Vim Color Editor HOW-TO (Vi Улучшенный с цветной подсветкой синтаксиса)

Al Dev (Alavoor Vasudevan) *alavoor@yahoo.com*
Russian Linux Documentation Project
v4.0, 31 Декабря 1999

Этот документ является руководством по быстрой настройке редактора vim в Linux или другом Unix. Информация, приведенная здесь ориентирована на программистов, так как vim поддерживает синтаксическую подсветку и утолщение шрифтов для повышения "читаемости" текстов программ. Информация в этом документе справедлива для всех операционных систем, в которых работает vim - Windows 95/NT и все разновидности Unix, типа Linux, Solaris, HPUX, AIX, SCO, Sinix, BSD, SCO, итд..

Содержание

1 Введение

1.1 О "Russian Linux Documentation Project"(RLDP)

Данный документ был переведен в рамках проекта - Russian Linux Documentation Project.
Цель данного проекта - перевод документации, посвященной различным аспектам настройки и использования ОС Линукс, на русский язык. Более подробно узнать о проекте вы можете на нашем Веб Сайте:
www.linux.ru.net/RLDP
В переводе данного документа участвовали:

- Александр Михайлов *alexmikh@linux.ru.net*

1.2 Что такое Vim ?

Vim это сокращение от 'Vi Improved'(улучшеный Vi).Vi - это один из самых популярных и мощных редакторов в мире Unix. **Vi** в свою очередь это сокращение от "***Vi***sual"editor (Визуальный редактор). В старые дни,первым редактором на планеты был line editor (линейный редактор), называемый 'ed' (и 'ex'). ***Ви***зуальный редактор такой как Vi был большим шагом вперед по сравнению с ним. Кстати 'ed' и 'ex' все еще доступны в Линукс.(см. 'man ed' и 'man ex')
Хороший редактор должен повышать производительность труда программиста. Vim поддерживает цветную подсветку синтаксиса программного кода и различные шрифты такие как normal,bold или italics. Редактор с цветной подсветкой (как Vim) повысит **производительность** труда программиста **в 2-3 раза**!!! Программист сможет читать код быстрее, т.к. синтаксис языка выделяется специальными средствами.

1.3 Установка Vim для RedHat Linux

Для использования Vim, вам нужно установить следующие RPM пакеты -

```
        rpm -i vim*.rpm
Или так -
        rpm -i vim-enhanced*.rpm
        rpm -i vim-X11*.rpm
        rpm -i vim-common*.rpm
        rpm -i vim-minimal*.rpm
```

Вы можете вывести на экран список файлов, которые вы установили такой командой:

```
        rpm -qa | grep ^vim | awk '{print "rpm -ql " $1 }' | /bin/sh | less
```

и просматривать его используя j,k, CTRL+f,CTRL+D,CTRL+B,CTRL+U или клавиши со стрелками + клавиши page up/down. (см. 'man less').

1.4 Установка Vim для GNU Debian Linux

Чтобы установить Vim для Debian Linux (GNU Linux),войдите в систему как root и (соединившись с интернет) введите -

```
apt-get install vim vim-rt
```

Эта команда загрузит последнюю версию vim,установит её,сконфигурирует,и удалит файл .deb который был загружен.Первый пакет в списке - это vim,стандартный редактор,собранный с поддержкой X11,а vim-rt это vim runtime,который содержит файлы синтаксиса и файлы помощи.

1.5 Установка Vim для Unix'ов

Для других Unix'ов, таких как Solaris, HPUX, AIX,Sinix,SCO,загрузите пакет с исходным кодом.

```
        zcat vim.tar.gz | tar -xvf -
        cd vim-5.5/src
        ./configure --enable-gui=motif
        make
        make install
```

1.6 Установка Vim для Microsoft Windows 95/NT

Для Windows 95/NT, загрузите zip архив, и начните установку, щелкнув на setup. Взять этот zip файл можно на -

- Домашней страничке vim <http://www.vim.org>
- На её зеркале в США <http://www.us.vim.org>

2 Настройка файлов инициализации gvim

Чтобы разрешить подсветку синтаксиса, вы должны скопировать файл vimrc в ваш домашний каталог. Это сделает возможным вызов меню "Syntax"внутри gvim. В этом меню вы сможете выбрать языки типа C++, Perl, Java, SQL, ESQL и т.д.

```
cd $HOME
cp /usr/doc/vim-common-5.3/gvimrc_example  ~/.gvimrc
```

Комментарии в .gvimrc начинаются с двойных кавычек ("). Вы можете настроить vim по своему вкусу отредактировав файл $HOME/.gvimrc и вставив следующие строки -

```
"Эта строка является комментарием ,который начинается с двойных кавычек
" Лучший шрифт это bold, но если он вам не понравиться вы можете выбрать другой по своему в
set guifont=8x13bold
"set guifont=9x15bold
"set guifont=7x14bold
"set guifont=7x13bold
"
" Рекомендуется устанавливать количество пробелов в TAB равное четырем
set tabstop=4
set shiftwidth=4
"
" Противоположная опция - это 'set wrapscan', полезна при поиске строк
set nowrapscan
"
" Противоположная опция - noignorecase
set ignorecase
```

Очень рекомендуется устанавливать *tabstop* равный 4 и *shiftwidth* равный 4. *tabstop* - это количество пробелов,которое будет вставлено в текст при нажатии TAB.
shiftwidth - то количество пробелов на которое будет сдвинута строка командами ">>"или "<<"(Нажмите соответствующий символ на клавиатуре два раза). Более подробно об этих и других командах можно почитать в пособии 8 ().

2.1 Пример конфигурационного файла gvimrc

Вы можете изменить такие параметры как цвет и тип шрифта в вашем $HOME/.gvimrc файле. Рекомендуется устанавливать *ярко-желтый*или*белый* цвет фона, и черный цвет для шрифтов. Т.к. доказано,что именно такая гамма наиболее удобна для глаз. Поэтому измените переменную 'guibg' в вашем $HOME/.gvimrc файле следующим образом:

```
        highlight Normal guibg=lightyellow
```

А это пример конфигурационного файла взятый из /usr/doc/vim-common-5.3/gvimrc_example:

```
" Vim
" Пример файла gvimrc.
" Эти команды исполняются при запуске GUI.
"
" Чтобы использовать, скопируйте этот файл
"     для Unix и OS/2:  ~/.gvimrc
"             для Amiga:  s:.gvimrc
"  для MS-DOS и Win32:  $VIM\_gvimrc

" Заставит внешние команды работать через pipe вместо pseudo-tty
"set noguipty

" Устанавливает используемый X11 шрифт
" set guifont=-misc-fixed-medium-r-normal--14-130-75-75-c-70-iso8859-1

" Делает командную строку двойной в высоту
set ch=2

" Заставляет комбинацию shift-insert работать как в Xterm
map <S-Insert> <MiddleMouse>
map! <S-Insert> <MiddleMouse>

" следующие команды исполняются только с vim версии 5.00 и старше.
if version >= 500

  " Я люблю когда строки внутри C комментариев подсвечены
  let c_comment_strings=1

  " Включает подсветку синтаксиса.
  syntax on

  " Включает подсветку выражения которое вы ищете в тексте.
  set hlsearch

  "Для Win32 версии по нажатию "K" начинается поиск в help файле

  "if has("win32")
  "  let winhelpfile='windows.hlp'
  "  map K :execute "!start winhlp32 -k <cword> " . winhelpfile <CR>
  "endif

  "Прячет курсор мыши во время набора текста
  set mousehide

  " Устанавливает приятные цвета
  " Фон для обычного текста ярко серый
  " Текст  ниже последней строки темно серый
  " Курсор зеленый
  " Константы не подчеркиваются но имеют чуть более яркий фон
  highlight Normal guibg=grey90
  highlight Cursor guibg=Green guifg=NONE
  highlight NonText guibg=grey80
  highlight Constant gui=NONE guibg=grey95
```

```
  highlight Special gui=NONE guibg=grey95

endif
```

Смотрите также vimrc используемый для консольного режима vim : /usr/doc/vim-common-5.3/vimrc_example.

