**Кабинет информационных технологий**

**Главного учебного центра ОАО «Укртелеком»**

**в г. Луганске**

***Реферат***

**на тему:**

***Работа с редактором Vi.***

***Другие текстовые редакторы UNIX.***

слушатель: ***Литвин В.Н.***

группа: ***11.2.8***

руководитель: ***Зайко К.В.***

**г. Луганск 2003г.**

**Содержание**

Введение 3

1. Начальные установки 3

2. Создание файла 4

3. Режимы редактора 5

4. Редактирование текста в командном режиме 6

5. Завершение работы c vi 7

6. Перемещение курсора по экрану 8

7. Позиционирование курсора на неотображаемом тексте 10

8. Переход на указанную строку 11

9. Номера строк 11

10. Поиск символов по шаблону: команды / и ? 12

11. Создание текста 12

12. Удаление текста 13

13. Команды удаления в командном режиме 14

14. Модификация текста 15

15. Извлечение и присоединение текста 17

16. Копирование и перемещение текста с использованием регистров 18

17. Специальные команды 18

18. Использование команд построчного редактора 19

19. Завершение работы с редактором vi 21

20. Специальные опции для vi 22

21. Другие текстовые редакторы UNIX 23

Список источников информации 28

### Введение

Редактор vi является мощным средством для создания и редактирования файлов. Он предназначен для работы на видеотерминале и использует окна, посредством которых можно просматривать текст файла. Несколько простых команд позволяют сделать изменения в тексте, которые очень быстро отображаются на экране. Редактор может отобразить одну или несколько строк текста. Он позволяет передвигать курсор в любую точку на экране или в файле, создавать, изменять или удалять текст. Можно также использовать некоторые команды построчного редактора, такие как глобальные команды, которые позволяют изменять множество появлений некоторой строки символов, используя одну команду. Чтобы двигаться по файлу, можно прокручивать текст вперед или назад.

Редактор vi имеет более 100 команд. В этом разделе описаны основные команды, которые позволят использовать vi просто, но эффективно. Особое внимание будет уделено выполнению следующих задач, позволяющих:

* изменить окружение shell, чтобы установить конфигурацию вашего терминала;
* сделать доступным автоматический возврат каретки;
* начать работу с редактором vi, создать текст, исправить ошибки, записать текст в файл и завершить работу;
* переместить текст внутри файла;
* разбить и слить текст;
* использовать специальные команды и сокращения;
* использовать команды построчного редактора, доступные внутри vi;
* временно вернуться в shell, чтобы выполнить команды shell;
* восстановить файл, потерянный из-за прерывания сессии редактирования;
* редактировать несколько файлов в одной сессии.

### 1. Начальные установки

В этом подразделе описано, как делать установку конфигурации терминала, чтобы правильно работал редактор vi.

#### *1.1. Установка конфирурации терминала*

Прежде чем начать работать с редактором vi, необходимо сообщить системе UNIX, какой тип терминала будет использоваться. Это связано с тем, что программное обеспечение vi по разному исполнимо на разных терминалах.

Каждому типу терминала соответствует кодовое имя, которое распознается системой UNIX.

Чтобы установить конфигурацию терминала, необходимо ввести:

TERM= terminal\_name<CR>

export TERM<CR>

tput init<CR>

В первой строке переменной TERM присваивается конкретное значение (тип терминала). Вторая строка экспортирует это значение; она передает это значение всем программам системы UNIX, выполнение которых зависит от типа используемого терминала.

Команда tput в третьей строке производит установку программного обеспечения для терминала, чтобы он правильно функционировал с системой UNIX.

#### *1.2. Изменение конфигурации*

Если редактор vi будет использоваться часто, то необходимо сделать соответствующие изменения по установке конфигурации терминала в файле .profile. Этот файл должен находиться в собственном справочнике.

Если указать соответствующие установки для терминала в файле .profile, то терминал будет автоматически настраиваться каждый раз при регистрации.

Если необходимо, чтобы клавиша RETURN отрабатывала автоматически во время работы с редактором vi, создайте файл с именем .exrc в собственном справочнике. В файле .exrc будут содержаться опции, которые управляют средой редактора vi.

Чтобы создать файл .exrc, нужно ввести при вызове редактора имя этого файла. Затем в первой строке текста введсти спецификацию циклического режима (автоматический возврат каретки). Формат этой опции:

: set wm=n<CR>

где n - число символов на строке экрана, после которого должен автоматически появляться возврат каретки. Он может стоять только между словами, но не между слогами слова.

Чтобы проверить установки и циклический режим для вашего терминала в редакторе vi, нужно ввести команду:

: set<CR>

Команду set также можете использовать и для создания или изменения опции циклического режима.

### 2. Создание файла

Чтобы создать файл, введите:

vi filename<CR>

Когда ввести команду vi с именем файла, vi очистит экран и отобразит окно, в которое можно вводить и редактировать текст.

### 3. Режимы редактора

Редактор vi функционирует в двух режимах: режим ввода и командный режим. В режиме ввода можно добавлять и модифицировать текст; в командном режиме можно:

* редактировать и изменять существующий текст;
* удалять, перемещать и копировать текст;
* передвигаться по всему файлу;
* выполнять другие задачи.

#### *3.1. Создание текста в режиме Append*

Если вход в редактор vi сделан успешно, то мы сразу же попадаем в командный режим и vi ожидает ввода ваших команд. Чтобы создать текст необходимо:

* ввести a, чтобы перейти к режиму ввода (не нажимая клавишу RETURN). Теперь можно добавить текст к файлу (файл не печатается на экране);
* ввести нужный текст;
* перейти на новую строку и нажать клавишу RETURN.

Если был указан циклический режим в файле .exrc, то будем переходить на новую строку всякий раз, когда достигнем точки автоматического возврата каретки.

#### *3.2. Выход из режима Append*

После окончания создания, нажать клавишу ESCAPE, чтобы из режима ввода перейти к командному режиму. Теперь можно редактировать текст, который был создан или записан <a>

Create some text<CR>

in the screen editor<CR>

and return to<CR>

command mode.<ESC>

Если нажать клавишу ESCAPE и будет слишен звуковой сигнал, то это значит, что мы уже перешли в командный режим. Нажатие клавиши ESCAPE в то время, когда находимся в командном режиме, никак не влияет на текст в файле, даже если мы нажмем ее несколько раз.

### 4. Редактирование текста в командном режиме

В общем случае редактирование файла - это добавление, изменение, удаление текста. Чтобы выполнить эти задачи вы должны научиться переходить от одной страницы текста к другой странице, от одной строки к другой и между определенными точками на строке. В этом подразделе описаны команды добавления и удаления текста и способы премещения курсора.

