**Vi** (Visual Text Editor) and **Vim** (Visual Improved Text Editor)

**To execute the editor from the command line:** vi *filename*

(If you execute vi, and VIM shows up in the terminal header, you are actually running vim.)

The vi editor has four basic modes of operation:

* *input* mode Text typed on the keyboard is entered into the document.
* *command* mode Keys pressed on the keyboard are interpreted as commands to the editor.
* *command-line* mode Commands are entered at the bottom of the window.
* *visual* mode Selecting text for an operation, such as saving to a file, or copy/paste.

The current mode determines what keyboard input does. The ESC key always returns you to *command mode*.

**Cursor navigation** (because all commands are executed at the current location of the cursor)

|  |  |
| --- | --- |
| ↑ ↓ → ← (arrow keys) | move one character or one line. |
| CTRL-↑ ↓ → ← (arrow keys) | move one page, or one word. |
| Home, End keys | move to the start or end of the current line. |
| page-up, page-down keys | move one screen at a time. |
| e (end) | move to the end of the current word |
| b (beginning) | move to the beginning of the current word |
| :LineNumber | move to the desired line (e.g., :23) |
| :$ | move to the last line in the file |
| *(using mouse)* | scroll using the wheel, click to place cursor |

**Inserting text** (which puts you into INSERT *input mode* until you hit ESC)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *insert* | i | at the cursor | I | at start of current line |
| *append* | a | after the cursor | A | after end of line |
| *open a new line* | o | after the current line | O | before current line |

**Replacing text** (which puts you into REPLACE *input mode* until you hit ESC)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *replace* | [n] r | one or more characters | R | enters REPLACE *input mode* |

**Deleting text** (and leaves you in *command mode*)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *delete* character | [n] x | at cursor | [n] X | before cursor |
| *delete* line | [n] dd | entire current line | [n] D | rest of line |
| *delete* | [n] de | to end of current word | [n] d$ | rest of line |
|  | [n] dw | to start of next word | [n] db | to beginning of current word |
| *join lines* | [n] J | deletes '\n' characters | | |

**Selecting text** (for operating on a block of text)

v starts visual mode; then move cursor to the end of the desired block.

(*using mouse*) click and drag to select a block of text.

**Copy/paste** (and leaves you in *command mode*)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *yank* (visual mode) | y | current selection |  |  |
| *yank* line(s) | [n] yy | entire current line | [n] Y | entire current line |
| *yank* characters | [n] ye | to end of current word | [n] y$ | rest of line |
|  | [n] yw | to start of next word | [n] yb | to beginning of current word |
| *put* | [n] p | after/below cursor | [n] P | before/above cursor |

Commands marked with a [n] can be preceded by an integer value that specifies the number of times to apply the command. For example:

|  |  |
| --- | --- |
| 3dd | deletes 3 lines of text (the current line and the two lines below it) |
| 5x | deletes 5 characters (the current character and the 4 characters after it) |
| 3de | deletes the rest of the current word and the next 2 words |

**Searching for text** (puts you in command-line mode)

|  |  |
| --- | --- |
| /pattern | find the pattern string in the text (uses *regular expression* syntax) |
| n | go to the next occurrence of the pattern (searches down) |
| N | go to the previous occurrence of the pattern (searches up) |
| /\<pattern\> | find the **exact** pattern in the text |

**Quitting, Saving Text, and Reading text** (colon, :, puts you in *command-line mode*)

|  |  |
| --- | --- |
| :q | quit vi assuming NO changes have been made to the text |
| :q! | quit vi and discard any changes made since the text was last saved |
| :w | write the current text to the current file |
| :wq | write the text to the file and quit the editor |
| :w *FileName* | write the current text to the specified file (the file name must not previously exist) |
| :w! *FileName* | write the current text to the specified file (it overwrites an existing file if it exists) |
| :r *FileName* | Read a file and place its contents after the cursor |
| :! cmd | Execute any UNIX command while still in vi |
| :r !cmd | Execute any UNIX command and put the output after the cursor |

**Undo and Redo**

|  |  |
| --- | --- |
| [n] u | Undo the previous command. |
| [n] CRTL-R | Redo an undo |

**Some useful *command-line* mode ex commands** (which work on one or more lines at a time)

:[*range*]cmd[*options*]

|  |  |
| --- | --- |
| range | n,n where n is a line number. (. is the current line; $ is last line in the file) If no range is specified, the current line is the "range". |
| m *TargetLine* | **moves** the specified group of lines below the specified "target line" |
| co *TargetLine* | **copies** the specified group of lines below the specified "target line" |
| d | **deletes** the specified group of lines |
| s/*old*/*new*/ | **substitutes** the first occurrence of "old" on each line with the "new" string |
| s/*old*/*new*/g | **substitutes** all occurrences of "old" on each line with the "new" string |
| w *FileName* | **write** the specified group of lines to the specified file |

examples:

:10,20 m 50 moves 11 lines (lines 10 thru 20) and places them after line 50

:10,20 d deletes 11 lines (lines 10 thru 20)

:100,$ d deletes all of the lines starting at line 100 to the end of the file

:1,$ s/wayne/fred/g changes every occurrence of "wayne" to "fred" in the entire file

:.,$ w temp writes the lines from the cursor to the end of the file to a file called "temp"

**Help** (accesses the manual pages that are displayed on half the editor screen; to exit help use :q)

|  |  |
| --- | --- |
| :help | Starts the help screens |
| :help cmd | Gets information about a specific command. |

**Configuring Vi** (Visual Text Editor) and **Vim** (Visual Improved Text Editor)

vim is highly configurable.

|  |  |
| --- | --- |
| :set | displays the current options that are set |
| :set all | displays all of the possible options that can be set options are either off (start with a "no")or on, (e.g. noshowmode or showmode) |
| :set *option* | sets the specified option; pretend "no" to disable an option |

Put common settings in the file "~/.vimrc". (If using vi, not vim, use "~/.exrc").

Here is a sample configuration file with Dr. Brown's desired settings:

" Display useful information while editing

set showmode " always display the current mode in the lower-left corner

set showcmd " display the commands as they are typed on the bottom line

set number " display line numbers

set ruler " always display the cursor's line number and column number

set hlsearch " highlight words that are searched for

set mouse=a " enable the use of the mouse in all modes

set ignorecase " will ignore case when performing searches

" Set the tab stops for editing code, always substitute spaces for tabs

set expandtab " insert spaces when the tab key is used

set shiftwidth=2 " indent 2 spaces

set softtabstop=2 " indent 2 spaces

set cindent " perform automatic indention for c code.

Notes:

* Color syntax highlighting is typically automatically enabled according to file type. File types are recognized by file name extensions.
* You can turn syntax highlighting on for any language, regardless of the file's extension  
  set syntax=c  
  set syntax=php  
  etc