2.2 Xdefaults параметры

Вы можете установить некоторые параметры Vim в файле Xdefaults.
Внимание!! *Не используйте* ***Vim*geometry*** *это испортит меню gvim, вместо этого используйте* ***Vim.geometry****.*
Отредактируйте $HOME/.Xdefaults и поместите туда следующие строки:

```
! Эргономичные цвета для GVim.
Vim*useSchemes:         all
Vim*sgiMode:            true
Vim*useEnhancedFSB:     true
Vim.foreground:         Black
!Vim.background:        lightyellow2
Vim*background:         white
! Не используйте Vim*geometry , это испортит меню вместо этого используйте
! Vim.geometry. Астерик между Vim и geometry использовать нельзя.
! Vim.geometry: widthxheight
Vim.geometry:           88x40
!Vim*font:              -cronyx-fixed-medium-r-normal-*-20-200-75-75-c-100-koi8-*
Vim*menuBackground: yellow
Vim*menuForeground: black
```

Чтобы изменения вошли в силу выполните следующие команды -

```
        xrdb -merge $HOME/.Xdefaults
        man xrdb
```

Вы также можете отредактировать файл ~/.gvimrc, чтобы изменить цвета

```
        gvim $HOME/.gvimrc
Лучший цвет для фона ярко-желтый (lightyellow) или белый (white) с черным шрифтом.
        highlight Normal guibg=lightyellow
```

3 Инициализационные файлы световой подсветки

Вместо использования меню "Syntax", вы можете отредактировать файл с настройками вручную. При редактировании файла в gvim примените команду : (двоеточие) so. Например:

```
        gvim foo.pc
        :so $VIM/syntax/esqlc.vim
```

Файлы синтаксиса находятся в /usr/share/vim/syntax/*.vim. Vim поддерживает более 120 различных файлов синтаксиса для различных языков программирования, таких как C++,PERL,VHDL,Javascript,... и так далее!!
Каждый файл синтаксиса поддерживает стандартное расширение имени файла, например файл синтаксиса для JavaScript поддерживает расширение *.js. Если вы используете другие расширения,

конфликтующие с файлом синтаксиса по умолчанию (например помещаете Javascript в html файл). То вы должны использовать команду подобную этой: so $VIM/syntax/javascript.vim. Лучше всего создать символическую ссылку -

```
        ln -s $VIM/syntax/javascript.vim js
        gvim foo.html  (... этот файл содержит javascript функции и HTML)
        :so js
```

4 Использование VIM

Вы можете использовать VIM в двух режимах - один с GUI и другой без GUI. Чтобы использовать GUI,используйте команду -

```
        gvim foo.cpp
```

Чтобы использовать не-gui режим -

```
        vim foo.cpp
или более ортодоксальную форму
        vi foo.cpp
```

Если у вас есть такая возможность, всегда используйте GUI режим, т.к. он поддерживает множество цветов и сильно повысит вашу производительность.
GUI режим позволяет делать следующее -

- Вы можете выделять текст с помощью мышки, чтобы копировать, вырезать и вставлять его.
- Вы можете использовать полосу меню с следующими разделами - File, Edit, Window, Tools, Syntax и Help.
- Также, в ближайшем будущем, Vim будет включать вторую полосу меню с перечислением открытых файлов, и вы сможете переключаться между файлами щелкая на их именах,а пока вы можете использовать команды vi - :e#, :e#1, :e#2, :e#3, :e#4, ...и так далее, чтобы выбирать файлы.

5 Компаньоны Vi

Обычно Vim используется совместно с другими мощными инструментами такими как **ctags** и **gdb**. **Ctags** для быстрой навигации сквозь миллионы строк "C/C++"кода и **gdb** для отладки "C/C++"кода. Эта глава посвящена именно этим двум незаменимым инструментам.
Ctags - это самая мощная и нужная команда для написания программ на C, C++, Java, Perl, Korn/Bourne shell scripts или Fortran. Разработчики широко используют **ctags** для навигации в тысячах функций программ на C/C++. (Смотрите 'man ctags'). Очень **важно**, чтобы вы научились использовать ctags. Это позволит вам эффективно писать программы на C/C++,Java и т.д. Навигации в море кода - одна из самых важных задач во время разработки программ на C/C++. Использование ctags позволит вам решать эту задачу быстро и эффективно. Ctags - позволяет очень быстро читать код, просматривать функции, перескакивая от одной функции к другой.
Без такой навигации, вы скорее всего заблудитесь в коде ! **Ctags** - компас для программиста.
Использование **ctags** :

```
        ctags *.cpp
        gvim -t foo_function
        gvim -t main
```

Эти команда откроет программу на C++, которая содержит функцию foo_function() и разместит курсор на первой строке функции foo_function(). Вторая команда, отправит вас на строку в которой определена функция main().
Внутри Vim вы можете перейти к функции введя : (двоеточие) tag < имя функции >как показано ниже:

```
:tag sample_function
```

Эта команда разместит курсор на первой строке функции sample_function()
Если вы захотите перейти прямо к функции от строки в файле, которая содержит имя этой функции, поместите курсор перед именем функции и нажмите **CTRL+]**, это отправит вас прямо на строку где начинается искомая функция.

```
// example code
switch(id_number) {
        Case 1:
                if ( foo_function( 22, "abcef") == 3 )
                    ^
                    |
                    |
                    |
  Разместите курсор здесь (перед foo_function) и нажмите CTRL+]
  Это отправит вас на начало "foo_function". Чтобы вернуться назад нажмите
```

Чтобы отправиться назад, на исходную строку, нажмите **CTRL+t**. Вы можете нажимать **CTRL+t** несколько раз чтобы вернуться в точку где вы начали навигацию. Таким образом можно "пройти"скозь все функции C/C++ кода.

5.1 Ctags для ESQL

T.к. ctags не поддерживают языка Embedded SQL/C (ESQL), то ,для создания тэгов для этого языка, необходим приведенный ниже shell script. ESQL/C - это команды SQL, для управления базой данных,встроенные в C программу. Oracle ESQL/C называется Pro*C и Sybase;для Infomix есть ESQL/C;для PostgreSQL есть "ecpg".
Сохраните этот файл как "sqltags.sh"и выполните chmod a+rx tags_gen.sh.

```
#!/bin/sh

# Программа для создания ctags для ESQL, C++ и C файлов
ESQL_EXTN=pc
tag_file1=tags_file.1
tag_file2=tags_file.2

which_tag=ctags

rm -f $tag_file1 $tag_file2 tags

aa=`ls *.$ESQL_EXTN`
#echo $aa
for ii in $aa
do
        #echo $ii
```

```
        jj=`echo $ii | cut -d'.' -f1`
        #echo $jj

        if [ ! -f $jj.cpp ]; then
                echo " "
                echo " "
                echo "***********************************************"
                echo "ESQL *.cpp не существуют.. "
                echo "Вы должны сгенерировать *.cpp из файла *.pc "
                echo "используя Oracle Pro*C пре-компилятор или Sybase"
                echo "либо Informix esql/c пре-компилятор."
                echo "А затем перезапустить эту программу"
                echo "***********************************************"
                echo " "
                exit
        fi

        rm -f tags
        $which_tag $jj.cpp
        kk=s/$jj\.cpp/$jj\.pc/g

        #echo $kk > sed.tmp
        #sed -f sed.tmp tags >> $tag_file1

        #sed -e's/sample\.cpp/sample\.pc/g' tags >> $tag_file1
        sed -e $kk tags >> $tag_file1
done

# Теперь обработаем все C++/C файлы, исключая ESQL *.cpp файлы
rm -f tags $tag_file2
bb=`ls *.cpp *.c`
aa=`ls *.$ESQL_EXTN`
for mm in $bb
do
        ee=`echo $mm | cut -d'.' -f1`
        file_type="NOT_ESQL"
        # Exclude the ESQL *.cpp and *.c files
        for nn in $aa
        do
                dd=`echo $nn | cut -d'.' -f1`
                if [ "$dd" = "$ee" ]; then
                        file_type="ESQL"
                        break
                fi
        done

        if [ "$file_type" = "ESQL" ]; then
                continue
        fi

        rm -f tags
        $which_tag $mm
        cat tags >> $tag_file2
```

```
done

mv -f $tag_file2 tags
cat  $tag_file1 >> tags
rm -f $tag_file1

# Чтобы все работало правильно необходимо отсортировать файл с тэгами ....
sort tags > $tag_file1
mv $tag_file1 tags
```