#### *4.1. Перемещение курсора*

Клавиши перемещения курсора позволяют вам перемещать курсор по файлу. Курсор можно переместить с помощью следующих клавиш:   
h - перемещает курсор на один символ влево;   
j - перемещает курсор вниз на одну строку;   
k - перемещает курсор вверх на одну строку;   
l - перемещает курсор на один символ вправо;

Команды j и k сохраняют позицию курсора на строке. Например, если курсор находился на 17-м символе слева, то когда вы введете команду j или k, то он переместится на новую строку, но будет находиться на 17-м символе. Если символов в новой строке меньше, то курсор переместитcя на последний символ строки.

Примечание. Некоторые терминалы имеют специальные символы управления курсором, помеченные стрелками. Эти стрелки можно использовать вместо клавиш h, j, k, l.

Если перед h, j, k, l поставить число, то курсор передвинется на количество строк/символов, равных числу. Например, если надо передвинуть курсор на 4 строки вверх, введите 4k. Если вы не можете продвинуться дальше в этом направлении, то vi подаст звуковой сигнал и курсор останется на прежнем месте.

Чтобы переместить курсор на 35 строк вниз, введите: 35j

Дополнительно к клавишам h и l можно использовать клавиши SPACEBAR и BACKSPACE для перемещения курсора вправо или влево.   
<SPACEBAR> - переместить курсор на один символ вправо;   
<nSPACEBAR> - переместить курсор на n символов вправо;   
<BACKSPACE> - переместить курсор на один символ влево;   
<nBACKSPACE> - переместить курсор на n символов влево;

#### *4.2. Удаление текста*

Если вы хотите удалить символ, переместите курсор к этому символу и нажмите x. Перед x может стоять число, которое говорит о том, сколько символов подряд нужно удалить. Редактор vi так корректирует строку, что на месте удаленного символа не остается пустого промежутка или на месте удаленной строки - пустой строки.

#### *4.3. Добавление текста*

В редакторе vi существует две команды для добавления текста: insert(i) и append(a). Чтобы добавить текст с помощью команды i, переместите курсор в нужное место с помощью клавишей h, j, k, l. Затем нажмите i и введите текст. Введенный вами текст появится на экране слева от символа, на который указывает курсор. Этот символ и все остальные справа от курсора будут передвинуты вправо, чтобы освободить место для вашего нового текста. Редактор vi будет хранить символы, вводимые вами, до тех пор, пока вы не нажмете клавишу ESCAPE. Если необходимо, передвинутые символы будут перенесены на следующую строку.

Таким же образом вы можете использовать команду a. Отличие состоит в том, что новый текст появится справа от символа, к которому подведен курсор.

### 5. Завершение работы c vi

Когда вы закончите работу с текстом, необходимо записать содержимое буфера в файл и возвратить управление shell. Для этого нажмите клавишу SHIFT и дважды нажмите Z(ZZ). Редактор сохранил имя того файла, который вы указали в начале сеанса редактирования, и в этот файл перепишет текст из буфера. Обратите внимание, что внизу экрана дается имя файла и количество строк и символов в файле. Затем shell выдает вам подсказку.

Пример.

<a>

This is a text file.<CR>

I am adding text to<CR>

a temporary buffer and<CR>

now it is perfect.<CR>

I want to write this file,<CR>

and return to the shell.<ESC><ZZ>

-

-

-

-

"stuff" [New file] 6 lines, 135 characters

$

Вы также можете использовать команды :w и :q редактора vi для записи файла и окончания работы с редактором. Команды редактора начинаются с двоеточия (:) и появляются в последней строке экрана. Команда :w записывает буфер в файл. Команда :q завершает работу редактора и возвращает управление shell. Вы можете ввести эти команды раздельно, или в одной команде :wq

### 6. Перемещение курсора по экрану

Ранее вы познакомились с тем, как перемещать курсор с помощью клавишей h, j, k, l, BACKSPACE и SPACEBAR. В этом подразделе приводятся еще несколько команд, которые помогут вам быстро перемещать курсор по экрану. Эти команды выполняются в командном режиме.

#### *6.1. Позиционирование курсора на символ*

Для позиционирования курсора на символ в строке существуют три способа:

* переместить курсор вправо или влево до нужного символа;
* переместить курсор в начало или конец строки;
* поиск символа на строке.

Переместить курсор на первый или последний символ в строке можно следующим образом:

$ - переместить курсор на последний символ в строке;   
0(ноль) - переместить курсор на первый символ в строке;   
^ - переместить курсор на первый ненулевой символ в строке.

Третий способ позиционирования курсора в строке - это поиск указанного символа на текущей строке. Если символ не найден на текущей строке, то появляется звуковой сигнал и курсор остается на прежнем месте. Существуют команды, которые проводят поиск по шаблону. Они будут описаны далее. Здесь приведем 6 команд для поиска внутри строки: f, F, t, T, ;(двоеточие) и ,(запятая). После f, F, t и T вы можете указать символ:

fx - осуществляет поиск указанного символа x справа от текущей позиции курсора;

Fx - осуществляет поиск указанного символа x слева от текущей позиции курсора;

tx - осуществляет перемещение курсора вправо на символ до указанного символа x;

Tx - осуществляет перемещение курсора влево на символ до указанного символа x;

; - повторяет предыдущий поиск символа. Команда ; запоминает символ и ищет следующее появление символа в текущей строке;

, - повторяет предыдущий поиск символа, но в обратном направлении. Команда ; запоминает символ и ищет следующее появление символа в текущей строке;

Команда - (минус) перемещает курсор на одну строку вверх, позиционируя его на первом ненулевом символе. Чтобы переместить курсор более чем на один символ, укажите пред знаком минус количество строк. Например, чтобы переместить курсор вверх на 13 строк, введите: 13-

Если эти строки находятся выше отраженного экрана, то окно переместится так, чтобы открыть их. Этот способ помогает быстро перемещаться по файлу. Если выше окажется меньше 13-ти строк, то звуковой сигнал подскажет вам, что вы сделали ошибку и курсор останется в своем прежнем положении.

Команда + (плюс) или <CR> перемещает курсор вниз на одну строку и позиционирует на первый непустой символ. Перед командой + можно указать количество строк, на которое вы хотите переместить курсор. Например, чтобы переместить курсор на 9 строк вниз, введите: 9+

Если некоторые строки расположены ниже текущего экрана, то окно переместится вниз, чтобы открыть их.

