

Statistics 202A - vi Tutorial

Ryan Rosario

October 16, 2007

vi is by far my favorite editor. The material for this handout came from <http://www.eng.hawaii.edu/Tutor/vi.html> and credit is given to them. I transferred text from their tutorial into this handout in a condensed form but you may want to check out the site.

Throughout this tutorial $\sim X$ denotes a control character, that is, this means hold down the CONTROL key and hit X.

Entering and Exiting vi

Type vi followed by a filename (the filename is optional if creating a new file) at a command prompt to enter it.

To exit, type :q while in command mode. To get into command mode, hit ESC once.

To save any changes you have made to a file, enter command mode by pressing ESC. Then type :w followed by RETURN.

Saving and quitting can be performed in one stroke: :wq.

If you have unsaved changes that you wish to discard on exit, enter command mode and enter :q!. The ! overrides the save check.

If you want to save the file you were working as another filename called filename2, you would type:w filename2 and RETURN.

The Two Modes of vi

The first thing most users learn about the vi editor is that it has two modes: command and insert. The command mode allows the entry of commands to manipulate text. These commands are usually one or two characters long, and can be entered with few keystrokes. The insert mode puts anything typed on the keyboard into the current file.

vi starts out in command mode. There are several commands that put the vi editor into insert mode. The most commonly used commands to get into insert mode are a and i. These two commands are described below. Once you are in insert mode, you get out of it by hitting the escape key. If your terminal does not have an escape key, $\sim [$ should work (control-[]). You can hit escape two times in a row and VI would definitely be in command mode. Hitting escape while you are already in command mode doesn't take the editor out of command mode. It may beep to tell you that you are already in that mode.

How to Type Commands in Command Mode

The command mode commands are normally in this format: (Optional arguments are given in the brackets)

[count] command [where]

Most commands are one character long, including those which use control characters. The commands described in this section are those which are used most commonly the vi editor. The count is entered as a number beginning with any character from 1 to 9. For example, the x command deletes a character under the cursor. If you type 23x while in command mode, it will delete 23 characters.

Some commands use an optional where parameter, where you can specify how many lines or how much of the document the command affects, the where parameter can also be any command that moves the cursor.

Some Simple vi Commands

a	enter insert mode, the characters typed in will be inserted after the current cursor position. If you specify a count, all the text that had been inserted will be repeated that many times.
h	move the cursor to the left one character position.
i	enter insert mode, the characters typed in will be inserted before the current cursor position. If you specify a count, all the text that had been inserted will be repeated that many times.
j	move the cursor down one line.
k	move the cursor up one line.
l	move the cursor to the right one character position.
r	replace one character under the cursor. Specify count to replace a number of characters
u	undo the last change to the file. Typing u again will re-do the change.
x	delete character under the cursor. Count specifies how many characters to delete. The characters will be deleted after the cursor.

Cutting and Yanking

The command commonly used command for cutting is **d**. This command deletes text from the file. The command is preceded by an optional **count** and followed by a movement specification. If you double the command by typing **dd**, it deletes the current line. Here are some combinations of these:

d[^]	deletes from current cursor position to the beginning of the line.
d\$	deletes from current cursor position to the end of the line.
d^w	deletes from current cursor position to the end of the word.
3dd	deletes three lines from current cursor position downwards.

There is also the **y** command which operates similarly to the **d** command which take text from the file without deleting the text.

Pasting

The commands to paste are **p** and **P**. The only differ in the position relative to the cursor where they paste. **p** pastes the specified or general buffer after the cursor position, while **P** pastes the specified or general buffer before the cursor position. Specifying **count** before the paste command pastes text the specified number of times.

Indenting Your Code and Checking

The VI editor has features to help programmers format their code neatly. There is a variable that to set up the indentation for each level of nesting in code. In order to set it up, see the customization section of this tutorial. For example, the command to set the shift width to 4 characters is **:set sw=4**.

The following commands indent your lines or remove the indentation, and can be specified with **count**:

<<	Shifts the current line to the left by one shift width.
>>	Shifts the current line to the right by one shift width.

The VI editor also has a helpful feature which checks your source code for any hanging parentheses or braces. The **%** command will look for the left parenthesis or brace corresponding to a particular right parenthesis or brace and vice versa. Place the cursor onto a parenthesis or brace and type **%** to move the cursor to the corresponding parenthesis or brace. This is useful to check for unclosed parentheses or braces. If a parenthesis or brace exists without a matching parenthesis or brace, VI will beep at you to indicate that no matching symbol was found.

Word and Character Searching

The VI editor has two kinds of searches: string and character. For a string search, the / and ? commands are used. When you start these commands, the command just typed will be shown on the bottom line, where you type the particular string to look for. These two commands differ only in the direction where the search takes place. The / command searches forwards (downwards) in the file, while the ? command searches backwards (upwards) in the file. The n and N

Special characters:

^	Beginning of the line. (At the beginning of a search expression.)
.	Matches a single character.
*	Matches zero or more of the previous character.
\$	End of the line (At the end of the search expression.)
[Starts a set of matching, or non-matching expressions... For example: /f[iae]t matches either of these: fit fat fet In this form, it matches anything except these: /a[~ bcd] will not match any of these, but anything with an a and another letter: ab ac ad
]	Put in an expression escaped with the backslash to find the ending or beginning of a word. For example: /]the] should find only word the, but not words like these: there and other.
>	See the '<' character description above.