5.2 Ctags для Javascript программ, оболочек Korn и Bourne

Этот shell скрипт может быть использован для генерации тэгов для очень большого количества программ написанных на JavaScript,PHP/FI скрипты,Korn shell,C shell,Bourne shell и многих других. Это очень "стандартный"модуль.
Сохраните этот файл как tags_gen.sh и выполните chmod a+rx tags_gen.sh.

```
#!/bin/sh

tmp_tag=tags_file
tmp_tag2=tags_file2

echo " "
echo " "
echo " "
echo " "
echo " "
echo "Генерируем тэги для ...."
while :
do
        echo "    Введите расширение файлов, для которых вы хотите сгенерировать тэги."
        echo -n "    Расширения файлов должны быть заданы как sh, js, ksh, etc... : "
        read ans

        if [ "$ans" == "" ]; then
                echo " "
                echo "Неверный ввод. Попробуйте еще раз!"
        else
                break
        fi
done

\rm -f $tmp_tag

aa=`ls *.$ans`

for ii in $aa
do
        jj=`echo $ii | cut -d'.' -f1`
        #echo $jj
        cp $ii $jj.c
        ctags $jj.c
        echo "s/$jj.c/$ii/g" > $tmp_tag2
```

```
        sed -f $tmp_tag2 tags >> $tmp_tag
        \rm -f tags $jj.c
done

sort $tmp_tag > tags

\rm -f $tmp_tag $tmp_tag2
```

5.3 Дебаггер gdb

Вы почти наверняка захотите использовать gdb (смотрите 'man gdb') вместе с Vim. Отладка - один из наиболее сложных и трудоемких процессов в программировании.
Чтобы использовать gdb, вы должны компилировать программу с опцией -g3. Например:

```
gcc -g3 foo.c foo_another.c sample.c
```

Чтобы легко настроить псевдонимы, сделайте следующее -

```
Настройте псевдоним в вашем:
~/.bash_profile
        alias gdb='gdb -directory=/home/src -directory=/usr/myname/src '
  Выполните -
        gdb foo.cpp
        gdb> dir /hom2/another_src
        Это добавит некоторые файлы к пути поиска
        gdb> break 'some_class::func<TAB><TAB>
  Это позволит дополнять имена функции -
        gdb> break 'some_class::function_foo_some_where(int aa, float bb)'
```

Двойное нажатие TAB приведет к автоматическому завершению командной строки, что сохранит вам массу времени.
Чтобы получить помощь во время работы -

```
        gdb> help
  Выдает  на экран помощь
        gdb> help breakpoints
  Выдает более детальную помощь о breakpoints.
```

Для установки breakpoints и входа в отладку

```
        unixprompt> gdb exe_filename
        gdb> b main
  Это поместит breakpoint на функции main()
        gdb> b 123
  Это поместит breakpoint на 123й строке текущего файла
```

Чтобы проанализовать core используйте -

```
        unixprompt> gdb exe_filename  core
        gdb> bt
  Начинает трассировку тех строк программы где произошла ошибка
        gdb> help backtrace
  Выдает более детальную помощь о режиме backtrace.
```

Вы также можете использовать GUI версию gdb называемую xxgdb.
Инструменты для отслеживания "утечек"памяти -

- Бесплатный Electric Fence доступный почти в любом дистрибутиве,
- Коммерческий Purify <http://www.rational.com>
- Insure++ <http://www.insure.com>

6 Как получить помощь по vim

Просмотрите man страницы по vim. Для этого введите **'man vim'** и **'man gvim'**
Или внутри gvim дайте команду :help, чтобы просмотреть страницу помощи. Смотрите также 8 ()

```
                        VIM - главный файл помощи

          Перемещение:  Используйте клавиши со стрелками, или "h" чтобы переместиться влево,
                        "j" вниз, "k" вверх, "l" вправо.
                        ":1" переход на первую строку страницы
                        ":n" переход на строку номер n
                        "<SHIFT>g" переход на край страницы
                        ":/someword/ найдет "someword" в документе

       Закрыть это окно:  Используйте ":q<Enter>".

        Перейти к секции:  Разместите курсор на тэге  между | | и нажмите CTRL-].

          С помощью мышки:  ":set mouse=a" чтобы включить мышку  (в xterm или GUI).
                            Двойной щелчок на тэге между ||.

          Вернуться назад:  Введите CTRL-T или CTRL-O.

      Специфичная помощь:  Можно отправиться прямо к секции,которая вас интересует,
                           указав в команде ":help" аргумент.
                           Также возможно более конкретно задать что вам необходимо:

                            Что                        База       Пример      ~
                    Команды нормального режима         (нет)     :help x
                    Команды визуального режима          v_       :help v_u
                    Команды режима вставки              i_       :help i_<Esc>
                    Команда командной строки            :        :help :quit
                    Редактирование командной строки     c_       :help c_<Del>
                    Аргументы команд vi                 -        :help -r
                             опции                      '        :help 'textwidth'

      Список файлов документации:

      |howto.txt|     как делать большинство простых вещей
      |intro.txt|     введение в Vim
      |index.txt|     алфавитный индекс для каждого режима
      |autocmd.txt|   автоматическое выполнение команд при каких либо событиях
      |change.txt|    удаление и замещение текста
```

7 Домашняя страница Vim и полезные ссылки

Домашняя страница Vim <http://www.vim.org>
Зеркало в США <http://www.us.vim.org>
Vim FAQ <http://www.grafnetix.com/~laurent/vim/faq.html> и на <http://www.vim.org/faq>
Страница "Eli's Vim Page" <http://www.netusa.net/~eli/src/vim.html>
Страница "Vi Lovers Home Page" <http://www.cs.vu.nl/~tmgil/vi.html>
Справочное пособие по Vim <http://scisun.sci.ccny.cuny.edu/~olrcc/vim/>
Список рассылки Vim <http://www.findmail.com/listsaver/vimannounce.html> и <http://www.vim.org/mail.html>
Архивы списков рассылки:

- <http://www.egroups.com/group/vim>
- <http://www.egroups.com/group/vimdev>
- <http://www.egroups.com/group/vimannounce>

Макросы Vim <http://www.grafnetix.com/~laurent/vim/macros.html>

8 Пособие по Vim

8.1 Vim Пособие

На Линукс системе просмотрите пособие /usr/doc/vim-common-5.*/tutor,на других unix системах отправьтесь в директорию где установлен vim и поищите директорию doc.

```
cd /usr/doc/vim-common*/tutor
less README.txt
cp tutor $HOME
cd $HOME
less tutor
```