#### *6.2. Позиционирование слов*

Редактор vi предполагает, что слово - это строка символов, которая может включать в себя буквы, цифры и символы подчеркивания. В редакторе vi существует 6 команд позиционирования: w, b, e, W, B и E. Команды w, b, e рассматривают любой символ, отличный от буквы, цифры или подчеркивания, как разделитель, означающий начало или конец слова. Пунктуация до или после пустого пространства является разделителем слов. Символ начала или конца строки также является разделителем.

Команды W, B, E рассматривают пунктуатию как часть слова. Слова разделяются пустым пространством, которое состоит из символов пробел, табуляция и новая строка.

Команды позиционирования:

w - перемещает курсор к началу следующего слова;

nw - перемещает курсор к началу n-го слова; конец строки не останавливает движение курсора. Он продолжает подсчет слов с начала следующей строки;

W - игнорирует все знаки пунктуации и перемещает курсор на следующее после пробела слово;

e - перемещает курсор на последний символ в следующем слове;

E - перемещает курсор на последний символ в слове. Игнорирует все символы пунктуации за исключением пробела. Слова разделяются пробелами;

b - перемещает курсор к первому символу предыдущего слова;

nb - перемещает курсор к первому символу n-го предыдущего слова. Команда b не останавливается в начале строки и продолжает движение к началу строки;

B - используется подобно команде b, за исключением того, что слова разделяются только символами пробел и новая строка. Она рассматривает все другие символы пунктуации как буквы.

#### *6.3. Позиционирование курсора на предложение*

Редактор vi также распознает предложения. В редакторе vi предложение заканчивается знаками !, . или ?. Если эти разделители появляются в середине строки, то за ними должны стоять два пробела, чтобы vi их мог распознать.

Вы можете перемещать курсор от предложения к предложению в файле с помощью команд ( - открывающая скобка и ) - закрывающая скобка.

( - перемещает курсор в начало текущего предложения;

n( - перемещает курсор в начало n-го предложения выше текущего предложения;

) - перемещает курсор в начало следующего предложения;

n) - перемещает курсор в начало n-го предложения, следующего за текущим предложением.

#### *6.4. Позиционирование курсора по параграфам*

Параграфы распознаются редактором vi, если они начинаются после пустой строки. Если вы хотите перемещать курсор по параграфам, то при создании текста не забудьте в конце каждого параграфа поставить пустую строку.

{ - переместить курсор в начало текущего параграфа, который ограничивается пустой строкой;

n{ - переместить курсор в начало n-го параграфа, следующего за текущим параграфом;

} - переместить курсор в начало следующего параграфа;

n} - переместить курсор в начало n-го параграфа ниже текущей строки.

#### *6.5. Позиционирование в окне*

Редактор vi предоставляет вам 3 команды, которые позволяют вам позиционировать курсор в окне. Эти команды должны быть введены на верхнем регистре:

H - переместить курсор к первой строке экрана;   
M - переместить курсор к средней строке экрана;   
L - переместить курсор к последней строке экрана.

### 7. Позиционирование курсора на неотображаемом тексте

В этом подразделе описывается, как переместить курсор на текст, который не отображен на текущем экране. Вы можете сделать это следующим образом:

* прокрутка вперед и назад файла;
* перейти к указанной строке в файле;
* поиск по шаблону в файле.

Четыре команды редактора vi позволяют вам осуществлять прокрутку текста файла. Команды ^f (<CTRL>f) и ^d (<CTRL>d) проводят прокрутку экрана вперед, а команды ^b (<CTRL>b) и ^u (<CTRL>u) - прокрутку экрана назад.

Команда ^f осуществляет прокрутку вперед на одно окно текста, расположенного ниже текущего окна. Редактор vi очищает экран и выводит новое окно. Две последние строки текущего окна отображаются в верхней части нового окна. Если окно дополнено пустыми строками, то они помечаются знаком ~ (тильда)

Команда ^d осуществляет прокрутку на половину экрана, чтобы отобразить текст, расположенный ниже окна. Если курсор находится на последней строке файла и вы нажмете ^d, то vi подаст звуковой сигнал.

Команда ^b осуществляет прокрутку экрана назад на полное окно, чтобы отобразить текст, который находится выше текущего экрана. Редактор vi очищает экран и отображает новое окно. В отличие от команды ^f, команда ^b не оставляет в новом окне строки из текущего окна.

Команда ^u осуществляет прокрутку половины экрана, чтобы отобразить текст, расположенный выше окна. Если курсор достигнет начала файла, то vi подаст звуковой сигнал.

### 8. Переход на указанную строку

Команда G позиционирует курсор на указанную строку в окне. Если эта строка не находится в текущем экране, то G очищает экран и отображает окно, содержащее эту строку. Если не указана строка, то команда G перемещает курсор на последнюю строку файла:   
G - переместить курсор на последнюю строку в файле;   
nG - переместить курсор на строку n.

### 9. Номера строк

Каждая строка файла имеет номер, соответствующий ее положению в буфере. Чтобы получить номер конкретной строки, переместите курсор на эту строку и введите команду: ^g

В нижней части экрана команда ^g поместит информацию:

* имя файла;
* был ли буфер модифицирован после поледней записи его в файл;
* номер строки, на которой находится курсор;
* общее количество строк в буфере.

### 10. Поиск символов по шаблону: команды / и ?

Для поиска символов по шаблону вы можете использовать команды /, ?, n и N. Эти команды позволяют осуществлять поиск вперед и назад в буфере следующего появления указанного шаблона символов. Команды n и N позволяют вам повторить последний поиск.

Команда / и следующий за ней шаблон (/pattern) ведет поиск вперед по буферу следующего появления символов, указанных в шаблоне и помещает курсор на первый символ из шаблона. Например, командная строка:

/Hello world<CR>

находит первое появление в буфере слов Hello world и помещает курсор под буквой H.

Команда ? и следующий за ней шаблон (?pattern) ведет поиск в обратном направлении в буфере первого появления символов, указанных в шаблоне и помещает курсор на первый из этих символов. Например, командная строка:

?data set design<CR>

находит предыдущее появление в буфере слов data set design и помещает курсор под буквой d в слове data.

Эти команды поиска не выполняют циклически возврат в случае окончания строки во время поиска двух или более слов. Например, если вам надо найти появление Hello world и окажется, что слово Hello находится в конце одной строки, а слово world - первое слово последующей строки, то эти команды не найдут Hello world.

В то же время, эти команды поиска выполняют циклический возврат в конце или начале буфера для продолжения поиска. Например, если вы находитесь почти в конце буфера, а символы, которые соответствуют шаблону в начале буфера, то команда поиска / найдет шаблон.

Команды n и N позволяют повторить предыдущий запрос поиска /pattern или ?pattern:   
n - повторить последнюю команду поиска;   
N - повторить последнюю команду поиска в обратном направлении.

