества буферов. (в 7-ой версии появились закладки)
- Подсветка синтаксиса.
- Редактирование командной строкой.
- Достраивание имен файлов.
- Визуальное редактирование.
- Кроссплатформенность.
- Графический интерфейс (GUI).
- Режим совместимости с vi.
- Проверка орфографии

Команды

Переход к началу файла в Vim отличается от Vi:

```
gg
```

Переход к строке под номером <number>:

```
<number>gg
```

Слово под курсором

Перейти к следующему вхождению слова под курсором

```
*
```

```

```

Перейти к предыдущему вхождению слова под курсором

```
#
```

Перейти к определению слова под курсором

```
gd (go definition)
```

Перейти к редактированию файла (под курсором должен быть путь)

```
gf (go file)
```

Визуальный режим

Это существенное расширение возможностей Vi, без которого не обходится ни один «нормальный» текстовой редактор. Режим предназначен для выделения блока текста и дальнейшей работы с ним и состоит из 3-х последовательных шагов:

1. Пометка начала блока с помощью «v», «V» или Ctrl-V. Блок помечается с того символа на котором находится курсор(по умолчанию).
2. Перемещение курсора в конец необходимого блока. Последний символ также будет включен в выделенный блок.
3. Вызов необходимой команды.

Команда «v» предназначена для выделения текста посимвольно. Символы до и после курсора в строке не будут входить в выделение. Необходима в первую очередь для копирования участков текста в строке.
«V» – для выделения текста построчно.
Ctrl-V – для выделения прямоугольного участка текста. Удобна в случае редактирования структурированного текста.

С выделенным участком текста можно производить стандартные действия по редактированию(копирование – «y», удаление – «d»). Напомню, что в Vim при удалении участка текста, он автоматически помещается в специальный регистр, что равноценно «вырезанию» участка текста, который потом можно будет вставить с помощью команды «p»(вставка текста после курсора) или «P»(вставка текста перед курсором).

~/.vimrc

vimrc – это файл конфигурации Vim. При запуске Vim сначала читает этот файл и применяет записанные в нем настройки. Рекомендуемое расположение этого файла для различных платформ:

```
Unix-like         $HOME/.vimrc
MS-DOS and Win32  $HOME/_vimrc или $VIM/_vimrc
```

Файл конфигурации используется для настройки различных аспектов поведения и внешнего вида Vim.

Комментарии в этом файле начинаются с символа " (двойная кавычка) и продолжаются до конца строки.

Для примера приводится выдержка из файла _vimrc автора:

```
" установить keymap, чтобы по Ctrl-^ переключался на русский и обратно
set keymap=russian-jcukenwin
" по умолчанию - латинская раскладка
set iminsert=0
" по умолчанию - латинская раскладка при поиске
set imsearch=0
" игнорировать регистр при поиске
set ic
" подсвечивать поиск
set hls
" использовать инкрементальный поиск
set is
" ширина текста
set textwidth=70
" минимальная высота окна пусть будет 0 (по умолчанию - 1)
set winminheight=0
" всегда делать активное окно максимального размера
set noequalalways
set winheight=9999
" установить шрифт Courier New Cyr
set guifont=courier_new:h10:cRUSSIAN
" настраиваю для работы с русскими словами (чтобы w, b, * понимали
" русские слова)
set iskeyword=@,48-57,_,192-255
" задать размер табуляции в четыре пробела
set ts=4
" отображение выполняемой команды
set showcmd
" перенос по словам, а не по буквам
set linebreak
set dy=lastline
```

Можно добавить привычное поведение и привычные сочетания клавиш.

- Например, можно будет прокручивать (скроллить) текст колесиком мыши и вставлять выделенное в X`ах мышкой в Vim нажатием средней кнопки мыши (нажать на колесико мыши):

```
 set mouse=a
 map <S-Insert> <MiddleMouse>
 map! <S-Insert> <MiddleMouse>
```

- По <F2> сохранение сделанных изменений:

```
imap <F2> <Esc>:w<CR>
map <F2> <Esc>:w<CR>
```

- По <F3> вставка содержимого заданного файла в редактируемый текст. Требуется ввести имя файла. Работает авто-дополнение имени файла по <Tab> в текущем каталоге.

```
imap <F3> <Esc>:read
map <F3> <Esc>:read
```

Также возможен интересный прием работы:

```
:read !ls -la
```

Эта команда вставит в текущую позицию редактируемого текста весь вывод указанной после "!" команды (в нашем случае: ls -la).

