How do you use the vi text editor?

1. What is the Visual Text Editor (vi)?

The Visual Test Editor, more commonly known as the vi text editor, is a standard "visual" (full screen) editor available for test editing in UNIX.

It is a combination of the capabilities of the UNIXed and ex text editors. It has specific modes for text insertion and deletion as well as command entering. Type view on the command line of the UNIX shell terminal to enter the read-only mode of vi. To switch between the line editor, ex, mode and the full screen wrapmargin vi mode:

- To enter the ex mode from the vi mode, type Q
- To enter the vi mode from the ex mode, type vi

When vi is entered on the command line, you enter vi's command (entering) mode. Positioning and editing commands can be entered to perform functions in this mode whilst advanced editing commands can be entered following a colon (:) that places you at the bottom of the file.

To enter text insertion mode, a vi text insertion command i, o, or a is typed so that text will no longer be interpreted as positioning or editing commands in this mode. Pressing escape will allow the user to return to the command mode.

To invoke the use of vi, use one of the following commands:

vi filename	Open or create a new file with the name filename
vi	Open a new file to be named later
vi –r filename	Recover crashed file
view filename	Open file read-only

2. Configuring your vi session

Use the line editor command :set during a vi editing session to configure various settings and functions of your current vi environment. To enable these

changes to be made automatically whenever the vi text editor is invoked; use the .exrc start-up file to configure those options.

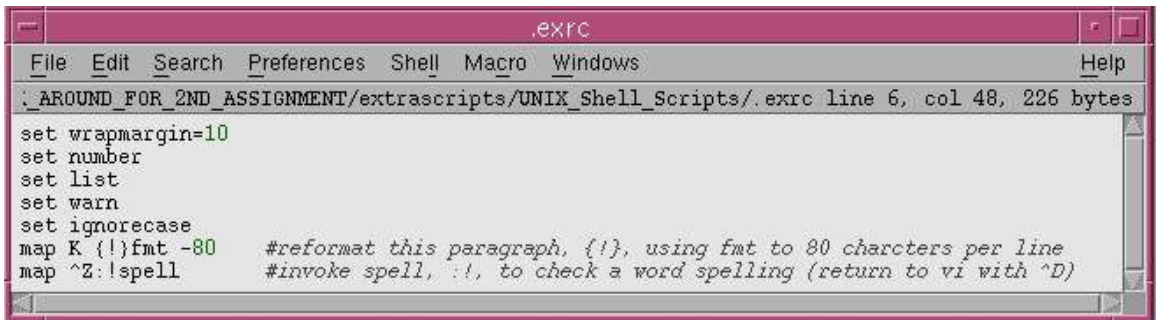
Macros that map keystrokes into functions using the :map command can be placed in the .exrc file. To insert control characters, type <control>-V (^V) followed by the desired control character.

2.1. Some of the options available in vi are:

Options	Descriptions
:set all	Display all options settings
:set ignorecase	Ignore the case of a character in a search
:set list	Display tabs and carriage returns
:set nolist	Turn off list options
:set number	Display line numbers
:set nonumber	Turn off line numbers
:set showmode	Display indication that insert mode is on.
:set noshowmode	Turn off showmode option
:set wrapmargin=n	Turn on word-wrap n spaces from the right margin
:set wrapmargin=0	Turn off wrapmargin option
:set warn	Display "No write since last change"
:set nowarn	Turn off "write" warning

2.2. Sample of the .exrc file

This is a sample of the .exrc file:



3. vi Quick Reference Guide

vi commands are case sensitive, unless otherwise indicated. The escape key needs to be pressed before entering a different command.

3.1. Cursor Movement Commands

- Moving the cursor by spaces

The basic movement commands to move up, down, left and right are:

(n)h or up the left arrow key	Move left by (n) spaces
(n)l or up the right arrow key	Move right by (n) spaces
(n)k or up the up arrow key	Move up by (n) spaces
(n)j or up the down arrow key	Move down by (n) spaces

Note that the (n) symbol indicates an optional number of spaces to be moved. For example, 8j or 8<down arrow key> will allow the user to move the cursor left by 8 characters. If no numbers are specified by the user, the cursor will be shifted by one character space as a default.