Команды / и ? не дают вам возможности указать конкретное появление шаблона с помощью числа. То есть, вы не можете, например, запросить поиск 3-го появления шаблона, начиная с текущей позиции.

### 11. Создание текста

Три основные команды позволяют в редакторе vi создавать текст:   
a - добавить текст;   
i - вставить текст;   
o - открыть новую строку, на которой может быть введен текст.

После того, как вы создали текст с помощью одной из этих команд, можете возвратиться в командный режим редактора vi нажав клавишу ESCAPE. Чтобы в файл добавить текст, воспользуйтесь следующими командами:   
a - добавить текст после курсора;   
A - добавить текст в конце текущей строки.

Чтобы в файл вставить текст, воспользуйтесь следующими командами:   
i - вставить текст до курсора;   
I - вставить текст в начале текущей строки до первого ненулевого символа.

Для возврата в командный режим, нажмите клавишу ESCAPE.

Для открытия строки для текста воспользуйтесь следующими командами:   
o - создать текст с начала новой строки, расположенной ниже текущей строки. Вы можете ввести эту команду в любом месте текущей строки;   
O - создать текст с начала новой строки, расположенной выше текущей строки. Вы можете ввести эту команду в любом месте текущей строки.

Команды открытия создают строку непосредственно выше или ниже текущей строки и переводят вас в режим ввода текста. Курсор располагается в начале новой строки и ожидает от вас ввода текста.

### 12. Удаление текста

Вы можете удалить текст с помощью нескольких команд в командном режиме и небольшой объем в режиме ввода.

#### *12.1. Удаление текста в режиме ввода текста*

Если вы находитесь в режиме ввода текста редактора vi и хотите удалить символ, воспользуйтесь клавишей BACKSPACE: BACKSPACE - удалить текущий символ (слева от курсора).

Клавиша BACKSPACE перемещает курсор назад в режиме ввода текста и удаляет каждый символ, который курсор переходит. Однако удаленный символ не стирается с экрана пока вы не нажмете клавишу ESCAPE для возврата в командный режим.

Удалить текст вы можете и с помощью следующих ключей:   
^w - отменить ввод текущего слова;   
@ - удалить на текущей строке весь введенный во время данного режима ввода текст;   
^v - удалить специальное значение, если есть, следующих вводимых символов.

Когда вы вводите ^w, курсор возвращается на последнее введенное слово и ожидает на первом символе этого слова. Он не стирает слово с экрана до тех пор, пока вы не нажмете клавишу ESCAPE или не введете новые символы на месте этого слова.

#### *12.2. Отменить последнюю команду*

Отменить последнюю введенную команду можно с помощью следующих команд:   
u - отменить последнюю команду;   
U - восстановить текущую строку в состояние до произведенных изменений.

Если вы случайно удалили строки, то для отмены команды удаления введите команду u. Удаленные строки вновь появятся на экране. Если вы ввели неправильно команду, то введите команду u и неверная команда будет отменена. Команда U будет аннулировать все изменения в текущей строке до тех пор, пока курсор будет находиться на этой строке.

Если вы введете команду u дважды, то вторая команда отменит первую. Например, если вы удалили по ошибке строку и восстановили ее с помощью команды u, то нажав второй раз u, вы вновь удалите строку. Знание этих команд может спасти вас от многих неприятностей.

### 13. Команды удаления в командном режиме

Многие команды редактора vi, такие как команды удаления и изменения, предоставляют вам возможность использовать в командной строке команды перемещения курсора. Команды перемещения курсора могут указывать на объект текста: слово, строка, предложение, параграф. В этом случае общий формат команды будет такой:

[number][command]text\_object

Квадратные скобки указывают, что данные компоненты не являются обязательными.

Все команды удаления, выдаваемые в командном режиме сразу же удаляют указанный текст с экрана и оставшуюся часть отображают на экране в обновленном виде.

Команда удаления имеет следующий формат:

[number]dtext\_object

где d - команда удаления;   
text\_object - объект удаления.

#### *13.1. Удаление слов*

Вы можете удалить слово или часть слова с помощью команды d. Переместите курсор к первому удаляемому символу и введите команду dw. Символ, расположенный над курсором и последующие за нми будут стерты. Команда dw удаляет одно слово или знак пунктуации и пробелы, которые следуют за ним. Вы можете удалить несколько слов или знаков пунктуации за один раз, если укажете перед командой соответствующее число. Например, чтобы удалить 3 слова и две запятые, введите команду 5dw.

#### *13.2. Удаление параграфов*

Чтобы удалить параграф, воспользуйтесь командой: d{ или d}

Помните, что вы можете восстановить удаленный текст с помощью команды u.

#### *13.3. Удаление строк*

Чтобы удалить строку, введите: dd

Для удаления нескольких строк укажите перед командой количество удаляемых строк. Например, команда: 10dd сотрет 10 строк.

Если вы удаляете несколько строк, то vi отображает в нижней части экрана сообщение типа: 10 lines deleted

Если ниже текущей строки осталось меньше 10 строк, то прозвучит сигнал и строки не удалятся.

#### *13.4. Удаление текста после курсора*

Для удаления всего текста на строке, расположенного после курсора, поместите курсор на первый удаляемый символ и введите: D или d$

Эта команда удаляет текст только на текущей строке.

### 14. Модификация текста

Один из способов модификации текста представляют команды удаления и команды ввода текста. Другой способ, с помощью которого вы можете изменить текст, это задать команду, позволяющую одновременно удалять и создавать текст. В редакторе vi существуют 3 основные команды изменения текста: r, s и c.

#### *14.1. Замена текста*

rx - Заменить текущий символ (символ, на который показывает курсор) на x. Эта команда не устанавливает режим ввода текста и, следовательно, нет необходимости нажимать после нее клавишу ESCAPE.

nrx - Заменить n символов на x. После этой команды также не надо нажимать клавишу ESCAPE.

R - Заменяет только те символы, которые введены в режиме ESCAPE. Если будет достигнут конец строки, то эта команда добавит ввод как новый текст.

Например, вы хотите заменить слово acts на ants в следующем предложении:

The circus has many acts.

Поместите курсор под буквой c в слове acts и введите команду:

rn

Предложение будет иметь вид:

The circus has many ants.

Чтобы заменить слово many на 7777, поместите курсор под буквой m в слове many и введите команду:

4r7

Команда r заменит слово many на четыре цифры 7:

The circus has 7777 ants.