- По <F4> открывается новая вкладка (tab) и выводится список каталогов и файлов текущего каталога. Клавишами управления курсором можно выбрать каталог или файл. Нажатие <Enter> на каталог отобразит его содержимое в том же режиме (можно путешествовать по каталогам), а нажатие <Enter> на файле - откроет его в этой же вкладке. Работает быстрый поиск-перемещение по "/".

```
imap <F4> <Esc>:browse tabnew<CR>
map <F4> <Esc>:browse tabnew<CR>
```

- По <F5> позволяет переключать вкладки справа-налево, по-порядку, отображая открытые в них файлы.

```
imap <F5> <Esc> :tabprev <CR>i
map <F5> :tabprev <CR>
```

- По <F6> позволяет переключать вкладки слева-направо, по-порядку, отображая открытые в них файлы.

```
imap <F6> <Esc> :tabnext <CR>i
map <F6> :tabnext <CR>
```

- Пользовательское меню. По <F9> позволяет приостановить работу Vim и вызвать соответствующие программы:

```
set wildmenu
set wcm=<Tab>
menu Exec.GForth  :!gforth % <CR>
menu Exec.Perl    :!perl % <CR>
menu Exec.Python  :!python % <CR>
menu Exec.Ruby    :!ruby % <CR>
menu Exec.bash      :!/bin/bash<CR>
menu Exec.xterm     :!xterm<CR>
menu Exec.mc        :!mc<CR>
menu Exec.xterm_mc  :!xterm -e mc<CR>
map <F9> :emenu Exec.<Tab>
```

  Обратите внимание на, например, конструкцию :!python % <CR> — символ % будет заменён на имя текущего редактируемого файла. В итоге, Vim приостановит работу и вызовет python filename.ext (если вы редактировали filename.ext) в текущем терминале (в том же, где запущен Vim), а затем, после завершения работы python filename.ext, вернётся к редактированию файла. Не забудьте сохранить изменения перед вызовом. Впрочем, Vim должен будет вам напомнить о том, что изменения нужно сохранить, если вы этого не сделали. Заметьте: что Exec. — это не специальная команда, в всего лишь идентификатор меню, объединяющая группу команд и их идентификаторы/названия пунктов меню/.

- Пользовательское меню. По <F10> позволяет вызвать меню различных вариантов завершения работы с Vim. (Глоток воздуха для новичка :)

```
set wildmenu
set wcm=<Tab>
menu Exit.quit     :quit<CR>
menu Exit.quit!    :quit!<CR>
menu Exit.save     :exit<CR>
map <F10> :emenu Exit.<Tab>
```

- Позволяет по <Tab>, более привычному некоторым пользователям, вызывать авто-дополнение для текущего активного синтаксиса:

```
function! InsertTabWrapper(direction)
   let col = col('.') - 1
   if !col || getline('.')[col - 1] !~ '\k'
       return "\<tab>"
   elseif "backward" == a:direction
       return "\<c-p>"
   else
       return "\<c-n>"
   endif
endfunction
inoremap <tab> <c-r>=InsertTabWrapper ("forward")<cr>
inoremap <s-tab> <c-r>=InsertTabWrapper ("backward")<cr>
```

Дополнительно

Для получения более подробной информации по файлу _vimrc можно набрать в командной строке Vim команду

```
:help vimrc
```

Vim и русская раскладка клавиатуры

В обычном режиме Vim по умолчанию ожидает, что консоль находится в режиме ввода латинских символов. Если вы, к примеру, редактируете с помощью Vim текст на русском языке, или в смешанных кодировках (к примеру, HTML-страницу на русском), то постоянная необходимость переключать системную клавиатурную раскладку очень быстро вам надоест. Существует несколько способов решить эту проблему.

Способ 1

Сделать так, чтобы русские буквы можно было вводить, когда системная раскладка находится в режиме ввода латинских символов.