8.2 Vi Пособия в Internet

- Purdue University <http://ecn.www.ecn.purdue.edu/ECN/Documents/VI/>
- Quick Vi tutorial <http://linuxwww.db.erau.edu/LUG/node165.html>
- Advanced Vi tutorial <http://www.yggdrasil.com/bible/bible-src/user-alpha-4/guide/node171.html>
- Tutorials <http://www.cfm.brown.edu/Unixhelp/vi_.html>
- Tutorials <http://www.linuxbox.com/~taylor/4ltrwrd/section3_4.html>
- Unix world online vi tutorial <http://www.networkcomputing.com/unixworld/unixhome.html>
- Univ of Hawaii tutorial <http://www.eng.hawaii.edu/Tutor/vi.html>
- InfoBound <http://www.infobound.com/vi.html>
- Cornell Univ <http://www.tc.cornell.edu/Edu/Tutor/Basics/vi/>
- Vi Lovers home page: <http://www.cs.vu.nl/~tmgil/vi.html>
- Ater Sept 2000, will moveto <http://www.thomer.com/thomer/vi/vi.html>
- Beginner's Guide to vi <http://www.cs.umr.edu/unixinfo/general/packages/viguide.html>
- vi Help file <http://www.vmunix.com/~gabor/vi.html>
- vim FAQ <http://www.math.fu-berlin.de/~guckes/vim/faq/>

В интернет существует огромное число пособий по Vi. Поищите их, используя какую нибудь из поисковых машин.

9 Руководство по Vi

В этом пособии,описываются некоторые "продвинутые"команды и концепции **vi**, так, чтобы вы могли оценить всю мощь **vi** и продумать какие команды вам необходимо изучить в первую очередь. Почти все руководства по **vi** перечисляют доступные команды, но не показывают их взаимосвязи;это и есть основное предназначение данного руководства.

9.1 Команды перемещения курсора

Команды **vi** для перемещения курсора,позволяют позиционировать курсор в файле и/или на экране эффективно,с минимальным количеством нажатий клавиш. Существует огромное количество команд перемещения курсора - не пытайтесь запомнить их все сразу ! Позже вы увидите,что мощь **vi** заключается в комбинации команд передвижения курсора с командами для удаления,изменения,копирования и фильтрования текста.
Пожалуйста выберите какой нибудь большой текст,чтобы вы смогли проэкспериментировать с каждой из описанных команд. Также помните,что эти команды будут работать только в командном режиме,не в режиме вставки;если вы начали вводить эти команды прямо в текст - нажмите ESC чтобы вернуться в командный режим.

- **Клавиши-стрелки** : Клавиши со стрелками перемещают курсор на одну позицию влево,вправо,вверх или вниз. Перемещение выше начала файла,ниже его последней линии или перемещение влево или вправо (автоматический перенос) за край строки не разрешены.
- **hjkl** : Когда **vi** был написан (примерно в 1978), многие терминалы на UNIX системах не имели клавиш-стрелок! **h, j, k,** и **l** были выбраны как команды для перемещения влево,вниз,вверх и вправо соотвественно. Попробуйте их! Большинство **vi** профессионалов предпочитают их, курсорным клавишам потому-что
  - **(а)** они располагаются одинаково на всех типах клавиатуры, и
  - **(б)** они приятней для пальцев, чем большинство курсорных клавиш, которые расположены ввиде "Т"или другой нелинейной форме.

  Почему h, j, k, и l? Хорошо, в наборе символов ASCII , CTRL-H это backspace (перемещение влево), CTRL-J это linefeed (перемещение вниз), и, конечно, k и l расположены за h и j,поэтому их легко запомнить.
- **0** : ("ноль", не "о") Перемещает курсор к началу текущей линии. (Чтобы попробывать эту и несколько последующих команд,используйте **h j k l** или клавиши со стрелками,чтобы перейти к строке с отступом,которая содержит несколько "е". Если вы не найдете такой строки - создайте её сами, вставив несколько пробелов в начало строки.
- **^** : Передвигает курсор на первый символ,не являющийся пробелом,в текущей строке.(Для строки с отступом, команды 0 и^действуют по разному).
- **$** : Перемещает курсор на последний символ текущей строки.
- **tC** : Передвигает к (не "на") следующий символ "с"в текущей строке. (Нажмите 0, а затем te. Это переместит вас к первой "е"в текущей строке.)
- **fC** : Находит (перемещает "на") первый символ "с"в текущей строке. (Нажмите fe, и курсор найдет (переместиться на позицию) следующего символа "е"в текущей строке.)
- **TC** : Передвигает к (но не "на") преведущий символ "с"в текущей строке. (Нажмите $, затем Te.)
- **FC** : Находит (передвигает "на") преведущий "с"в текущей строке. (Нажмите Fe.)

- **n|** : Передвигает к колонке к колонке n в текущей строке. (Попробуйте 20 |. Цифры 2 и 0 не будут отображаться во время ввода, но когда вы нажмете | курсор переместиться к колонке 20.)

  Попробуйте поэксперементировать с t f T F |. Когда вы сделаете что нибудь неправильное,**vi** сообщит вам о этом звуковым сигналом.

- **w** : Перемещает курсор вперед к началу следующего "маленького"слово ("маленькое"слово - состоит из символов алфавита и цифр не "разорванных"пробелом или знаков пунктуации, но не их смеси). Попробуйте нажать w несколько раз,чтобы понять как работает эта команда.

- **W** : Перемещает курсор вперед к началу следующего "большого слова"(смесь алфавитно-цифровых символов и пунктуации). Попробуйте применить эту команду несколько раз.

- **b** : Назад к началу "маленького"слова.

- **B** : Назад к началу "большого"слова.

- **e** : Вперед к концу "маленького"слова.

- **E** : Вперед к концу "большого"слова.

- **+ Enter** : Передвигает курсор к первому символу (не пробелу) на следующий строке. (+ и Enter имеют одинаковый эффект)

- **-** : Передвигает курсор к первому символу (не пробелу) на преведущей строке.

- **)** : Перемещает курсор к концу предложения. (Предложение кончается либо на пустой строке либо точкой или вопросительным/восклицательным знаком после которых идут два пробела,или на конце строки. Точка или восклицательный знак с пробелом после них,не заканчивают предложение;это корректное поведение,основанное на классических правилах составления печатных документов,хотя оно часто кажеться неправильным людям, которые не когда не изучали правила набора.)

- **(** : Перемещает курсор к началу предложения.

- } : Перемещает курсор к концу параграфа. (Параграфы разделяются пустыми строками по определению,используемому **vi**)

- { : Перемещает курсор к началу параграфа.

- **H** : Отправляет курсор на первую строку видимую на экране.

- **M** : Отправляет курсор на среднюю строку видимую на экране.

- **L** : Передвигает курсор к последней строке видимой на экране.

- **nG** : Перемещает курсор на строку n. Если n не задано, передвигает на последнюю строку файла. (Например: попробуйте 15G чтобы перейти к 15й линии. Комбинация CTRL-G отображает имя файла, некоторую информацию о нем, и номер текущей строки. Чтобы перейти к началу файла нажмите: 1G)

- **CTRL-d** : Скроллирует вниз половину экрана (смотрите примечание).

- **CTRL-u** : Скроллирует вверх половину экрана (смотрите примечание).

- **CTRL-f** : Перемещает курсор вперед на один экран (смотрите примечание).

- **CTRL-b** : Перемещает курсор на один экран назад (смотрите примечание).

- **Примечание** : Эти четыре команды для скроллинга не могут быть использованы совместно с командами удаления,изменения,копирования или фильтрования.