#### *14.2. Подстановка текста*

Команда подстановки текста заменяет символы, но затем позволяет вам продолжать ввод текста с того места, где вы нажали клавишу ESCAPE.

s - удалить символ, на который указывает курсор и добавить текст. После окончания ввода текста нажмите клавишу ESCAPE;

ns - удалить n символов и добавить текст. После окончания ввода текста нажмите клавишу ESCAPE;

S - заменить все символы в строке.

Когда вы вводите команду s, последний символ в строке символов, который будет заменен, переписывается со знаком $. Символ не стирается с экрана, пока вы не введете на его место новый символ, или не нажмете клавишу ESCAPE.

#### *14.3. Изменение текста*

Команда подстановки заменяет символы. Команда изменения заменяет объект текста и затем продолжает добавлять текст с этого места до тех пор, пока вы не нажмете клавишу ESCAPE.

Команда изменения может содержать аргумент text\_object. Вы можете заменить символ, слово и так далее на новый текст:

ncx - заменить n-ый объект текста типа x (предложение, параграф);

cw - заменить слово или оставшиеся символы в слове на новый текст. Редактор vi печатает знак $, чтобы показать последний символ, который будет заменен;

ncw - заменить n слов;

cc - заменить все символы в строке;

ncc - заменить все символы в текущей строке и n верхних строк;

C - заменить оставшиеся символы в строке от курсора до конца строки; dfn nC заменить оставшиеся символы, начиная от курсора в текущей строке и во всех n верхних строках.

Команда "c" использует знак $, чтобы отметить последний символ, который должен быть замещен. После того как выполнилась команда изменения, вы находитесь в режиме ввода текста и можете вводить любой текст. Тескт будет храниться в буфере, пока вы не нажмете клавишу ESCAPE.

### 15. Извлечение и присоединение текста

Редактор vi предоставляет набор команд, которые разбивают текст в файл и набор команд, которые копируют часть текста и помещают в другое место в файле.

#### *15.1. Перемещение текста*

Вы можете переместить текст из одного места в буфере редактора vi в другое, удалив строки, а затем поместив их в требуемое место. Удаленный текст сохраняется во временном буфере. Переместите курсор на то место, куда вы хотите поместить текст и нажмите клавишу "p". Удаленные строки будут добавлены ниже текущей строки.

p - поместить содержимое временного буфера после курсора или ниже текущей строки.

Удаленная с помощью команды удаления строка может быть помещена в середине другой строки. Позиционируйте курсор между двумя словами, затем нажмите клавишу "p". Удаленная строка будет помещена после курсора.

Так как временный буфер сохраняет результат только последней команды, то команда "p" должна использоваться сразу же после команды удаления. Команда "p" используется также для копирования текста, помещенного во временный буфер командой "y"

#### *15.2. Перемещение букв*

Быстрым способом перемещения букв является комбинация команд "x" и "p" (xp). Команда "x" удаляет букву, "p" - помещает ее после следующего символа. Например, в следующем предложении есть ошибка:

A line of tetx

Поместите курсор под буквой "t" в комбинации "tx" и затем нажмите клавиши "x" и "p". В результате получим:

A line of text

#### *15.3. Копирование текста*

Вы можете копировать одну или несколько строк текста во временный буфер и затем поместить копию этого текста в определенное место в файле. Чтобы поместить текст на новое место, введите команду "y". Команда "y" имеет следующий общий формат:

[number]y[text\_object]

Копирование строк с помощью команды "y" не удаляет их с прежнего места в файле, т.е. вы можете поместить один и тот же кусок текста в нескольких местах.

Обратите внимание, что эта команда позволяет указывать номер объекта текста, который будет копироваться.

### 16. Копирование и перемещение текста с использованием регистров

Редактор vi предоставляет набор команд, которые разбивают текст в файл и набор команд, которые копируют часть текста и помещают в другое место в файле.

Перемещение и копирование нескольких разделов текста в различные части файла является утомительной работой. Редактор vi предлагает вам использовать так называемые регистры, в которых вы можете хранить текст до тех пор, пока он вам не понадобится. Сохранить текст вы можете либо с помощью команды "y", либо с помощью команды удаления. Использование регистров особенно полезно, если часть текста должна появиться во многих местах в файле. Извлекаемый текст остается в указанном регистре пока вы не завершите работу с ним, либо закончите работу с редактором, либо поместите в этот регистр другой текст.

Общий формат команды:

[number]["X]command[text\_object]

где X - имя регистра, состоящее из одной буквы верхнего регистра. Имени регистра должны предшествовать двойные кавычки.

### 17. Специальные команды

В этом подразделе приведены специальные команды, которые будут вам весьма полезны:   
. - повторить последнюю команду;   
j - объединить две строки;   
^l - очистить экран и вновь отобразить его;   
~ - заменить нижний регистр на верхний и vice versa (?).

#### *17.1. Повторить последнюю команду*

Команда "." (точка) повторяет последнюю введенную команду. Очень часто она используется с командой поиска.

#### *17.2. Объединение строк*

Команда "j" объединяет строки. Чтобы ввести эту команду, поместите курсор на текущей строке и нажмите клавиши SHIFT и j одновременно. Текущая строка объединяется со следующей строкой.

Например, вы имеете две строки текста:

Dear Mr.

Smith:

Объединим эти две строки в одну. Поместите курсор под любым символом в первой строке и введите команду "j". На экране появится строка:

Dear Mr. Smith:

Обратите внимание, что редактор vi автоматически ставит пробел между последним словом в первой строке и первым словом второй строки.

#### *17.3. Очистить окно и восстановить текст*

Если другой пользователь системы UNIX посылает вам собщение с использованием команды write, а вы в это время редактируете с помощью vi, то посланное сообщение появится в вашем текущем окне поверх текста, который вы редактируете. Чтобы восстановить редактируемый текст после того, как вы прочтете посланное сообщение, нужно перейти в командный режим. Если вы находитесь в режиме ввода текста, то нажмите "^l". Редактор vi сотрет сообщение и восстановит окно в том виде, в котором оно было до прихода сообщения.

#### *17.4. Заменить нижний регистр на верхний и vice versa*

Чтобы быстро заменить букву нижнего регистра на букву верхнего регистра, надо поместить курсор под этой буквой и нажать клавишу "~" (тильда). Вы можете заменить несколько букв, нажав столько же раз клавишу "~", либо поставив перед тильдой число требуемых замен.

**18. Использование команд построчного редактора**

Редактор vi имеет доступ ко многим командам построчного редактора ex. В этом подразделе обсуждаются некоторые наиболее часто используемые команды.

Команды редактора ex очень похожи на команды редактора ed.

Команды построчного редактора начинаются с символа ":" (двоеточие). После того, как вы введете двоеточие, курсор опускается в низ экрана и отображает двоеточие. Оставшаяся часть команды тоже появляется внизу экрана.