Для этого в файл конфигурации Vim, который называется .vimrc (или _vimrc в Microsoft Windows) нужно добавить следующие строки:

```
set keymap=russian-jcukenwin
set iminsert=0
```

После этого системную раскладку клавиатуры можно будет оставить в режиме ввода латинских символов, а переключение между языками осуществлять уже внутри самого редактора с помощью команды Ctrl-^ (Ctrl-6 в версии для Windows)

Чтобы настроить переключение языков на другую кнопку (например на F12), можно использовать такие команды:

для режима ввода символов:
imap <F12> {Ctrl-k}{Ctrl-6}{Ctrl-6}

При этом {Ctrl-k} нужно вводить нажатием клавиш <Ctrl> и k. Это означает, что сначала нажимается клавиша <Ctrl> и удерживается нажатой, затем нажимается клавиша k и после этого обе клавиши одновременно отпускаются. После нажатия {Ctrl-k} появится подсвеченный значок вопроса, который потом заменится на код клавиши\комбинации, которую Вы введёте, в данном случае {Ctrl-6}{Ctrl-6}, что будет выглядеть как ^^.

> для режима ввода командной строки:
> cmap <F12> {Ctrl-k}{Ctrl-6}{Ctrl-6}

В итоге, визуально в .vimrc это должно выглядеть так:

```
imap <F12> ^^
cmap <F12> ^^
```

Кроме этого, в версии для Windows Vim по умолчанию не настроен на использование кодировок клавиатуры, содержащих символы кириллицы, например CP1251 и KOI8-R (в версии же для Linux настройка по умолчанию корректна). Как следствие этого, Vim при редактировании русского текста будет неправильно обрабатывать команды перемещения по тексту, основанные на поиске слова, такие как **w** (одно слово вперед), **b** (одно слово назад), ***** (найти в тексте слово под курсором) и т. п. Чтобы настроить Vim для корректной работы с этими командами независимо от языка, достаточно добавить в файл конфигурации _vimrc следующую строку:

```
set iskeyword=@,48-57,_,192-255
```

Способ 2

Сделать так, чтобы Vim понимал, какие клавиши нажимаются, если включена русская раскладка.

Для этого в .vimrc добавьте следующую строчку:

Для стандартной раскладки (Windows, Linux):

```
set
langmap=ёйцукенгшщзхъфывапролджэячсмитьбю;`qwertyuiop[]asdfghjkl\;'zxcvbnm\,.,ЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧС
МИТЬБЮ;QWERTYUIOP{}ASDFGHJKL:\"ZXCVBNM<>
```

Для раскладки Mac:

```
set langmap=йцукенгшщзхъфывапролджэячсмитьбю/
ЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ/;qwertyuiop[]asdfghjkl;'zxcvbnm,./QWERTYUIOP[]ASDFGHJKL:'ZXCVBNM,./
```

(Так как в Mac-раскладке символы под цифрами вводят другие символы, соответствующие английским, например, нажатие Shift-4 вводит "%" вместо "$", их не получится переназначить, поэтому способ 1 подходит лучше.)

После этого Vim будет нормально понимать команды, вводимые при включенной русской раскладке.

Если у вас локаль UTF, то данный способ не работает в старых версиях Vim. Необходимо каждый символ назначить индивидуально:

```
  map ё `
  map й q
  map ц w
  map у e
  map к r
  map е t
  map н y
  map г u
  map ш i
  map щ o
  map з p
  map х [
  map ъ ]
  map ф a
  map ы s
  map в d
  map а f
  map п g
  map р h
  map о j
  map л k
  map д l
  map ж ;
  map э '
  map я z
  map ч x
  map с c
  map м v
  map и b
  map т n
  map ь m
  map б ,
  map ю .
  map Ё ~
  map Й Q
  map Ц W
  map У E
  map К R
  map Е T
  map Н Y
  map Г U
  map Ш I
  map Щ O
  map З P
  map Х {
  map Ъ }
  map Ф A
  map Ы S
  map В D
  map А F
  map П G
  map Р H
  map О J
  map Л K
  map Д L
  map Ж :
  map Э "
  map Я Z
  map Ч X
  map С C
  map М V
  map И B
  map Т N
  map Ь M
  map Б <
  map Ю >
```

Возможно это зависит от версии, у меня работает так и причём в UTF:

```
set
langmap=ёйцукенгшщзхъфывапролджэячсмитьбюЁЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ;`qwertyuiop[]asdfghjkl;'zxcvb
nm,.~QWERTYUIOP{}ASDFGHJKL:\"ZXCVBNM<>
```

