Vim Basics

1. Modes

   Vim, by default, runs in a mode which does not let you begin typing (unlike notepad, Word, etc). Instead, vim begins in *command* mode. Each mode is accessed by pressing a key, and *escape* returns to command mode. The various modes are as follows.

   (a) Insert mode: The mode that lets you type normally. When in command mode, vim allows you to start typing before the cursor (i/I), after the cursor (a/A), or in adjacent lines (o/O).

   (b) Visual mode: v allows you to highlight text, as you would by using a mouse in a graphical editor.

      i. Block: *ctrl+V* Visual Block selects characters in a rectangular shape, controlled by the movement (*h, j, k, l*) keys.

      ii. Line: *shift+V* Selects entire lines at a time.

   (c) Movement: *h, j, k, l* These four keys are located adjacent to each other on the keyboard, where the right hand sits. This is intentional; they seem to be a strange collection, but the arrangement allows easy movement of the cursor without having to move your hand to a mouse, or the arrow keys.

   Vim deals with text by the character, word, line, or paragraph. With many commands, you can combine a command with one of the letters w, W, (for 'words'), b, B, (for words, moving backwards from the cursor), or l (for lines). Simply pressing w, W, b, or B moves forward or backward by that amount while in command mode. The brackets, { and } move forward and backward one paragraph at a time.

2. Deleting text: d and x delete text. x removes one character at a time; X deletes backwards. dw and dW delete words, whereas db and dB delete words backwards. dd deletes an entire line, and D deletes everything from the cursor to the end of the line.

3. Copy/Pasting text

   (a) "yanking" is the equivalent to copying, so named because you use the y key to copy selected text. Anything highlighted (in Visual mode) will be placed into a buffer. There are other buffers available for your use, named a - z. To use the registers, prefix the commands with a quotation mark, *"*, followed by the letter of the buffer. For example, to yank into register q, the command is *"qy*. Pressing *yy* grabs an entire line.

   (b) "pasting" puts the contents of a buffer back into the file being edited. To paste, p pastes after the cursor, P before. Similar to yanking, prefixing the command with *"* and a letter will paste from the specified buffer.

4. Files

   (a) Opening: The most basic way to open a file is from the terminal, when launching vim. Once you are using vim, the *:e* command, followed by a filename, opens a file for reading.

   (b) Saving: The *:w* command writes your document to disk. Adding a name, i.e. *:w main.cpp* will save to that filename.

   (c) Quitting: *:q* quits vim. This can be used in conjunction with *:w*, to form the command *:wq*, which writes your file, then quits.

   If you are editing a file and have not saved it when trying to quit or open a new file, you will get a message saying *No write since last change (use ! to override).* By appending an *!* to the *:e* or *:q* commands, you can discard any changes since your last save.

Advanced Vim Features

1. The tilde (˜) key will change the capitalization of the current selection (i.e., 'vim' becomes 'VIM')
2. Moving to the beginning or end of a line can be accomplished with *0* and *$*, respectively.
3. By pressing *m*, followed by any letter a-z, you can save ("mark") a location in your document, and jump to it using the apostrophe ´ key, followed by the same letter. This lets you jump to important points of a file quickly.
4. Syntax highlighting can be turned on using the *:syntax on* command. It can be turned off with *:syntax off*.
5. Vim allows you to run terminal commands without having to quit vim. Pressing *:!*, followed by a terminal command, vim will execute the command, then return you to the vim window. For example, *:!ls* will print out the contents of the current directory, prompt you to hit Enter, then return to vim.
6. The *.vimrc* file resides in your home directory, and contains a list of preferences. In this file, you can set your favorite options (syntax highlighting, etc) to be set every time vim is launched.
7. *u* undoes the last action. *ctrl+r* redoes it.
8. The / command allows you to search a file for a string. */linux* will search the document for the string 'linux'. Pressing *n* jumps to the next occurrence of the string, and *N* moves backwards.
9. Folding

   In a class such as cs225, Folding is one of the most useful features of vim. Folding lets you 'hide' areas of your document, which you can collapse and expand at will. Vim creates folds based on syntax, indentation, or you can create them manually. When dealing with a language such as C++, vim's syntax detection does an excellent job of creating folds.

   (a) Enabling folds: Folds are turned on and off with the *:set foldenable* and *:set nofoldenable* options.

   (b) Fold method: There are three primary methods of doing this; *syntax*, *indent*, *manual*. Toggle between them with *:set foldmethod=OPTION*, i.e. *:set foldmethod=syntax*.

   (c) Folding Commands: There are several ways to expand and collapse folds.
   za toggles a fold.
   zc and *zC* close a fold, and recursively close all folds that contain the area under the cursor.
   zo and *zO* opens a fold, and recursively opens all folds under the cursor.
   zR and *zM* open all folds in a document, and close all folds in a document.
   In manual folding mode, zfNj creates a fold from the cursor down N lines. *zd* deletes the fold at the cursor, and *zE* deletes all folds.
10. Tabs

    New versions of Vim allow you to open up separate tabs with different documents in each. To create a tab, use the command *:tabnew*. The commands *:tabnext* and *:tabprev* move to the next and previous open tabs.
11. Split Windows

    Vim lets you divide your window into multiple areas to edit different regions of the same file, or different files concurrently.

    :split divides the window in half horizontally. Prefixing it with a number, such as *:20 split*, creates a split of 20 characters.

    :vsplit divides the window vertically.

    Pressing *ctrl+w*, followed by one of the direction (*j, k, l,*) keys, will move focus from one area to another.

    The quit (*:q*) command will close a split area.