#### *18.1. Временный возврат к shell*

Когда вы входите в vi, то содержимое буфера заполняет ваш экран, делая невозможным выдачу команд для shell. Однако вам может это понадобиться. Например, вам необходимо будет получить информацию из другого файла для присоединения ее к текущему тексту. Вы можете получить эту информацию, запустив одну из команд shell, которая отображает текст файла на экране. Например, cat или pg. Однако процедура завершения работы с редактором и повторный вход в редактор потребуют времени и сил. Редактор vi предлагает вам два способа временного выхода из редактора для того, чтобы вы могли ввести команды shell (и даже редактировать другие файлы) не покидая сам редактор. Это команды :sh и :!.

Команда :! позволяет вам выйти из редактора и запустить команду shell одной командной строкой. Находясь в командном режиме редактора vi, введите :!. Эти символы будут напечатаны в низу экрана. Сразу же после символа ! введите команду shell. Shell запустит вашу команду, даст вам ответ и напечатает сообщение:

Hit return to continue

После того, как вы нажмете клавишу RETURN, vi обновит экран и курсор примет свое прежнее положение.

Команда редактора :sh позволяет вам делать то же самое, но ведет по другому себя на экране. Из командного режима редактора vi введите :sh и нажмите клавишу RETURN. Подсказка команды shell появится на следующей строке. Ведите ваши команды после подсказки так, будто бы вы работаете в нормальном режиме с shell. Когда вы будете готовы вернуться к редактору vi, введите ^d или exit. Экран обновит информация из вашего буфера и курсор вновь переместится на прежнее место.

При временном переходе к shell даже изменение справочников не мешает возврату к буферу редактора vi при вводе ^d или exit.

#### *18.2. Запись текста в новый файл*

Команда "w" позволяет вам создавать файл, копируя строки текста из файла, который вы в данный момент редактируете, в файл, который вы указали. Чтобы создать новый файл нужно указать строку или строки (номера строк) и имя нового файла в командной строке. Общий формат этой команды:

:line\_number[, line\_number]w filename

Например, чтобы записать третью строку буфера в файл three, введите:

:3w three<CR>

При успешном создании нового файла, редактор vi напечатает следующую информацию:

"three" [New file] 1 line, 20 characters

Чтобы записать текущую строку в файл, укажите . (точку) как адрес строки:

:.w junk<CR>

Создастся новый файл с именем junk. Он будет содержать только текущую строку буфера vi.

Вы можете записать часть буфера в новый файл, обозначив границы строк. Например, чтобы строки с 23-й по 37-ю записать в файл, введите следующую команду:

:23,37w newfile<CR>

#### *18.3. Переход к указанной строке*

Вы можете переместить курсор к любой строке в буфере, если введете команду:

:n<CR>

где n - номер строки в буфере.

#### *18.4. Удалить остаток буфера*

Одним из самых простых способов удаления всех строк от текущей строки и до конца буфера является использование команды "d" построчного редактора со специальнымим символами для текущей и последней строк:

:.,$d<CR>

где .(точка) - обозначает текущую строку;   
$ - обозначает последнюю строку.

#### *18.5. Добавить файл в буфер*

Чтобы добавить текст из файла ниже указанной строки в буфер редактора, воспользуйтесь командой "r". Например, чтобы перенести содержимое файла data в ваш текущий файл, поместите курсор на то место, где должен стоять текст и введите команду:

:r data<CR>

Вместо перемещения курсора вы можете указать номер строки. Например, чтобы вставить файл data после 56-й строки в буфере, введите:

:56r data<CR>

#### *18.6. Провести глобальные изменения*

:g/pattern/command<CR> -

для каждой строки, содержащей pattern выполнить команду command редактора ex. Построчный редактор напечатает све строки, сождержащие pattern;

:s/pattern/new\_words<CR> -

это команда замены. Построчный редактор отыскивает первое появление символов pattern в текущей строке и заменяет его на new\_words;

:s/pattern/new\_words/g<CR> -

редактор ex заменяет каждое появление pattern в текущей строке;

:g/helix/s//double helix/g<CR> -

эта команда осуществляет поиск слова helix. Каждое найденное слово helix заменяется на double helix. Ограничители после команды "s" позволяют не вводить слово helix каждый раз. Команда запоминает слово между ограничителями после команды "g".

### 19. Завершение работы с редактором vi

Пять основных команд могут быть использованы для завершения работы с редактором vi. Команды, которым предшествует ":", являются командами построчного редактора:

:wq<CR> - записать содержимое буфера vi в файл, который в данный момент редактируется и завершить работу с vi;

ZZ - записать содержимое буфера, если он изменялся с момента последней записи;

:q<CR> - записать временный буфер в новый файл filename и завершить работу с редактором vi;

:q!<CR> - завершить работу с редактором vi без записи содержимого буфера в файл; не учитываются все изменения, проведенные в буфере;

:q<CR> - завершить работу с редактором vi без записи содержимого буфера в файл. Эта команда работает в том случае, если вы не производили изменений в буфере. В противном случае vi выдаст предупреждающее сообщение о том, что вы должны сохранить буфер или воспользоваться для завершения работы с vi командой :q!<CR>. Что нужно сделать, чтобы дать файлу другое имя?

Например, вы хотите записать в новый файл junk. Введите команду:

:w junk<CR>

Затем завершите работу с vi:

:q<CR>

Если вы попытаетесь записать в уже существующий файл, то получите предупреждающее сообщение. Например, если вы попытаетесь записать в файл johnson, система выдаст сообщение:

"johnson" File exists - use "w!johnson"

to overwrite

Если вы хотите заменить содержимое существующего файла на содержимое буфера, то воспользуйтесь командой ":w!":

:w! johnson<CR>

### 20. Специальные опции для vi

Команда vi имеет специальные опции. Они позволяют:

* восстановить потерянный файл, если работа vi была прервана;
* поместить в буфер редактирования несколько файлов и редактировать их по порядку;
* просмотр файла без риска случайного внесения изменений в него.

#### *20.1. Восстановление потерянного файла*

Если случайно произошло прерывание работы редактора vi, то не производится запись текста из буфера обратно в файл. Но vi сохраняет копию буфера. Когда вы вновь зарегистрируетесь в системе UNIX, вы можете восстановить файл с помощью опции -r. Введите:

vi -r filename<CR>

Все или почти все изменения, которые вы выполнили до прерывания, вновь появятся в буфере vi, и вы снова сможете продолжить редактирование файла или записать буфер в файл и завершить работу с vi.

#### *20.2. Редактирование нескольких файлов*

Если вы хотите редактировать в одном сеансе более одного файла, то введите команду vi, указав имя каждого файла:

vi file1 file2<CR>

vi ответит, сколько файлов вы будете редактировать.