- **/регулярное_выражение** : Перемещает курсор к следующему регулярному выражению. Когда вы нажимаете /,курсор перемещается на командную строку и **vi** ждет ввода выражения Нажмите Enter чтобы завершить ввод;**vi** начнет поиск регулярного выражения вперед по тексту. Например введите /ро и нажмите Enter. Это передвинет курсор к первому "ро"встреченному при поиске вперед по тексту. Если вы просто введете / и нажмете Enter - **vi** начнет поиск выражение которые вы искали в последний раз.
- **n** : Имеет тот же эффект что и нажатие / и Enter; т.е., ищет то выражение которое было аргументом для команды поиска в последний раз.
- **?регулярное_выражение** : Ищет в обратном направлении. Если регулярное выражение не указано то производиться поиск аргумента последней операции поиска. Обе операции поддерживают автоматический перенос,т.е. поиск за "границами"файла разрешен.
- **N** :Тоже самое что ? и Enter.

9.2 Коэффициент повторения

Многие из приведенных выше команд перемещения курсора могут быть предварены коэффициентом повторения;в таком случае команда просто повторяется несколько раз.

- **3w** : Передвигает курсор вперед на три слова
- **5k** : Переходит вверх на пять символов
- **3fa** : Находит третье "а"в текущей строке
- **6+** : Переходит вниз на шесть линий

Для некоторых комманд коэффициент повторения имеет особое значение.

- **4H** : Переходит на четвертую строку видимую на экране
- **8L** : Переходит к восьмой строке от края(низа) экрана
- **3$** : Переходит к концу третьей строки (вниз)

Для некоторых комманд (например,) коэффициент повторения игнорируется; для других (например., / и ?) он применяться не может.

9.3 Удаление текста

Мы уже видели,что **dd** удаляет текущую строку. Эту команду можно использовать совместно с коэффициентом повторения: 3dd удаляет три строки, текущую и две следующих.
Команда d может быть использована совместно с практически любой командой перемещения по тексту, что позволяет удалять произвольные фрагменты текста. При использование совместно с d команды перемещения называются спецификаторами цели. (во время экспериментов со следующими примерами, не забывайте нажимать u чтобы восстанавливать то-что вы удалили).

- **dw** : Удаляет следующее "маленькое"слово
- **d3w** : Удаляет три следующих "маленьких"слова
- **3dw** : Три раза удаляет следующее "маленькое"слово
- **3d3w** : Три раза удаляет три следующих "маленьких"слова (т.е. удаляет девять "маленьких"слов)
- **d+** : Удаляет текущую строку и следующую за ней

- **d/the** : Удаляет текст начиная с текущего положения курсора вплоть до первого встреченного "the", не включая его .
- **d$** : Удаляет до конца строки
- **d0** : Удаляет до начала строки
- **d30G** : Удаляет от текущей строки до 30 строки (включая её)
- **dG** : Удаляет от текущей строки до последней (включая её)
- **d1G** : Удаляет от текущей строки до строки номер 1 (включая её)

Для удаления одного символа используйте x. x можно использовать совместно с коэффициентом повторения.

- **15x** : Удаляет текущий и 14 последующих символов

x на самом деле - макрос для d1; удаление одного символа.

9.4 Изменение текста

Команда c в основном идентична d, исключая то,что она переключает **vi** в режим вставки,позволяя изменить изначальный текст на что то другое.
Например : разместите курсор на начале какого-либо слова (нажмите w чтобы перейти к началу следующего слова). Затем нажмите cw , чтобы изменить это слово.На экране последний символ заменяемого слова будет изменен на **$**, индицирующий границы производимых изменений; попробуйте ввести новое слово и нажмите ESC когда закончите.Новый фрагмент текста может быть длинее/короче чем тот который был удален.
Поместите курсор на начале строки содержащей как минимум три слова, и попробуйте c3w для изменения этих слов. Попробуйте $ чтобы изменить текст до конца текущей строки. Во всех случаях когда изменение касается только текущей строки конец изменяемого текста помечается символом $.
Когда изменение касается не только текущей строки,**vi** удаляет оригинальный текст и переходит в режим вставки. Например попробуйте c3+ для изменения текущей и трех последующих строк;**vi** удалит четыре оригинальных строки на экране и перейдет в режим вставки на новой пустой строке. Как и обычно нажмите ESC чтобы вернуться в командный режим.
Некоторые примеры команд изменения:

- **cc** : Изменяет текущую строку
- **5cc** : Изменяет пять строк (текущую и четыре следующие)
- **c/the** : Изменяет текст с текущего символа вплоть до (но не включая) следующего встреченного в тексте the
- **c$** : Изменяет до конца строки
- **c30G** : Изменяет от текущей до строки номер 30, включая её
- **cG** : Изменяет от текущей до последней строки, включая её
- **c1G** : Изменяет от текущей строки до строки номер 1, включая её

9.5 Копирование текста

Команда y копирует фрагмент текста в буфер;из буфера скопированный текст может быть вставлен в любую часть файла используя команду p или P.
Самая простая форма копирования - скопировать текущую строку;введя yy, попробуйте p, чтобы вставить копию скопированной строки после курсора. Вы можете вставить эту строку в файл столько раз, сколько это необходимо, перемещаясь по файлу и нажимая p.
Чтобы скопировать множество строк, попробуйте, например, 5yyy (копирует текущую строку и следующие четыре строки). p вставляет скопированные линии после курсора; команда 5yyp "работает"но вероятно не делает того, что вам необходимо. Команда P делает тоже что и p, но помещает копию скопированного текста перед курсором; попробуйте эту последовательность - 5yyP.
Другие команды для вставки:

- **y3w** : Копирует три слова
- **y$** : Копирует до конца текущей строки
- **y1G** : Копирует от текущей и до строки номер 1 (включая её)

9.6 Фильтрование текста

Команда фильтрования **vi**, запрашивает у пользователя имя команды UNIX (которая должна быть фильтром), затем пропускает выбранные строки сквозь фильтр,заменяя эти строки на строки обработанные фильтром. Возможность **vi** по фильтрованию произвольных фрагментов текста, через любой UNIX фильтр, добавляет гибкость и значительно расширяет ваши возможности по редактированию текста, без "дополнительной платы"в виде размера или производительности самого **vi**.
Несколько примеров помогут это проиллюстрировать. Создайте строку в вашем файле, содержащую слово who и ничего более. Поместите курсор на эту строку и нажмите **!!**. Эта команда похожа на dd,cc или yy но вместо удаления,изменения или копирования текущей строки,она фильтрует её. Когда вы нажимаете ! второй раз,курсор переходит на командную строку и в ней высвечивается подсказка !,приглашающая вас ввести название фильтра. В качестве имя фильтра введите sh и нажмите Enter. **sh** (Bourne shell) это фильтр! Он считывает команды со стандартного ввода, исполняет их и выдает результат исполнения на стандартный вывод. Команда who будет выполнена и в качестве результата выдаст список пользователей подключенных в данный момент к системе. В итоге вы получите этот список прямо в вашем файле, вместо слова who.
Попробуйте проделать то же самое с **date**. Создайте строку содержащую только слово **date**, затем поместите курсор на этой строке, введите **!!sh** и нажмите Enter. Строка будет заменена на вывод команды **date**
Поместите ваш курсор на какую-нибудь строку. Отсчитайте несколько строк, скажем 6. Введите **6!!sort** и нажмите Enter. Оригинальные 6 строк будут заменены на вывод команды **sort**.
Команду фильтрования можно применять только на всей строке сразу, но не на отдельных словах или символах.
Некоторые другие команды фильтрования (здесь, < CR > означает - нажмите Enter):

- **!/the** < **CR** > **sort** < **CR** > : Сортирует текст от текущей строки до следующей строки содержащей the.
- **!1Ggrep the** < **CR** > :Уничтожает все строки от текущей до 1й (включая её), если они не содержат the.
- **!Gawk '{print $1}'** < **CR** > : Оставляет от всех строк (начиная с текущей и до конца файла) только первое слово.

