Using vi, the Unix Visual Editor

Description

You will learn to use vi, the full screen editor found on nearly all Unix systems. Basic commands are covered, including those that enable you to insert, delete, change, replace, and copy text, and to move around within and between files. In addition, you will learn how to set editing options for your files, temporarily or permanently, globally or locally.

What You Will Learn

You will learn (1) how to start and end vi edit sessions, (2) how to move around in a file, (3) how to enter new text, (4) how to modify, move, and delete old text, and (5) how to read from and write to files other than the one you are editing.

Instructor

Rick Ells

Prerequisites

R105 or equivalent knowledge.

Table of Contents

A. VI BASICS

1. About vi

vi is Found on Nearly Every Unix Computer

- vi is the standard Unix editor
- Other documents on vi:
  - How To Use The vi Editor
  - Vi Reference
  - Mastering the VI editor - University of Hawaii at Manoa
  - Vi Lovers Home Page
  - Visual Editor (vi): A Tutorial - University of Southwestern Louisiana

vi is Powerful and Fast

- Your terminal displays a section of the file you are editing
- vi can do anything you want
- You don't need to remove your fingers from the standard typing keys-the keys themselves give commands to vi

vi Stays Out of Your Way

- vi has no menus
- vi commands are short

2. Starting vi

Open a file with vi. Type: **vi myfile.txt**

- If **myfile.txt** does not exist, a screen will appear with just a cursor at the top followed by tildes (~) in the first column.
- If **myfile.txt** does exist, the first few line of the file will appear.
- The status line at the bottom of your screen shows error messages and provides information and feedback, including the name of the file.

3. vi Modes

Command Mode

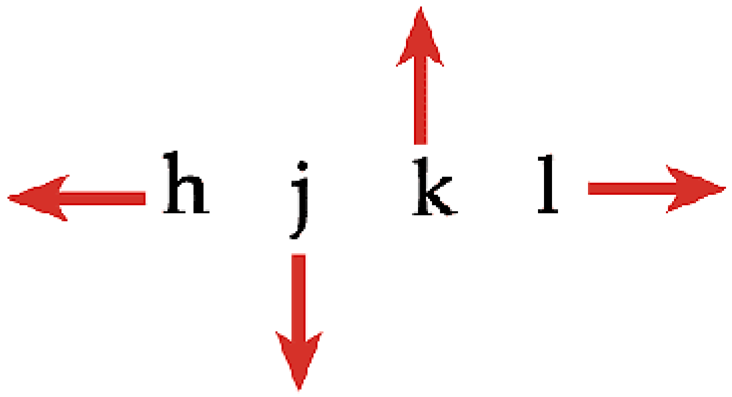
- Command mode is the mode you are in when you start (default mode)
- Command mode is the mode in which commands are given to move around in the file, to make changes, and to leave the file
- Commands are case sensitive: j not the same as J
- Most commands do not appear on the screen as you type them. Some commands will appear on the last line: : / ?

Insert (or Text) Mode

- The mode in which text is created. (You must press **<Return>** at the end of each line unless you've set wrap margin.)
- There is more than one way to get into insert mode but only one way to leave: return to command mode by pressing **<Esc>**

When in doubt about which mode you are in, press **<Esc>**

4. Basic Cursor Movement



From Command Mode

k	*Up one line*
j	*Down one line*
h	*Left one character*
l	*Right one character (or use **<Spacebar>**)*
w	*Right one word*
b	*Left one word*

NOTE: Many vi commands can take a leading count (e. g., 6k, 7e).

5. Entering, Deleting, and Changing Text

```
From Command Mode

i    Enter text entry mode

x    Delete a character

dd   Delete a line

r    Replace a character

R    Overwrite text, press <Esc> to end
```

6. Setting Basic Options in vi

Displaying Line Numbers

```
From Command Mode

:set nu    Display line numbers

:set nonu  Hide line numbers
```

Setting Right Margin

```
From Command Mode

:set wm=number  Set Wrap Margin number of spaces from right
                          edge of screen

:set wm=10      Set Wrap Margin 10 spaces from right edge
                          of screen

:set wm=0       Turn off Wrap Margin
```

7. Exiting vi

- To exit you must be in command mode-press **<Esc>** if you are not in command mode
- You must press **<Return>** after commands that begin with a : (colon)

```
From Command Mode

ZZ    Write (if there were changes), then quit
```