Например:

2 files to edit

После того, как вы отредактировали первый файл, запишите все изменения из буфера в файл file1:

:w<CR>

Система ответит на эту команду сообщением в низу экрана. В нем она напечатает имя файла, число строк и символов в этом файле. Затем вы можете начать редактировать следующий файл, задав команду:

:n<CR>

Система в низу экрана сообщит вам имя следующего файла, который вы будете редактировать, и число строк и символов в этом файле.

#### *20.3. Просмотр файла*

Один из способов просмотра файла, это воспользоваться командами прокрутки и поиска. Однако вы можете защитить себя от случайного внесения изменений в файл во время сеанса редактирования. Чтобы воспользоваться опцией "только чтение", вызовите редактор следующим образом:

view file<CR>

#### 

#### 21.Другие текстовые редакторы Unix

#### *21.1. Редактор ED.*

Редактор текстов ed является диалоговой (интерактивной) программой, которая позволяет добавлять, перемещать, удалять текст и извлекать его из других файлов. Имеется возможность замены слова или части слова в тексте как для одной указанной строки, так и для части файла или всего файла. Кроме того, можно просмотреть текстовый файл, напечатать его частично или целиком, напечатать строки, содержащие определенное слово и т.д.

ED работает с копией обрабатываемого файла, которая создается в буфере редактора и хранится там в течение всей обработки. Это предохраняет исходный файл от возможных разрушений при совершении грубых ошибок. Сделанные в копии изменения не влияют на исходный файл до тех пор, пока не будет выдана команда записи.

Редактор запоминает имя обрабатываемого файла (первоначально - имя исходного загруженного файла). По желанию пользователя это имя можно заменять для получения новых вариантов файла.

Если размер файла превышает размер буфера редактора, файл приходится делить на части с помощью команды split, редактировать его по частям, а затем снова объединять с помощью команды cat.

Пользователь сообщает редактору, что делать с его текстом, с помощью определенных инструкций, называемых "командами". Команды редактора имеют простую структуру, и в большинстве случаев состоят из одной буквы. Каждая команда печатается на отдельной строке. Иногда перед командой указывается информация о том, какая строка или строки должны подвергаться действию команды. ed не отвечает на большинство команд, не печатает подсказки или сообщения типа "готово".

Все вводимые командные строки должны заканчиваться символом <CR>.

***21.2. GNU Emacs 19.19***

Говоря о текстовых редакторах, разработанных специально под UNIX, в первую очередь упомянем Emacs, который распространяется бесплатно в рамках проекта GNU. Некоторые компании распространяют его на коммерческой основе, исправив имеющиеся ошибки и адаптировав под конкретные платформы с некоторыми улучшениями возможностей "родного" Emacs. Таковым, например, является GNU Emacs 19.19 фирмы Ready-to-Run Software, который распространяется в составе пакета утилит обработки текста. Безусловно, данный редактор предназначен для квалифицированных программистов, обладая мощными средствами разработки программ. Однако GNU Emacs требует основательного изучения; для расширения его возможностей приходится программировать на языке LISP, что не очень нравится даже квалифицированным программистам, а уж обыкновенным пользователям - тем более. Для поклонников Emacs могу сообщить, что исходные тексты версии 19.28 появились на prep.ai.mit.edu: в каталоге pub/gnu.

***21.3. UniPress Emacs 2.20e***

Немногим отличается от собрата UniPress Emacs 2.20e, выпущенный компанией UniPress Software. В отличие от GNU Emacs данный редактор поставляется с документацией (хотя и неудачно написанной), обладает более привлекательным интерфейсом, имеет поставляемые за дополнительную плату (от 695 до 995 долларов) языко-ориентированные реализации для С, С++, Fortran и Ada. В целом, различия между GNU Emacs и UniPress Emacs незначительны. Если Вы являетесь поклонником Emacs, то, несомненно, будете удовлетворены обеими реализациями.

***21.4. Siren Editor 1.0***

Siren Editor разработан компанией Siren Software, целиком основан на интерфейсе X/Motif и является интуитивным и легким в использовании. Все команды располагаются в выпадающих меню, хорошо продумана компоновка экрана и работа с мышью. Инсталляция и запуск редактора под силу даже неквалифицированному пользователю. Недостатком редактора Siren является то, что в нем отсутствует какая-либо поддержка процесса программирования, а также базовые средства форматирования текста. Поэтому Siren Editor может быть полезен, главным образом, при первичном наборе и минимальном редактировании текстов. По своей функциональности этот редактор чем-то напоминает NortonEdit из MS-DOS, который очень удобен, когда в текст надо быстро внести пару-другую изменений.

***21.5. Iris Х Editor 1.3.7***

Редактор Iris Х Editor (разработка Iris Computing Laboratory) предназначен для программистов и имеет собственный интерфейс под Х Window. При запуске Вы увидите чистый экран - ни меню, ни строки статуса - т.е. все нацелено на то, чтобы отобразить на экране как можно больше исходного кода программы. Информация всегда появляется во всплывающих окнах, что поначалу немного раздражает, но потом привыкаешь. Для работы с блоками текста предусмотрено четыре буфера временного хранения (clipboard). Вставку текста из такого буфера можно произвести одним нажатием кнопки мыши, что очень удобно. Недостатком редактора является полное отсутствие базовых средств форматирования.

***21.6. Nedit 3.1.1***

Этот редактор разработан в Национальной Лаборатории Ферми, распространяется свободно в исходных текстах, имеет стопроцентный графический пользовательский интерфейс под Motif и рассчитан прежде всего на программистов и случайных пользователей. Nedit чрезвычайно прост в установке, изучении и использовании, обладает исчерпывающей Help-системой. И в то же время в нем отсутствует много возможностей, которые являются стандартом для профессиональных текстовых редакторов (макро, прямоугольные блоки, базовые средства форматирования). Nedit имеет реализации практически для любых рабочих станций с UNIX (SGI, Sun, HP, DEC, IBM), а также для VAX/VMS. Перенос на другие платформы с X/Motif не вызывает никаких проблем.

***21.7. TED 2.la***

Редактор TED разработан английской фирмой Eagle Dynamics, имеющей представительство в Москве, и на сегодняшний день является единственным полностью русифицированным профессиональным текстовым редактором для UNIX, распространяемым в России. Редактор работает как на текстовых терминалах, так и под Х Windows, при этом CUA-интерфейс делает его похожим на редактор MultiEdit для MS-DOS или среду разработки Borland IDE. Это единственный из всех редакторов, у которого имеется документация на русском языке (два тома). TED ориентирован как на профессиональных разработчиков, так и на неподготовленных пользователей, может быть легко перенастроен в соответствии с индивидуальными вкусами. Редактор распространяется практически на любых платформах UNIX. Единственным, но несущественным, по мнению автора, недостатком редактора является отсутствие развитого макроязыка, что, впрочем, компенсируется возможностью создания командных и клавишных макро.