9.7 Установка пометок

Вы можете выделить фрагмент текста, чтобы переместить,удалить,изменить,скопировать или отфильтровать его используя команду mc.
Например разместите курсор в центре какого-либо слова и нажмите ma. Это пометит символ под курсором меткой a.
Теперь переместите курсор с помеченного символа в другое место (используйте клавиши со стрелками или CTRL-u). Чтобы вернуться к помеченной строке нажмите 'a (одинарная скобка и a). Это переместит курсор на первый символ (не пробел) на помеченной строке.
Вновь уберите курсор с этой строки. Чтобы вернуться к помеченному символу нажмите ‘a (обратная одинарная скобка и a). Это поместит курсор поверх символа помеченного как a.
Установка меток обычно используется для его удаления,изменения,копирования или фильтрования. Например : переместите курсор на строку без метки и нажмите d'a (d,одинарная скобка,a). Эта команда удалит все начиная от текущей строки и заканчивая строкой с меткой.
Поместите курсор в центре какого-либо слова и нажмите mb чтобы установить метку b. Отодвиньте курсор на несколько строк и нажмите d‘b (d,обратная одинарная скобка,b). Эта команда удалит все начиная от текущего положения курсора и заканчивая помеченным символом (включая его).
Другой пример : для сортировки вывода команды who, поставьте метку на первой строке вывода (ma), затем передвиньте курсор на последнюю строку и введите !'asort и нажмите Enter.
Если после того как вы перешли к метке вы захотите вернуться назад,нажмите '' чтобы вернуться к строке или `` чтобы вернуться к символу.

9.8 Назначение имени буферам

Когда вы удаляете,изменяете или копируете текст, оригинальный текст сохраняется (до следующей операции удаления, изменения или копирования) в безымянном буфере, из которого его можно извлечь с помощью p или P. Таким образом, в восстановить из безымянного буфера можно лишь недавно помещенный туда текст.
Если вы хотите удалить,изменить или скопировать несколько секций текста и при этом сохранить их все (вплоть до максимально возможных 26), вы можете задать имя буфера перед командой удаления или копирования.Имя буфера имеет следующий синтаксис "c (двойные кавычки и c).
Например введите "ayy чтобы скопировать текущую строку в буфер с именем a,затем перейдите на другую строку и нажмите "byy чтобы скопировать её в буфер b. Теперь перейдите в другое место в файле и нажмите "ap и "bp чтобы вставить содержимое буферов a и b.
Несколько других команд использующих буфер:

- **"a6yy** : Копирует шесть строк (текущую и пять следующих) в буфер a
- **"bd1G** : Удаляет все строки от текущей до первой (включая её),и размещает их в буфере b.
- **"cy'c** : Копирует текст начиная с текущей строки вплоть до строки отмеченной c и помещает скопиррованый фрагмент в буфер c. (отметки и буферы не связаны друг с другом и могут иметь одинаковые имена).

9.9 Замещение текста

Чтобы заместить один фрагмент текста другим используйте команду :s. Несколько примеров:

- **:1,$s/the/THE/g** Начиная с первой строки до последней (строки $), заместить все встреченные the на THE (g - означает глобальную замену)
- **:'a,.s/.*/ha ha/** От строки помеченной как a до текущей (строки .), заменить любой текст на строку "ha ha".

9.10 Команды начинающиеся с двоеточия

Когда вы вводите двоеточие в командном режиме,курсор автоматически перемещается на командную строку и ждет ввода команды.
Некоторые примеры таких команд:

- **:w** Записывает содержимое буфера в файл без выхода из **vi**
- **:w abc** Записывает содержимое буфера в файл abc (если его нет он будет создан,иначе - перезаписан.) не выходя из **vi**
- **:1,10w abc** Записывает строки с 1 по 10 в файл abc
- **:'a,$w abc** Записывает строки от строки, помеченной как "a", до последней строки abc
- **:e abc** Загружает для редактирования файл abc, вместо текущего файла. Выдает ошибку если изменения в текущем файле не были сохранены.
- **:e! abc** Загружает для редактирования файл abc, не обращая внимание на несохраненные изменения в текущем файле.
- **:e #** Загружает для редактирования предшествующий файл (команда :e# следующий)
- **:f abc** Изменяет имя текущего буфера **vi** на abc
- **:q** Выходит из редактора, если нет несохраненных изменений
- **:q!** Выходит и уничтожает при этом все несохраненные изменения
- **:r abc** Считывает файл abc в текущий буфер **vi**, и размещает его за курсором.
- **:!cmd** Выполняет команду (who, sort, ls, etc.)

9.11 Установка опций

Существует множество опций которые влияют на работу **vi**. Вы можете отобразить все доступные опции с помощью команды :set all. Вы также можете использовать команду set для установки различных опций.
Например если вы хотите чтобы для строк в файле отображались их порядковые номера, используйте команду ":set number". Чтобы отключить отображение номеров строк введите ":set nonumber". Для указания большинства опций можно использовать аббревиатуры, например ":set nu включит нумерацию строк, а ":set nonu"отключит её.
Если вы установите ":set nomagic", то символы регулярных выражений имеющие специальное значение (период,астерик,квадратные скобки и т.д.) будут восприниматься как обычные. Используйте ":set magic"чтобы вернуть все назад.
Некоторые опции могут иметь параметры. Например ":set tabstop=4"заставит символ табуляции отображаться как четыре пробела, вместо обычных восьми.
Если вы хотите чтобы ваши опции сохранились и при следующем сеансе работы,вы можете поместить их в файл ".exrc",или установить переменную окружения EXINIT соответствующим образом.
Например если в качестве shell вы используете Bourne shell,вы можете разместить в вашем .profile такую строчку.

```
EXINIT='set nomagic nu tabstop=4'; export EXINIT
```

Если вы используете C shell, необходимо разместить следующую строку в вашем .login файле.

```
setenv EXINIT 'set nomagic nu tabstop=4'
```

9.12 Настройка клавиатурных комбинаций

Если вам приходиться выполнять серию простых команд опять и опять, то вы можете назначить этим командам определенную клавиатурую комбинацию (из неиспользуемых) с помощью команды :map.
Например следующая команда настраивает CTRL-A для передвижения курсора на 55 строк вперед, затем назад на самую последнюю пустую строку,на изменение этой командной строки на formfeed (CTRL-L) и три пустых строки. Т.е. CTRL-A отформатирует следующую страницу, без разделения параграфов между страницами.
Примечание: В этой строке каждый символ, который нужно нажимать совместно с CTRL показан как ^C, где C это большая буква. Например, CTRL-M показано как ^M. Также во время ввода вы не увидете ^V, т.к. CTRL-v нужно для того, чтобы подавить управляющее значение следующей за ним комбинации клавиш. Т.е. при вводе ^VM вы увидете на экране лишь ^M(соотвествует Enter). ^[соотвествует ESC.

```
:map ^A  55+?^$^V^Mcc^V^L^V^M^V^M^V^M^V^[
```

9.13 Редактирование нескольких файлов

Вы можете редактировать несколько файлов, задав их имена как аргументы при запуске **vi**.

```
vi croc fatherw  wknight
```

Для перемещения между файлами используются две команды:

- **:n** Перейти к следующему файлу в списке аргументов (вы должны сохранить изменения при помощи :w или **vi** выдаст сообщение о ошибке)
- **:rew** Перейти к редактированию первого файла в списке аргументов.