Так как в последних версиях возможна проблема с парсером символа экранирования '\' – его необходимо записывать как '\\' перед специальными символами ',', ';', '"' и '\'. Потому в версии 7.3 правильная строка может быть такой:

```
set
langmap=ёйцукенгшщзхъфывапролджэячсмитьбюЁЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ;`qwertyuiop[]asdfghjkl\\;'zxc
vbnm\\,.~QWERTYUIOP{}ASDFGHJKL:\\"ZXCVBNM<>
```

Вставка форматированного текста с отступами

У начинающих часто возникает ситуация, когда при копировании форматированного текста с отступами (например, текст страницы из браузера или пример форматированного кода из документации) в vim строки «съезжают» вправо. Это происходит потому, что редактор пытается помочь отформатировать текст «красиво» и копирует отступы с предыдущей строки в новую. Бороться с этим легко:

```
:set paste
```

...вставляем форматированный текст с отступами в режиме редактирования (например, средней кнопкой мыши) и строки «съезжать» вправо не будут. После этого режим «вставки» можно отключить командой:

```
:set nopaste
```

Поиск в файлах с выводом в список

В Microsoft Visual Studio (возможно и в других IDE тоже) есть очень удобная функция Find In Files. Она ищет заданную строку или регулярное выражение в файлах в заданной папке и выводит результат поиска в виде списка. Если щёлкнуть по строке этого списка то откроется файл в том месте где была найдена заданная строка. Подобная функция есть и в Vim. Чтобы найти в файлах нужную строку надо набрать команду:

```
:vimgrep /искомая строка/ Путь к файлам | :copen
```

Более конкретный пример:

```
:vimgrep /translator/ R:\projects\**\*.txt | :copen
```

Команда vimgrep ищет строку или регулярное выражение заключенное между символами // в файле или файлах и сбрасывает результаты в специальный буфер. А так же открывает файл с первым найденным результатом. Команда copen открывает буфер с результатами поиска. Путь к файлам может указывать как на один файл так и на набор файлов. Чтобы путь указывал на несколько файлов надо использовать символы ? и *. Значение этих символов такое же как и в командной строке операционной системы. Чтобы поиск прошел по всем подпапкам указанной папки можно использовать две звездочки **. Пример использования **:

```
/usr/inc**/*.h
```

может означать:

```
/usr/include/types.h
/usr/include/sys/types.h
/usr/inc_old/types.h
```

Как открыть файл в нужной кодировке

Автоматически

При открытии файла Vim попытается определить кодировку, и открыть файл в этой кодировке. Для этого, в ~/.vimrc:

```
set encoding=utf-8                                  " set charset translation encoding
set termencoding=utf-8                              " set terminal encoding
set fileencoding=utf-8                              " set save encoding
set fileencodings=utf8,koi8r,cp1251,cp866,ucs-2le   " список предполагаемых кодировок, в порядке
предпочтения
```

К сожалению Vim не всегда удается определить кодировку файла правильно, и вместо осмысленного текста мы можем наблюдать крокозябры.

Можно указать кодировку текста вручную

Например, можно вручную указать кодировку cp1251 (MS Windows) уже открытому тексту:

```
:e ++enc=cp1251
```

Можно предварительно, перед открытием файла, указать ожидаемую кодировку

Для этого нужно установить переменную encoding (сокращенно enc). Например, установить кодировку ucs-2le (unicode MS Windows):

```
:set enc=ucs-2le
```

После этого можно открывать файл. Он откроется в этой кодировке.

Пользовательское меню

Часто, для упрощения работы в Vim с файлами в различных кодировках, в ~/.vimrc определяют специальное пользовательское меню, которое будет вызываться, например, по <F12>.

```
set wildmenu
set wcm=<Tab>
menu Encoding.koi8-r  :e ++enc=koi8-r<CR>
menu Encoding.cp1251  :e ++enc=cp1251<CR>
menu Encoding.cp866   :e ++enc=cp866<CR>
menu Encoding.ucs-2le :e ++enc=ucs-2le<CR>
menu Encoding.utf-8   :e ++enc=utf-8<CR>
map <F12> :emenu Encoding.<Tab>
```

Перемещаться по пунктам пользовательского меню можно по <Tab> или клавишами управления курсором.