If the numlock key is on, arrow keys found on the number pad will yield a number if those arrow keys are pressed. To use those arrow keys, the numlock feature must be deactivated.

- Moving the cursor by screens

^F	Move the cursor forward one screen
^B	Move the cursor backward one screen
^D	Move the cursor down half screen
^U	Move the cursor up half screen

The symbol ^ is a control key; case does not matter for the subsequent letter

- Moving the cursor within a screen

H	Moves cursor to the beginning of the top line of the screen

M	Moves cursor to the beginning of the middle line of the screen
L	Moves cursor to the beginning of the last line of the screen
G	Moves cursor to the beginning of the last line of the file
(n)G	Moves cursor to the beginning of line (n)

- Moving the cursor within a line or amongst words

0	Move cursor to the beginning of the line
$	Move cursor to the end of the line
(n)w	Move cursor forward by (n) word(s)
(n)b	Move cursor back by (n) word(s)
E	Move cursor to the end of the current word

3.2. Inserting and Deleting Text

- Text insertion

i	Insert text before the cursor
I	Insert text at the beginning of the line
a	Append text after the cursor (does not overwrite other text)
A	Append text to the end of the line

- Text replacement

R	Overwrites characters until the end of the line or until escape is pressed to change command
r	Replace the character under the cursor with the next character typed

- Opening a new line of text

o	Adds a new line after the current line
O	Adds a new line before the current line

- Deletion of characters

(n)dw	Deletes (n) word(s)
D	Deletes from cursor to end of line
x	Deletes current character
(n)x	Deletes (n) character(s)
X	Deletes previous characters

- Deletion of lines

dd	Deletes current line
(n)dd	Deletes (n) lines

3.3. Change Commands

(n)cc	Cuts (n) line(s) and place the line(s) in the buffer
cw	Changes characters of word until end of the word (or until escape is pressed)
(n)cw	Changes characters of the next (n) words
c$	Changes text to the end of the line
C	Changes remaining text on the current line (until stopped by escape key)
~	Changes the case of the current character
J	Joins the current line and the next line
U	Undo the last command just done on this line

.	Repeats last change
s	Puts current character in the buffer so that it may be pasted somewhere else
S	Puts current line of text into the buffer so that it may be pasted somewhere else
:s	Substitute new word(s) for old :<line numbers affected>s/old/new/g For example, :1,$s/echo/yeah/g Or :4,19s/echo/yeah/g
&	Repeats last substitution (:s) command
(n)yy	Copies (n) lines to buffer (the lines are not deleted)
y(n)w	Copies (n) words to buffer (the words are not deleted)
p	Puts copied or deleted text after cursor
P	Puts copied or deleted text before cursor

3.4. File Manipulation

:w (filename)	Write changes to file with the name filename (default is current file) If filename does not exist in the designated working directory, the file named filename is created. For example, :w ../Hello.txt saves the text in this vi session as a Hello.txt file in the parent directory
:wq	Write changes to the current file and quits edit session
:w! (filename)	Overwrites file (default is current file)
:q	Quits edit session with no changes made
:q!	Quits edit session and discards changes

:n [filename]	Edits next file in argument list :n Hello.txt opens the file Hello.txt in vi Having typed vi file1 file2 at the command line, whilst viewing file1, entering :n opens file2 in vi
:f (filename)	Changes name of current file to filename
:r (filename)	Inserts contents of filename into current edited file at the current position
:!(UNIX command)	Escapes to the shell and allows users to enter and execute a UNIX command that does not affect the contents of the directly. For example, :!pwd shows the current working directory at the current line until enter is pressed.
:r!(UNIX command)	Escapes to the shell and allows users to enter and execute a UNIX command. The result of the UNIX shell command is inserted at the current position. For example, :r!ls inserts the contents of the current working directory at the current line
ZZ	Write changes to the current file and exit