Команды :n и :rew, в каком то смысле неповоротливы, но они предоставляют несколько весьма полезных возможностей: содержимое поименованных буферов ("a,"b и т.д.) запоминается глобально, и вы сможете использовать :n и :rew команды совместно с p или P, чтобы копировать текст из одного файла в другой. Более того, запрос поиска, также запоминается глобально, и вы сможете искать одну строку в нескольких файлах.
Попробуйте провести следующий эксперимент: Выйдете из **vi**, запустите **vi** с croc и wknight в качестве аргументов:

```
$ vi croc wknight
```

В файле croc, найдите
/the < **CR** >
Скопируйте это в буфер a:
"ayy
Перейдите к следующему файлу (мы не сделали никаких изменений в croc,поэтому это сработает):
:n < **CR** >
Проведем поиск еще раз, использовав в качестве аргумента поиска преведущий образец.
n
Поместите копию буфера a после текущей строки в wknight:
"ap
Перейдите на две строки вниз, и скопируйте текущую строку в буфер b:
jj"byy
Сохраним изменения в wknight
:w < **CR** >

Теперь вернемся к croc
:rew < **CR** >
Произведем поиск еще раз и разместит копию буфера b после найденной линии:
n"bp
Сохраним изменения и выйдем из **vi**
ZZ

9.14 Финальные замечания

Это пособии представляет некоторые возможности **vi** которые вы могли пропустить или не понять читая пособие по **vi** установленное в вашей системе.(на различных системах встречаются разные пособия).
Вы не станете экспертом по **vi** после чтения этого пособия,но вы ,по крайней мере, сможете оценить его возможности. Только время и тренировка могут сделать вас экспертом. Но универсальность и эффективность **vi** стоят этих усилий.
Вы можете решить что вы ненавидите **vi**. Но не забывайте что это единственный редактор доступный практически на любой UNIX системе, поэтому необходимо знать хотя бы самый минимум его команд.

10 Справочник по командам vi

10.1 Режимы Vi

Vi имеет 3 режима:

1. ***командный режим*** - Включается при запуске vi; для перехода в командный режим из других режимов (используйте **ESC**)
2. ***режим ввода*** - активируется специальными командами **a i A I o O c C s S R** и выключается при помощи **ESC** или в случае ошибки.
3. ***линейный режим*** - т.е. ожидание ввода команд после нажатия **:** , **/** , **?** или **!** (выключается с **CR**, обрывается с помощью **CTRL-c**). **CTRL** это клавиша control: **CTRL-c** означает "control c"

10.2 Команды Shell

1. **TERM=** *code* Помещает имя вашего терминала в переменную **TERM**
2. **export TERM** Передаёт значение переменной **TERM** (код терминала) любой программе в UNIX системе.
3. **tput init** Инициализирует терминал таким образом, чтобы он функционировал правильно с различными UNIX программами.
4. **vi** *filename* Запускает **vi** и загружает на редактирование указанный файл.
5. **vi** *file1 file2 file3* Записывает в буфер **vi** три файла для последующего их редактирования. Эти файлы *file1, file2,* и *file3.*
6. **view** *file* Загружает файл *file* в режиме только-для-чтения.
7. **vi -R** *file* Аналогично преведущей команде.
8. **vi -r** *file* Восстанавливает *file* и последние изменения после краха системы.

10.3 Установка параметров

1. **:set** *опция* Активизирует *опцию*
2. **:set** *опция=значение* Назначает *значение* для *опции*
3. **:set no** *опция* Деактивирует *опцию*
4. **:set** Показывает опции установленные пользователем.
5. **:set all** Показывает все опции, как установленные пользователем, так и опции "по умолчанию".
6. **:set** *опцию*? Показывает значение *опции*

10.4 Используемые обозначения

1. **CTRL-c CTRL** Это клавиша control: **CTRL-c** означает "control c"
2. **CR** символ возврата каретки (клавиша ENTER)

10.5 Обрывание и отмена

- **ESC** Обрывает ввод неправильно или неполностью введенной команды
- **CTRL-? CTRL** это клавиша control : **CTRL-?** означает "control ?"удаляет или стирает прерывание.
- **CTRL-l** обновляет экран если CTRL-? замусорила его.

10.6 Манипуляции с файлами

- **ZZ** Сохраняет файл и выходит из vi
- **:wq** Сохраняет файл и выходит из vi
- **:w** Записывает файл
- **:w!** Записывает файл даже если он защищен от записи
- **:w*имя*** Записывает файл по именем *имя*
- **:q** Выходит из vi
- **:q!** Выходит из vi (независимо от того были ли внесены изменения)
- **:e имя** Загружает для редактирования файл под именем *имя*
- **:e!** Перезагрузить файл,стирая любые изменения которые были внесены
- **:e + name** Загружает для редактирования файл под именем *имя*, начиная с конца.
- **:e + n** Начинает редактирование со строки номер *n*
- **:e #** Редактирует альтернативный файл
- **:n** Редактирует следующий файл в списке файлов
- **:args** Показывает файлы в текущем списке файлов
- **:rew** Обновить текущий список файлов и начать редактировать первый файл.
- **:n args** Указать новый список файлов

- **:f** Показывает имя текущего файла и номер строки
- **CTRL-G** Синоним для :f
- **:ta tag** to tag file entry *tag*
- **CTRL-]** :ta, следующее слово - тэг

10.7 Перемещение

- **Arrows** Перемещают курсор
- **CTRL-d** Скроллирует полстраницы вниз
- **CTRL-u** Скроллирует полстраницы вверх
- **CTRL-f** Скроллирует целую страницу вниз
- **CTRL-b** Скроллирует целую страницу вверх
- **:0** Переходит к началу файла
- **:n** Переходит к строке номер n
- **:$** Переходит к концу файла
- **0** Переходит к началу строки
- **^** Переходит к первому символу, не пробелу
- **$** Переходит к концу строки
- **CR** Переходит к началу следующей строки
- **-** Переходит к началу преведущей строки
- **%** Находит соответствующую скобку
- **G** Переходит к строке (по умолчанию - к последней строке)
- **]]** Следующая секция/функция
- **[[** Преведущая секция/функция

10.8 Позиционирование по строкам

- **H** Первая строка экрана
- **L** Последняя строка экрана
- **M** Средняя строка экрана
- **+** Следующая строка, первый символ не являющийся пробелом
- **-** Предшествующая строка, первый символ не являющийся пробелом
- **CR** возврат, тоже что и +
- **j** Следующая строка, таже колонка
- **k** Преведущая строка, таже колонка

10.9 Позиционирование по символам

- **0** Начало строки
- **$** Конец строки
- **h** Вперед
- **l** Назад
- **SPACE** Тоже что и l
- **fx** Найти x вперед по тексту
- **Fx** Найти x назад по тексту
- **;** Повторить последнюю операцию f F
- **,** Инверсно ;
- | К специфицированной колонке
- **%** Найти соответствующий символ { или }

10.10 Слова, предложения параграфы

- **w** Слово вперед
- **b** Слово назад
- **e** Конец слова
- **)** К следующему предложению
- **(** К преведущему предложению
- } К следующему параграфу
- { К преведущему параграфу
- **W** К концу большого слова, включая пробел на его конце
- **B** Назад на большое слово W
- **E** К концу большого слова W

10.11 Установка отметок и возврат к ним

- **‘‘** (дважды нажмите клавишу ‘) К преведущей отметки
- **’’** (press twice the single-quote ‘ key) Последняя отметка, первый не пробел в строке
- **mx** Установка отметки с именем x
- **‘x** (обратная кавычка и x) перейти к отметке x
- **'x** Перейти к отметке x, на первый не пробел в строке