Дополнительно

Чтобы узнать список всех поддерживаемых кодировок, наберите команду:

```
:help encoding-values
```

Конвертация кодировки файла

Vim автоматически пытается распознать текущую кодировку открываемого файла, отображает открываемый файл в кодировке отображения терминала, а при сохранении конвертирует в кодировку сохранения файла.

- Текущая используемая в Vim кодировка задается переменной encoding (enc).
- Кодировка отображения задается переменной termencoding (tenc).
- Кодировка сохранения файла задается переменной fileencoding (fenc).

Кодировка файла encoding определяется при его открытии или берется из настроек системы. При отображении, Vim производит трансляцию текущей кодировки encoding в кодировку отображения termencoding. При сохранении в файл Vim конвертирует из текущей кодировки encoding в кодировку сохранения fileencoding.

Обычно encoding и fileencoding совпадают. Для того, чтобы сохранить файл в другой кодировке, надо перед сохранением задать соответствующее значение переменной fileencoding.

Чтобы узнать список всех доступных значений переменных encoding и fileencoding наберите команду:

```
:help encoding-values
```

Пример: как сконвертировать файл из MS Windows cp1251 в Linux utf-8

- Открываем файл в Vim:

```
vim filename.txt
```

- Задаем текущую кодировку cp1251:

```
:e ++enc=cp1251
```

- Текст конвертируется и Vim отобразит изменения, если они произошли. Исходный файл не изменился.
- Задаем кодировку сохранения файла utf-8:

```
:set fileencoding=utf-8
```

- Задаем формат переводов строки (dos, unix или mac):

```
:set fileformat=unix
```

- Сохраняем файл.

```
:w
```

- Выходим

```
:q
```

Пример: конвертировать файл из Linux utf-8 в MS Windows cp1251

Обратная операция – конвертировать файл из Linux utf-8 в MS Windows cp1251 – будет такой:

```
vim filename.txt
:e ++enc=utf-8
:set fileencoding=cp1251
:set fileformat=dos
:w
:q
```

Vim при переводах текста, или для чтения английской документации

Итак, чтобы можно было успешно читать и переводить документацию на английском языке, нужно ИМХО всего 3 вещи:

- Редактор.
- Переводчик.
- Проверка орфографии

Если их удобно объединить, то можно получить хорошую среду, для перевода/чтения.

Что возьмем

Редактор мы естественно возьмем VIM хотя бы потому что статья так называется. Переводчик mueller консольная версия, работает так:

```
% mueller sex
sex  [seks] _n. 1> _биол. пол; the weaker sex слАбый пол, жЕнщины; the sterner (или stronger) sex
сИльный пол, мужчИны; the sex _шутл.  жЕнщины 2> секс 3> _at
tr. половОй, сексуАльный; sex instinct половОй инстИнкт; sex intergrade гермафродИт
```

И орфографмометр: это spell в виме, словари скачаем с vim.org

Как настроим

Мне интересна возможность выпадающего списка, при наведением на неизвестное мне слово мышкой. Для этого, после долгого чтения манов (или гугленья) узнаем что с помошью команды system можно выполнить какое-то действие, вывод которой куда-либо записать. Короче, пишем где-нибудь в ~/.vimrc

```
function! MyBalloonExpr()
       return system("mueller ". v:beval_text)
endfunction
set bexpr=MyBalloonExpr()
set balooneval
set balloondelay=100
```

Определили функцию MyBallonExpr которая будет формировать строку mueller [слово на которое навели] и выполнять. Результат выполнения будет выдавать в «балоне» выпадающем окошке, через 100 миллисекунд.

Естественно вместо mueller вы можете поставить любую другую программу, которая переводит текст.