The character search searches within one line to find a character entered after the command. The f and F commands search for a character on the current line only. f searches forwards and F searches backwards and the cursor moves to the position of the found character.

The t and T commands search for a character on the current line only, but for t, the cursor moves to the position before the character, and T searches the line backwards to the position after the character.

These two sets of commands can be repeated using the ; or , command, where ; repeats the last character search command in the same direction, while , repeats the command in the reverse direction.

Summary of VI commands

This list is a summary of VI commands, categorized by function. There may be other commands available, so check the [on-line manual on VI](#). For easy reference, you can save this file as text and delete any commands you don't think you would use and print out the resulting shorter file.

Cutting and Pasting/Deleting text

Command	Description
-	Specify a buffer to be used any of the commands using buffers. Follow the " with a letter or a number, which corresponds to a buffer.
D	Delete to the end of the line from the current cursor position.
P	Paste the specified buffer before the current cursor position or line. If no buffer is specified (with the " command.) then 'P' uses the general buffer.
x	Delete the character before the cursor.
Y	Yank the current line into the specified buffer. If no buffer is specified, then the general buffer is used.
d	Delete until <i>where</i> . "dd" deletes the current line. A count deletes that many lines. Whatever is deleted is placed into the buffer specified with the " command. If no buffer is specified, then the general buffer is used.
p	Paste the specified buffer after the current cursor position or line. If no buffer is specified (with the " command.) then 'p' uses the general buffer.
x	Delete character under the cursor. A count tells how many characters to delete. The characters will be deleted after the cursor.
y	Yank until , putting the result into a buffer. "yy" yanks the current line. a count yanks that many lines. The buffer can be specified with the " command. If no buffer is specified, then the general buffer is used.

Inserting New Text

Command	Description
A	Append at the end of the current line.
I	Insert from the beginning of a line.
O	Enter insert mode in a new line above the current cursor position.
a	Enter insert mode, the characters typed in will be inserted after the current cursor position. A count inserts all the text that had been inserted that many times.
i	Enter insert mode, the characters typed in will be inserted before the current cursor position. A count inserts all the text that had been inserted that many times.
o	Enter insert mode in a new line below the current cursor position.

Moving the Cursor Within the File

Command	Description
^B	Scroll backwards one page. A count scrolls that many pages.
^D	Scroll forwards half a window. A count scrolls that many lines.
^F	Scroll forwards one page. A count scrolls that many pages.
^H	Move the cursor one space to the left. A count moves that many spaces.
^J	Move the cursor down one line in the same column. A count moves that many lines down.
^M	Move to the first character on the next line.
^N	Move the cursor down one line in the same column. A count moves that many lines down.
^P	Move the cursor up one line in the same column. A count moves that many lines up.
^U	Scroll backwards half a window. A count scrolls that many lines.
\$	Move the cursor to the end of the current line. A count moves to the end of the following lines.
%	Move the cursor to the matching parenthesis or brace.
^	Move the cursor to the first non-whitespace character.

Command	Description
(Move the cursor to the beginning of a sentence.
)	Move the cursor to the beginning of the next sentence.
{	Move the cursor to the preceding paragraph.
}	Move the cursor to the next paragraph.
	Move the cursor to the column specified by the count.
+	Move the cursor to the first non-whitespace character in the next line.
-	Move the cursor to the first non-whitespace character in the previous line.
_	Move the cursor to the first non-whitespace character in the current line.
0	(Zero) Move the cursor to the first column of the current line.
B	Move the cursor back one word, skipping over punctuation.
E	Move forward to the end of a word, skipping over punctuation.
G	Go to the line number specified as the count. If no count is given, then go to the end of the file.
H	Move the cursor to the first non-whitespace character on the top of the screen.
L	Move the cursor to the first non-whitespace character on the bottom of the screen.
M	Move the cursor to the first non-whitespace character on the middle of the screen.
W	Move forward to the beginning of a word, skipping over punctuation.
b	Move the cursor back one word. If the cursor is in the middle of a word, move the cursor to the first character of that word.
e	Move the cursor forward one word. If the cursor is in the middle of a word, move the cursor to the last character of that word.
h	Move the cursor to the left one character position.
j	Move the cursor down one line.
k	Move the cursor up one line.
l	Move the cursor to the right one character position.
w	Move the cursor forward one word. If the cursor is in the middle of a word, move the cursor to the first character of the next word.

Moving the Cursor Around the Screen

Command	Description
<code>^E</code>	Scroll forwards one line. A count scrolls that many lines.
<code>^Y</code>	Scroll backwards one line. A count scrolls that many lines.
<code>z</code>	Redraw the screen with the following options. "z<return>" puts the current line on the top of the screen; "z." puts the current line on the center of the screen; and "z-" puts the current line on the bottom of the screen. If you specify a count before the 'z' command, it changes the current line to the line specified. For example, "16z." puts line 16 on the center of the screen.