10.12 Коррекция во время ввода текста

- **CTRL-h** Удаляет последний символ
- **CTRL-w** Удаляет последнее слово
- **erase** Нажмите DELETE, тоже что и CTRL-h
- **kill** Ваша клавиша kill, стирает ввод сделанный на этой строке
- \ Экранирует CTRL-h, DELETE и kill
- **ESC** Окончить ввод, назад в командный режим
- **CTRL-?** Прервать, терминирует ввод
- **CTRL-d** Обратная табуляции на *autoindent* пробелов
- **CTRL-v** Экранирует непечатный символ

10.13 Обновления экрана

- **CTRL-l** Очистить и перерисовать
- **CTRL-r** перенабор, убирает @строки
- **z-CR** перерисовать, текущую строку на вершине окна
- **z-** перерисовать, текущую строку на краю окна
- **z.** перерисовать, текущую строку в центре экрана
- **/pat/z-** *сгладить* край строки
- **tn** Использовать экран с строкой n
- **CTRL-e** Скроллировать окно вниз на 1 строку
- **CTRL-y** Скроллировать окно вверх на 1 строку

10.14 Удаление

- **x** Удаляет символ под курсором
- **X** Удаляет символ перед курсором
- **D** Удаляет до конца строки
- **d^** Удаляет до начала строки
- **dd** Удаляет текущую строку
- **ndd** Удаляет *n* строк начиная с текущей
- **dnw** Удаляет *n* слов начиная с положения курсора

10.15 Вставка, замена

- **i** Входит в режим вставки (с вставкой перед курсором)
- **I** Входит в режим вставки (перед первым не пробелом)
- **a** Входит в режим вставки (вставка после курсора)
- **A** Входит режим вставки (вставка после конца текущей строки)
- **o** Создать новую строку после текущей и перейти в режим вставки
- **O** Создает новую строку перед текущей и входит в режим вставки
- **r** Заменить символ под курсором не переходя в режим вставки
- **R** Войти в режим замены
- **C** shift-c. Изменить остаток строки
- **D** shift-d. Удалить остаток строки
- **s** Заменить символы
- **S** Заменить строки
- **J** Объединить строки

10.16 Копирование и вставка

"Буфер копирования"заполняется *КАЖДОЙ* командой удаления,или с помощью **Y** и **yy**.

- **Y** Копирует текущую строку в буфер
- ***n*yy** Копирует *n* строк начиная с текущей в буфер копирования
- **p** Вставляет содержимое буфера копирования после курсора (или перед текущей строкой)
- **P** Вставляет содержимое буфера обмена перед курсором (или перед текущей строкой)
- **"*x*p** Вставить из буфера x
- **"*x*y** Скопировать в буфер x
- **"*x*d** Удалить в буфер x

10.17 Операции (для применения к строкам вводятся дважды)

- **d** удалить
- **c** изменить
- **<** сдвинуть влево
- **>** сдвинуть вправо
- **!** отфильтровать через команду
- **=** отдать для LISP
- **y** скопировать текст в буфер

10.18 Поиск и замена

- ***/text*** Искать вперед образец *text*
- **?*text*** Искать назад образец *text*
- **n** Повторить последний поиск в том же направлении
- **N** Повторить последний поиск в обратном направлении
- **/** Повторить последний поиск вперед
- **?** Повторить последний поиск назад
- **[addr] s/from/to/ [g]** Искать *from* и заменить его на ***to*** в текущей строке, или в указанном диапазоне ***addr*** (два номера строки разделенные командой; 1,$ весь файл). Заменить одно совпадение на строку либо все совпадения если ***g*** указано. Например, :3,20s/someword/anotherword/g. Заменит "someword"на "anotherword"начиная от строки номер 3 до строки 20. 'g' - означает замену всех совпадений.

10.19 Общие

- **:sh** Вызывает shell (выход по CTRL-d)
- **:!*команда*** Вызывает shell для исполнения *команды*
- **:set number** Включает нумерацию строк
- **:set nonumber** Выключает нумерацию строк

10.20 Команды линейного редактора

- **:** Говорит **vi** что следующая введенная команда является командой линейного редактора.
- **:sh** Временный выход в shell чтобы исполнить какие либо команды не покидая **vi**.
- **CTRL-d** Выходит из shell запущенного преведущей командой в **vi**.
- **:***n* Переходит к строке номер *n*th текущего буфера.
- **:***x*,*z***w** *filename* Записывает строки от *x* до *z* в новый файл называемый *filename*.
- **:$** Передвигает курсор к началу последней строки буфера.
- **:.,$d** Удаляет все строки от текущей до последней.
- **:r** *filename* Вставляет содержимое файла *filename* после текущей строки буфера.
- **:s***/text/new_text/* Заменяет первый встреченный образец *text* на текущей строке на *new_text*
- **:s***/text/new_text/g* Заменяет все образцы *text* на текущей строке на *new_text*
- **:g***/text/s//new_text/g* Заменяет все встреченный в буфере образцы *text* на *new_text*.

10.21 Другие команды

- **u** Отменить последнее изменение
- **U** Восстановить текущую строку
- **~** Изменить регистр
- **J** Соединить текущую строку со следующей
- **.** Повторить последнюю команду изменения текста
- **CTRL-g** Показать имя файла и номер строки

11 Другие форматы данного документа

Этот документ (его английский оригинал) доступен в 10 различных форматах - DVI, Postscript, Latex, LyX, GNU-info, HTML, RTF(Rich Text Format), Plain-text, Unix man pages и SGML.

- Вы можете забрать этот HOWTO в виде одного тар-болла в HTML,DVI,Postscript или SGML форматах с - <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO/other-formats/> or <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO/other-formats/>
- Текстовый вариант доступен по адресу: <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO> или <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO>
- Переводы на другие языки такие как French, German, Spanish, Chinese, Japanese находятся на <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO> или <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO> Любая помощь по переводу на другие языки приветствуется ;) .

Этот документ написан с использованием "SGML tool",который вы можете найти по адресу:
<http://www.xs4all.nl/~cg/sgmltools/>
Компилирование в различные формы производиться с помощью команд:

- sgml2html vim-howto.sgml (для генерации html файла)
- sgml2rtf vim-howto.sgml (для генерации RTF файла)
- sgml2latex vim-howto.sgml (для генерации latex файла)

Этот документ можно найти по адресу -

- <http://metalab.unc.edu/LDP/HOWTO/VIM-HOWTO.html>

Также вы можете найти этот документ на следующих зеркалах -

- <http://www.caldera.com/LDP/HOWTO/VIM-HOWTO.html>
- <http://www.WGS.com/LDP/HOWTO/VIM-HOWTO.html>
- <http://www.cc.gatech.edu/linux/LDP/HOWTO/VIM-HOWTO.html>
- <http://www.redhat.com/linux-info/ldp/HOWTO/VIM-HOWTO.html>
- Список зеркал можно найти по адресу <http://metalab.unc.edu/LDP/hmirrors.html> выберете сайт и идите в директорию /LDP/HOWTO/VIM-HOWTO.html

Чтобы просмотреть этот документ в dvi формате, используйте программу xdvi. (пакет tetex-xdvi*.rpm в дистрибьюции RedHat)

```
Чтобы читать dvi документы задайте команду.
        xdvi -geometry 80x90 howto.dvi
И изменяйте размер окна мышкой. Также смотрите соответствующую man страницу.
Для навигации используйте стрелки, Page Up, Page Down, или
'f', 'd', 'u', 'c', 'l', 'r', 'p', 'n'
чтобы перемещаться по тексту.
Чтобы выключить режим эксперта нажмите 'x'.
```

Вы можете прочитать postscript файл используя программу gv. (пакет gv*.rpm)

```
Чтобы читать postscript документы дайте команду
        gv howto.ps

или
        ghostscript howto.ps
```

Вы можете читать HTML документы используя Netscape Navigator, Microsoft Internet Explorer, или любой другой web браузер.
Вы также можете читать latex файлы используя LyX.