Рецепт для оч.умелых ручек

Если у вас нет консольной версии mueller, но есть много энтузиазма, то можно поступить так:

- Идем на http://sourceforge.net/projects/mueller-dict/ и забираем оттуда mueller-dict-3.1.1.tar.gz
- Извлекаем из полученного архива файл mueller-base.dict.dz и кладем его, скажем, в ~/dict/
- Там же создаем файл mueller-base.sh со следующим содержимым:

```
#!/bin/bash
#
WORD=`echo $1 | sed -e 's/[A-Z]*/\L&/g;s/[,.;:]//g'`;
if [ -z $WORD ]; then exit; fi;
LINES=50;
DICTIONARY="$HOME/dict/mueller-base.dict.dz";
#
zgrep -aA$LINES '^'$WORD'$' $DICTIONARY | sed -e '/^$/d; 1s/^/ &/;/^[^ ]/,$d'
#
```

- даем нашему новому консольному словарику право на исполнение:

```
$ chmod +x ~/dict/mueller-base.sh
```

- и проверяем его работоспособность:

```
$ ~/dict/mueller-base.sh head
```

- Для графической версии редактора (gvim) функция перевода в нашем ~/.vimrc должна будет выглядеть так:

```
function! MyBalloonExpr()
   return system("$HOME/dict/mueller-base.sh ". v:beval_text)
endfunction
set bexpr=MyBalloonExpr()
```

```
set balooneval
set balloondelay=500
```

Примечание:

- Естественно, что функция MyBalloonExpr() работает только в gVim, зато выглядит красиво. Наведите курсор мышки на слово и оставьте его неподвижным секунду-две (поначалу, можно вообще убирать руку с мышки, позже приловчитесь).
- LINES=50 ограничивает количество выводимых строчек. Можете подобрать цифру под свой монитор.
- Рекомендую выставить balloondelay=500 миллисекунд. Да, помедленнее. Но мышка будет меньше цепляться за слова.

- Для текстовой версии редактора (vim) придётся делать специальную функцию TranslateWord() и вешать её вызов на специальную функциональную клавишу, например <F9>:

```
function! TranslateWord()
   let s:dict    = "$HOME/dict/mueller-base.sh"
   let s:phrase  = expand("<cword>")
   let s:tmpfile = tempname()
   silent execute "!" . s:dict . " " . s:phrase . " > " . s:tmpfile
   execute "botright sp " . s:tmpfile
endfunction
map <F9> :call TranslateWord()<CR>
```

Примечание:

- Это работает точно так же и в графической версии редактора (gvim). Поэтому метод можно считать универсальным, но требует нескольких нажатий клавиш, и сплит-окно выглядит менее элегантно, чем подсказки.
- Открывшееся окно с переводом можно закрыть стандартным способом (:q). Ну, а если вы воспользовались советом выше и забиндили на клавишу <F10> предложенное пользовательское меню, то алгоритм работы будет совсем простой: <F9> смотрим перевод <F10><Enter> вернули как было.
- Этот метод использует временный файл
- Т.к. в этом методе нет необходимости в ограничении, чтоб подсказка помещалась на экран, то для LINES можно увеличить лимиты, хоть до LINES=500. Этот метод не имеет ограничений на количество выводимых строчек.

Строка статуса редактора

Строка статуса редактора бывает очень полезна, если вам не жалко пары строчек на экране. Добавьте в ~/.vimrc:

```
set laststatus=2   " всегда показывать строку статуса
set statusline=%f%m%r%h%w\ %y\ enc:%{&enc}\ ff:%{&ff}\ fenc:%{&fenc}%=(ch:%3b\ hex:%2B)\ col:%2c\
line:%2l/%L\ [%2p%%]
```

...и у вас в предпоследней строке будет выводиться что-то вроде этого:

```
.vimrc [vim] enc:utf-8 ff:unix fenc:utf-8       (ch: 92 hex:5C) col:26 line:158/168 [94%]
```

Комментарий:

%f - имя файла и путь к нему, относительно текущего каталога
%m - флаг модификации/изменения, выводит [+] если буфер изменялся
%r - флаг "только для чтения", выводит [RO] если буфер только для чтения
%h - флаг буфера помощи, выводит [help] если буфер со справкой vim
%w - флаг окна превью, выводит [Preview]
'\ ' - экранированный символ пробела. Пробел можно указывать только экранированным, иначе ошибка синтаксиса
%y - тип файла в буфере, например [vim]
enc:%{&enc} - отображение кодировки encoding (enc). Обратите внимание: "enc:" - текст, "%{&enc}" - вывод значения внутренней переменной (enc)
ff:%{&ff} - отображение формата перевода строки fileformat (ff)
fenc:%{&fenc} - отображение кодировки сохранения в файл fileencoding (fenc)
%= - далее выравнивать вправо
ch:%3b - код символа под курсором в десятичной чистеме счисления, минимум 3 символа
hex:%2B - код символа под курсором в шестнадцатеричной системе счисления, минимум 2 символа
col:%2c - курсор в колонке, минимум 2 символа
line:%2l/%L - курсор в строке (минимум 2 символа)/всего строк в файле
%2p - число % в строках, где находится курсор (0% - начало файла; 100% - конец файла), минимум 2 символа
%% - т.к. символ '%' используется для переменных, то вывод символа '%' в строке статуса нужно делать особым образом - %%