:wq	*Write, then quit*
:q	*Quit (will only work if file has not been changed)*
:q!	*Quit without saving changes to file*

8. Basics Summary

```
UNIX   ---> vi file --->  COMMAND  ---> i I a A o O --->  TEXT
SHELL  <---- ZZ <-------  MODE     <------ <Esc> <------  MODE
```

A Basic vi Session

1. To enter vi, type: **vi** *filename* **<Return>**
2. To enter insert mode, type: **i**
3. Type in the text: **This is easy.**
4. To leave insert mode and return to command mode, press: **<Esc>**
5. In command mode, save changes and exit vi by typing: **:wq <Return>**

   You are back at the Unix prompt.

B. INTERMEDIATE VI

1. More On Cursor Movement

From Command Mode

e	*Move to end of current word*
$	*Move to end of current line*
^	*Move to beginning of current line*
+	*Move to beginning of next line*
-	*Move to beginning of previous line*
G	*Go to last line of the file*
:*n*	*Go to line with this number (:10 goes to line 10)*
<Ctrl>d	*Scroll down one-half screen*
<Ctrl>u	*Scroll up one-half screen*
<Ctrl>f	*Scroll forward one full screen*

<Ctrl>b	*Scroll backward one full screen*
)	*Move to the next sentence*
(	*Move to the previous sentence*
}	*Move to the next paragraph*
{	*Move to the previous paragraph*
H	*Move to the top line of the screen*
M	*Move to the middle line of the screen*
L	*Move to the last line of the screen*
%	*Move to matching bracket: ({ [ ] })*

2. Entering Text Mode

From Command Mode

i	*Insert text before current character*
a	*Append text after current character*
I	*Begin text insertion at the beginning of a line*
A	*Append text at end of a line*
o	*Open a new line below current line*
O	*Open a new line above current line*

3. Commands and Objects

Format	**Example**
operator number object	**c2w**
number operator object	**2cw**

Operators

c	*change*
d	*delete*
y	*yank*

Objects and Locations

w	*one word forward*

b	*one word backward*
e	*end of word*
H, M, L	*top, middle, or bottom line on screen*
), (	*next sentence, previous sentence*
}, {	*next paragraph, previous paragraph*
^, $	*beginning of line, end of line*
/pattern/	*forward to pattern*

4. Replacing and Changing Text

From Command Mode

```
r       Replace only the character under the cursor.
               (Note: using r you remain in command mode.)

R       Beginning with the character under the cursor,
               replace as many characters on this line as you
               want. (You are in overtype mode until you
               press <Esc>

cw      Beginning with the character under the cursor,
               change a word to whatever you type.  (You are
               in insert mode until you press <Esc>)

c$      Beginning with the character under the cursor,
C              change a line to whatever you type. (You are
               in insert mode until you press <Esc>)
```

5. Deleting Text

From Command Mode

```
x       Delete a character

dw      Delete an alphabetic word and the following space
               (6dw deletes six words)

dW      Delete a blank-delimited word and the following space

dd      Delete a line (6dd deletes six lines)

d$      Delete all characters to the end of the line.
D

d}      Delete all characters to the end of the paragraph.

:5,30d  Delete lines 5 through 30
```

Deleted text goes into a temporary buffer that is replaced each time you delete (or copy) more text. The current contents of the buffer can be put back into your file.

6. Copying and Pasting Text

From Command Mode

yy	*Copy (yank) the current line*
6yy	*Copy (yank) six lines, beginning with the current line*
yw	*Copy the current word*
p	*Put the text after the cursor position*
P	*Put the text before the cursor position*

Copied text goes into a temporary buffer that is replaced each time you copy (or delete) more text. Only the current contents of the temporary buffer can be put back into your file. As a result, when you use copy (y), use the put (p) command immediately.