Replacing Text

Command	Description
<code>C</code>	Change to the end of the line from the current cursor position.
<code>R</code>	Replace characters on the screen with a set of characters entered, ending with the Escape key.
<code>S</code>	Change an entire line.
<code>c</code>	Change until . "cc" changes the current line. A count changes that many lines.
<code>r</code>	Replace one character under the cursor. Specify a count to replace a number of characters.
<code>s</code>	Substitute one character under the cursor, and go into insert mode. Specify a count to substitute a number of characters. A dollar sign (\$) will be put at the last character to be substituted.

Searching for Text or Characters

Command	Description
,	Repeat the last f, F, t or T command in the reverse direction.
/	Search the file downwards for the string specified after the /.
;	Repeat the last f, F, t or T command.
?	Search the file upwards for the string specified after the ?.
F	Search the current line backwards for the character specified after the 'F' command. If found, move the cursor to the position.
N	Repeat the last search given by '/' or '?', except in the reverse direction.
T	Search the current line backwards for the character specified after the 'T' command, and move to the column after the if it's found.
f	Search the current line for the character specified after the 'f' command. If found, move the cursor to the position.
n	Repeat last search given by '/' or '?'.
t	Search the current line for the character specified after the 't' command, and move to the column before the character if it's found.

Manipulating Character/Line Formatting

Command	Description
-	Switch the case of the character under the cursor.
<	Shift the lines up to <i>where</i> to the left by one shiftwidth. "<<" shifts the current line to the left, and can be specified with a count.
>	Shift the lines up to <i>where</i> to the right by one shiftwidth. ">>" shifts the current line to the right, and can be specified with a count.
J	Join the current line with the next one. A count joins that many lines.

Saving and Quitting

Command	Description
^\	Quit out of "VI" mode and go into "EX" mode. The EX editor is the line editor VI is build upon. The EX command to get back into VI is ":vi".
Q	Quit out of "VI" mode and go into "EX" mode. The ex editor is a line-by-line editor. The EX command to get back into VI is ":vi".
ZZ	Exit the editor, saving if any changes were made.

Miscellany

Command	Description
^G	Show the current filename and the status.
^L	Clear and redraw the screen.
^R	Redraw the screen removing false lines.
^[Escape key. Cancels partially formed command.
^^	Go back to the last file edited.
!	Execute a shell. If a is specified, the program which is executed using ! uses the specified line(s) as standard input, and will replace those lines with the standard output of the program executed. "!!" executes a program using the current line as input. For example, "!4jsort" will take five lines from the current cursor position and execute sort. After typing the command, there will be a single exclamation point where you can type the command in.
&	Repeat the previous ":s" command.
.	Repeat the last command that modified the file.
:	Begin typing an EX editor command. The command is executed once the user types return. (See section below.)
@	Type the command stored in the specified buffer.
U	Restore the current line to the state it was in before the cursor entered the line.
m	Mark the current position with the character specified after the 'm' command.
u	Undo the last change to the file. Typing 'u' again will re-do the change.

EX Commands

The VI editor is built upon another editor, called EX. The EX editor only edits by line. From the VI editor you use the `:` command to start entering an EX command. This list given here is not complete, but the commands given are the more commonly used. If more than one line is to be modified by certain commands (such as `:s` and `:w`) the range must be specified before the command. For example, to substitute lines 3 through 15, the command is `:3,15s/from/this/g`.

Command	Description
<code>:ab string strings</code>	Abbreviation. If a word is typed in VI corresponding to <code>string1</code> , the editor automatically inserts the corresponding words. For example, the abbreviation <code>:ab usa United States of America</code> would insert the words, "United States of America" whenever the word "usa" is typed in.
<code>:map keys new_seq</code>	Mapping. This lets you map a key or a sequence of keys to another key or a sequence of keys.
<code>:q</code>	Quit VI. If there have been changes made, the editor will issue a warning message.
<code>:q!</code>	Quit VI without saving changes.
<code>:s/pattern/to_pattern/options</code>	Substitute. This substitutes the specified pattern with the string in the <code>to_pattern</code> . Without options, it only substitutes the first occurrence of the pattern. If a 'g' is specified, then all occurrences are substituted. For example, the command <code>:1,\$s/Dwayne/Dwight/g</code> substitutes all occurrences of "Dwayne" to "Dwight".
<code>:set [all]</code>	Sets some customizing options to VI and EX. The <code>:set all</code> command gives all the possible options. (See the section on customizing VI for some options.)
<code>:una string</code>	Removes the abbreviation previously defined by <code>:ab</code> .
<code>:unm keys</code>	Removes the remove mapping defined by <code>:map</code> .
<code>:vi filename</code>	Starts editing a new file. If changes have not been saved, the editor will give you a warning.
<code>:w</code>	Write out the current file.
<code>:w filename</code>	Write the buffer to the filename specified.
<code>:w >> filename</code>	Append the contents of the buffer to the filename.
<code>:wq</code>	Write the buffer and quit.