Подробнее о формате вывода statusline можно узнать из справки:

```
:help statusline
```

Проверка орфографии

Пользовательское меню

Часто бывает удобно оформить процедуры, связанные с проверкой орфографии, в виде пользовательского меню:

```
set wildmenu
set wcm=<Tab>
" проверка орфографии:
menu SetSpell.ru  :set spl=ru spell<CR>
menu SetSpell.en  :set spl=en spell<CR>
menu SetSpell.off :set nospell<CR>
map <F7> :emenu SetSpell.<Tab>
" выбор альтернатив:
imap <F8> <Esc> z=<CR>i
map <F8> z=<CR>
```

Здесь, по <F7> вызывается подменю: проверить орфографию русского языка [ru], проверить орфографию английского языка [en] или отключить проверку орфографии [off].

По <F8> для слова под курсором можно найти корректную альтернативу из словаря. Порядок работы следующий:

- Наводим курсор на выбранное слово (или подсвеченное проверкой орфографии слово)
- Нажимаем <F8> и видим список альтернатив. Каждый вариант в списке помечен цифрой.
- Находим понравившийся вариант и запоминаем его номер.
- Смело начинаем набирать запомненный номер, неважно, одна это цифра или две.
- Снова нажимаем <F8>.
- Слово под курсором будет заменено альтернативой с указанным номером.

Последовательность: <F8><одна-или-две-цифры><F8>

Как это работает?

Попробуйте на примере слова "Шпоргалка".

- Откройте Vim, перейдите в режим редактирования и наберите слово "Шпоргалка".
- Выйдите из режима редактирования в командный режим (обычно, <Esc><Esc>)
- Установите курсор на слово "Шпоргалка"
- Нажмите последовательно два символа "z" и "=" (два символа: z=)
- Как результат, вы должны увидеть экран с альтернативами, подобранными для слова под курсором. Нажатие любой клавиши скроет этот экран, но безопасными и не оставляющими следов в буфере клавиатуры будут либо <Esc><Esc>, либо <Enter>.
- Запоминаем, что хорошей альтернативой слову "Шпоргалка" было слово "Шпаргалка", под номером "1".
- А теперь магия! Нажимаем последовательно три символа: "1", "z" и "=" (три символа: 1z=)
- Слово под курсором "Шпоргалка" заменено на "Шпаргалка".
- С использованием пользовательского меню, операция будет выглядеть, как три нажатия на кнопки: <F8>1<F8>.

Теперь мы с вами знаем, что если навести на слово "Шпоргалка" курсор и нажать три магических символа 1z= ,то ошибочное слово будет волшебным образом заменено на правильное ("Шпаргалка"). А если нажать четыре магических символа 39z= ,то слово "Шпоргалка" под курсором будет волшебным образом заменено, скорее всего, на слово "Прогулка" (это если у вас стоят словари той же версии, что и у меня).

А теперь, самая главная тайна:

```
для того, чтоб все вышеуказанное работало,
проверка орфографии русского языка должна быть включена.
```

На всякий случай, напоминаю, что выход без сохранения изменений:

```
:q!<Enter>
```

Выход, с сохранением всех изменений:

```
:x<Enter>
```

Ссылки

- Vim.org (http://www.vim.org) — официальный сайт vim — документация, файлы, плагины
- Vim-howto (http://www.opennet.ru/docs/HOWTO-RU/Vim-HOWTO.html) — руководство по использованию vim
- Vim. Первая установка (http://jenyay.net/Programming/Vim)
- Очередной HowTo по Vim (http://konishchevdmitry.blogspot.com/2008/07/howto-vim.html) — HowTo, описывающий основные проблемы, возникающие при работе в Vim. Содержит примеры конфигурационных файлов и список наиболее полезных команд.