A yank and put procedure using colon commands:

1. **`:5,10y`** *Copy lines 5-10*
2. *Move cursor*
3. **`:put`** *Put after cursor*

7. Other Useful Commands

From Command Mode

.	*Repeat last command*
n.	*Repeat last command n number of times*
J	*Join next line to current line*
u	*Undo last single change*
U	*Restore current line*
~	*Change letter's case (capital to lower and vice versa)*

8. Buffers

Temporary Buffer

Deleted or copied text goes into a temporary unnamed buffer. The contents of the temporary buffer may be retrieved by using the **p** or **P** commands.

```
    p   Put words from temporary buffer after cursor or
               put lines from temporary buffer below current line

    P   Put words from temporary buffer before cursor or
               put lines from temporary buffer above current line
```

Lettered Buffers

There are 26 lettered buffers (a-z). Contents of a lettered buffer are saved until you copy or delete more characters into it, or until you quit your current vi session.

```
From Command Mode

"ayy     Copy (yank) a line into buffer a

"Ayy     Appends to buffer a

"a10yy   Copies 10 lines into buffer a

"a10dd   Deletes 10 lines of text into buffer a

"ap      Put contents of lettered buffer a below the current line
```

Both temporary and lettered buffers last only for the current vi session.

9. Copying, Deleting, or Moving Text Using Line Numbers

- These commands start with a colon (**:**) and end with a **<Return>** or **<Enter>**
- **<Ctrl>g** shows the line number of the current line
- The basic form of colon commands is

  :beginning_line, ending_line command destination

  where destination is the line after which you want the text placed.

```
From Command Mode

:5,10 co 105   Copy lines 5-10 to the line after 105

:5,20 m $      Move lines 5-20 to end of file

:7,300 d       Delete lines 7-300 (to buffer)
```

10. Searching for Text

```
From Command Mode

/text   Search forward (down) for text (text can include spaces
                    and characters with special meanings.)

?text   Search backward (up) for text
```

```
n       Repeat last search in the same direction

N       Repeat last search in the opposite direction

fchar   Search forward for a charcter on current line

Fchar   Search backward for a character on current line

;       Repeat last character search in the same direction

%       Find matching ( ), { }, or [ ]
```

11. Substitutions

The simplest way to do substitutions over a range of lines, or throughout the file, is to use the **s** colon command. The basic form of this command is the following:

```
    :n1,n2s/old/new/gc

    n1 is the beginning line

    n2 is the ending line number

    s means to substitute text matching the pattern (old)
             with text specified by (new)

    g (global) is optional.  It indicates you want to substitute
             all occurrences on the indicated lines.  If you use
             g, the editor substitutes only the first occurrence
             on the indicated lines.

    c (confirm) is optional.  It indicates you want to confirm
             each substitution before vi completes it.


From Command Mode

:%s/old/new/g     Substitutes old with new throughout the file

:.,$s/old/new/g   Substitutes old with new from the current
                         cursor position to the end of the file

:^,.s/old/new/g   Substitutes old with new from the beginning
                         of the file to the current cursor position

:&                 Repeats the last substitute (:s) command
```

C. ADVANCED VI

1. Writing to and Reading from Files

```
From Command Mode

:w file           Write current file to file
```

:w>>*file*	*Append current file to file*
:5,10w *file*	*Write lines 5 through 10 to file*
:5,10w>>*file*	*Append lines 5 through 10 to file*
:r *file*	*Read a copy of file into current file*
:!ls	*See a list of files in your current directory*

2. More About Options

From Command Mode-within vi for the current file only

:set all	*Display all options*
:set	*Display current settings of options*
:set nooption	*Unset option*
:set ai	*Set Auto Indentation during text entry*
:set ic	*Set Ignore Case during searches*
:set nu	*Show line Numbers*
:set sm	*Show Matching (or { when) or } is entered*
:set wm=10	*Set Wrap Margin 10 spaces from right edge of screen*

3. Customizing vi Sessions

Options can be set four ways:

1. During a vi session

```
:set nu
```

2. In a **.exrc** file in your home directory.

Sample contents of a .exrc file

```
set nu
set ai
set wm=10
```

3. In a **.exrc** file in a subdirectory.
4. By setting the **EXINIT** environmental variable.

Example of setting the EXINIT environmental variable

```
  setenv EXINIT "set nu ai ic"
```

On the Uniform Access systems (Homer, Saul, Mead, Alcott), the EXINIT environmental variable is used to set the shell within which the vi editor operates. Since the EXINIT environmental variable, if it has been defined, overrides anything set by a **.exrc** file, customizing vi on these computers requires redefining EXINIT. For example, to add numbering and auto indent, you would take the following steps:

1. Check to see what EXINIT is set to:

   ```
   % printenv EXINIT
     set shell=/bin/csh
   ```

   This response indicates that the shell is set to the C shell.