***21.8. AUIS ez***

В рамках распространяемого бесплатно в исходных текстах для Х11 многофункционального проекта AUIS (The Andrew User Interface System), разработанного Консорциумом Эндрю (Andrew Consortium), имеется текстовый редактор ez с модулями поддержки языков программирования С, С++, Modula-2, и PL/MI. Этот редактор будет интересен разработчикам программного обеспечения благодаря наличию средств цветового и шрифтового выделения синтаксиса, перехвата ошибок компиляции, встроенного набора стандартных языковых конструкций. Средства обработки текстов минимальные, но достаточные для подготовки простых документов. Исходные тексты имеются для большого числа платформ.

***21.9. VEDIT 2.30***

Редакторы семейства VEDIT разработаны фирмой Greenview Data и предназначены для использования в текстовом режиме под управлением различных операционных систем на платформе Intel (MS-DOS, QNX, XENIX, UNIX 286/386, FlexOS). Редактор целиком написан на ассемблере, что ограничивает его переносимость на другие аппаратные платформы, а также исключает возможность использования в рамках системы Х Window. Интерфейс VEDIT вполне современный (CUA-подобные меню, окна, интенсивное использование цвета, напоминает Borland IDE), функциональная раскладка клавиатуры достаточно интуитивна, возможна эмуляция некоторых известных редакторов и текстовых процессоров. VEDIT может редактировать любые файлы (в том числе двоичные, размером до 2 ГБайт), имеет удобный режим шестнадцатеричного редактирования. Отдельные реализации редактора включают макроязык, который, однако, чрезмерно мнемоничен, что затрудняет написание и понимание программ. Так, например, команда удаления текущей строки выглядит как "0КК". В общем, по своим возможностям VEDIT похож на ранее упомянутый редактор TED, однако поддержка процесса программирования и интерфейс с операционной системой здесь явно слабее. К тому же редактор, не русифицирован.

***21.10. Edit\*2000 1.40***

Компания Computer Innovations известна переносом собственных компиляторов для языков С и С++, а также средств разработки на различные платформы (MS-DOS, UnixWare, LynxOS, QNX и др.). Разработанный компанией редактор Edit\*2000 поставляется с хорошей документацией, достаточно легок в освоении и использовании, хотя имеет менее продуманный интерфейс, чем, скажем, Siren Editor. Назначение функциональных клавиш произведено на интуитивном уровне. В целом редактор может быть эффективно использован для проектов среднего уровня. В числе недостатков Edit\*2000 отсутствие макроязыка, некоторых базовых средств форматирования текста й программной поддержки (тэги, языковые шаблоны).

***21.11. SlickEdit 2. 3***

Редактор SlickEdit фирмы MicroEdge известен как "быстрый" редактор для разработчиков под MS-DOS. Однако его реализация под UNIX с точки зрения скорости и интерфейса оставляет желать лучшего. Более того, на некоторых из заявленных платформ редактор функционирует неустойчиво. Представляет интерес поддержка редктором таких языков программирования, как С, С++, Pascal, COBOL, dBase, Modula-3, Assembler. Макроязык имеет синтаксис, схожий с языком REXX, использующимся на мейнфреймах IBM и в OS/2. Рекомендовать SlickEdit можно лишь тем, кто имел с ним дело на других платформах.

***21.11. YUI 1.0 (бета-версия)***

Разработанный Инженерно-Технической Компанией в Ижевске текстовый редактор YUI распространяется в исполняемом виде как freeware и находится в стадии бета-тестирования. Заявлено, что бета-версии редактора имеются для платформ MS-DOS, BSD, Solaris, INTERACTIVE и Linux. Интерфейс реализован в стиле Turbo Vision, поэтому YUI внешне очень напоминает Borland С++ З.х. Однако назвать этот продукт редактором для разработчиков нельзя: в нем отсутствует какая-либо поддержка процесса программирования, да и базовые средства форматирования представлены не в полной мере. Продукт достаточно сырой и, по моему мнению, пока может представлять чисто академический интерес. Поэтому маловероятно, что YUI в ближайшее время привлечет внимание разработчиков и квалифицированных пользователей.

***21.12. Эмуляторы***

Эмуляторы редакторов из других операционных систем (как правило, с мэйнфреймов и суперминикомпьютеров) представляют интерес только для экс-программистов тех систем, которые перешли под UNIX и не желают привыкать к какому-либо другому текстовому редактору. Зачастую оказывается, что такой редактор уже морально устарел: новые редакторы используют новые технологии обработки текстов. Поэтому рекомендовать редактор-эмулятор новичку можно лишь с оглядкой на vi.

***21.13. nu/TPU 3.0***

Редактор nu/TPU компании a/Soft Development эмулирует известный редактор EVE, использующийся в операционной системе VAX/VMS. Программная поддержка в данном редакторе явно недостаточна для современных технологий разработки программ (отсутствует поиск ошибок компиляции, работа с тегами). Есть определенные проблемы с установкой nu/TPU (файлы описания клавиатуры имеют ошибки, инструкции по инсталляции иногда непонятны и даже неправильны), В то же время, руководство по использованию редактора написано в хорошем стиле. В целом, редактор может удовлетворить только бывших программистов VAX/VMS.

***21.14. REDT***

Редактор REDT разработан в Государственном Университете Вашингтона, базируется на библиотеке curses и следует модели редактирования, принятой в EDT - известном редакторе из VAX/VMS. Однако чистым эмулятором EDT этот редактор назвать нельзя: средства обработки текста в REDT выполнены в стиле редакторов для MS-DOS (выделение в прямоугольник, курсор может выходить за конец строки, наличие строки статуса). Раскладка функциональной клавиатуры может быть полностью изменена. REDT по достоинству оценят прежние пользователи редакторов EVE, EDT и SEDT.

**Список источников информации**

1. Джек Такет (мл.) и Стив Барнет. Использование Linux. Специальное издание.:, 5-е изд.: Пер.с англ.: Уч.пос. – М.: Издательский дом «Вильямс», 2000. – 183 с.

2. Сергей Ивановский. Операционная система Unix. Серия книг «Справочное руководство пользователя персонального компьютера» - М.: Познавательная книга плюс, 2000 – 512 с.

3. Internet.: www.unix.kg/rus/misc

4. Internet.: www.linuxshop.ru