2. Reset EXINIT:

   ```
   % setenv EXINIT "$EXINIT nu ai"
   % printenv EXINIT
     set shell=/bin/csh nu ai
   ```

Order of Precedence

1. If a **.exrc** file exists in the current directory, vi reads it when beginning a session.
2. If no **.exrc** file exists in the current directory, vi checks the home directory for a **.exrc** file. If such a file exists, vi reads it when beginning a session.
3. If no **.exrc** file is found, vi uses its defaults.
4. Values set in the **EXINIT** environmental variable override any values set in a **.exrc** file.

---

4. Creating a .exrc File

1. At the system prompt, type: `vi .exrc`
2. Type the following commands, each on a separate line:

   ```
   set ai
   set ic
   set nu
   set wm=8
   ```

   Do not leave blank lines at the beginning or end of the **.exrc** file.

3. When you are finished, type: **<Esc> ZZ**

---

5. Abbreviations & Mapping

Abbreviations are text strings that automatically expand into larger strings during insert mode.

From Command Mode

```
:ab UW University of Washington
```

Mapping defines a single key to execute a sequence of keystrokes when the single key is pressed in command mode. In the following example,the @ key is mapped to replace the current word with "University of Washington". The <Control>v allows you to enter the <Esc> key into the command sequence.

From Command Mode

```
:map @ cwUniversity of Washington <Control>v <Esc> <Return>
```

Mapping can also be used to call commands external to vi, such as **sort** or **fmt**. In the following example, the @ sign is mapped to the **sort** command, so that the current paragraph (indicated by the }) will be sorted. The <Control>v allows you to enter the <Return> key into the command sequence. The second <Return> completes the map command.

From Command Mode

```
:map @ !}sort <Control>v <Return> <Return>
```

Note: You can also put abbreviation and mapping commands in your .exrc file.

D. TIPS AND TRICKS

1. Making vi an Editor in Pine

1. In your home directory, type: **vi .pinerc**
2. Find the line that reads

   ```
   editor=
   ```

3. Change it to read

   ```
   editor=vi
   ```

4. Write and quit the file. (**ZZ** or **:wq**)
5. Start Pine
6. In Pine in Compose mode, when you are ready to enter message text, you will see there is an option available called **Alt Edit**. (Alternate Editor). To evoke the Alternate Editor mode, press: <Cntrl><Shift>_

When finished editing in vi, exit vi and you will be returned to the compose screen.

---

2. vi-ing More Than One File

You can edit more than one file at a time with vi.

```
From The Unix Shell Prompt

vi file1 file2   vi two (or more) files at the same time

From Command Mode

:n               Move to file2 from file1

:rew             Rewind back to file1

:e!              Restore original file1 file2 (start all over)

ZZ               Save and quit file. (Must be done for each file.)
```

---

3. Moving the Left Margin

When you print a file you may want the left margin moved to the right. This leaves room for a three-hole punch.

```
From Command Mode

:1,$>               Move entire file 1 shift width (eight spaces)
                    to the right

:1,$<               Move entire file eight spaces to the left

:%s/^/        /g    Insert any number of spaces at the
                    beginning of each line in the entire file.
                    Simply press the space bar the
                    desired number of times.

:20>>                Moves next 20 lines over 1 shift width.
```

---

4. Issuing Shell Commands From vi

You can issue a single shell command while in the vi editor. For example, to list the files in your directory (ls), follow these steps:

```
From Command Mode

:w     Write changes to your file (just in case).

:!ls   List contents of your current directory on the screen.

Press <Return> to return to vi.
```

You can issue many shell commands by temporarily leaving the vi editor.

```
From Command Mode

:w     Write changes to your file.

:sh    Return to the shell to enter a number of commands
       without leaving vi.

Press <Control>d to return to vi editing.
```

---

5. Double Spacing a File

Occasionally, you may want a double spaced version of your file for editing or review.

```
In Command Mode

:w original.backup Save a backup copy of the original file

:%! sed G          Double space the entire file.

:1,5! sed G        Double space the lines from 1-5.
```

Class notes URL: *http://weber.u.washington.edu/~rells/R110/*
Last Modified: *February 24, 1997*