Wonders of Vim

This document describes some cool features of vim that help me during a development cycle.

Being a programmer, one requires to modify/re-structure the code under development quite a few times. One may also have to read/write/modify (no pun intended) a huge code. Vim provides many cool features to relieve a developer from the pains that may arise during that proccess. Let's look at some of those features.

Note: (The double quotes ("…") are added to demarcate the commands/key bindings separately, one should not type those quotes while trying hands-on.

1. **Find:** Often times we need to find a particular phrase or word (string/sub-string) in a source file. Vim offers a quick way to find/search. Just open the source file and type in "/ " in the Normal Mode. You may notice that the line & column vanish from the bottom right corner once you do this. After typing " / " just type the string you wish to search for, e.g. " /int main(void) " and hit the enter key. For moving onto the next found instance of the string just hit the "n" key int the Normal Mode.

2. **Replace:** Vim provides the substitute command to searches for a text pattern, and replaces it. In order to enter a command in vim one needs to type a colon (:) followed by the command (again in the Normal Mode). For example to enter the substitute command one should type- :substitute. Vim also provides a short-hand representation of these commands in order to save time, so one can use the substitute command by just typing- :s. You may learn it as s=substitute. One may also use regex to find/replace with vim.

   The substitute command gives you many options to replace, but these are some good ones that you'd want for sure:

   a. **:s/hello/word/g -** Find each occurrence of 'hello' (in the current line only), and replace it with 'world'.

   b. **:%s/hello/world/g -** Find each occurrence of 'hello' (in all lines), and replace it with 'world'.

      The above two commands will directly replace all occurrences of hello with world, i.e. without asking a confirmation for each replacement. But in some situations, one may not want to replace all occurrences of hello. In order to command vim to ask for confirmation before every replacement, one may use the following.

   c. **:%s/hello/world/gc -** Change each 'hello' to 'world', but ask for confirmation first. (c flag tells that).

      One may also want to replace some words that are exactly the same and not a sub-string on any string, For Example: If there's a comment that says /* Payhellomoney */ the above commands will modify it to /* Payworldmoney */. In order to avoid that vim allows the user to tell if they want vim to look for the exact pattern.

   d. **:%s/\<hello\>/world/gc -** Change only whole words exactly matching 'hello' to 'world', but ask for confirmation.

Do you require a case insensitive replacement? This is one is for you!

e. **:%s/hello/world/gci -** Change each 'hello' (case insensitive due to the i flag) to 'world', again ask for confirmation.

f. **:%s/hello\c/world/gc** is the same because \c makes the search case insensitive.

3. **Buffered Copy-Paste:** The last document taught you about copy paste in vim. Selection maybe done by pressing v and entering the visual mode, and to copy the selected text one may press y (yank). And to paste one may press p. But sometimes, we developers need to copy 2 or 3 or even more blocks of text at a time. For Example, if two functions have the same arguments and return types, but the names are different, if we could simultaneously copy only the arguments and the return type (without the function name) and paste them separately. This would speed-up the typing process. Vim provides a really cool feature that makes this happen.

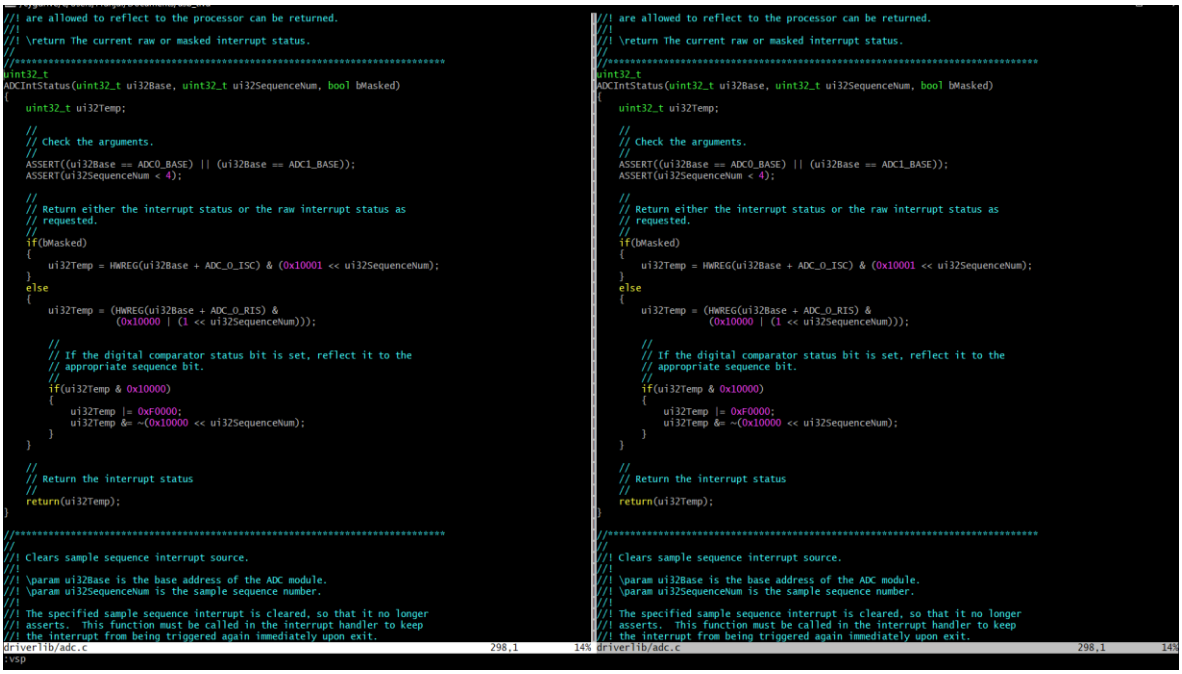
   Just select some text to copy as described above (by pressing v). But this time, don't simply press y to copy. Before pressing y, type "a and then y. Basically, type in **"ay**. Doing this, copies your data into the 'a' buffer. Now go ahead and select a different text and this time type **"by**. This copies the other text into the 'b' buffer. Now go to an empty space and type **"ap**. You'll see the text that was copied first (former) was pasted. Now type **"bp** and you'll see the text that was copied later was pasted. Vim provides you the flexibility to choose many copy-paste buffers from a-z.

4. **Undo & Redo:** Everyone is a human being and everyone may hit the mistake mountain at times (unless you are using an AI). An easy correction to your mistakes can be undone by pressing **u** in the vim editor. Also, it is human to figure out that some mistake that was just undone was a brilliancy, want that mistake/brilliancy back? Just press **CTRL + r** and vim will bring it back for you.

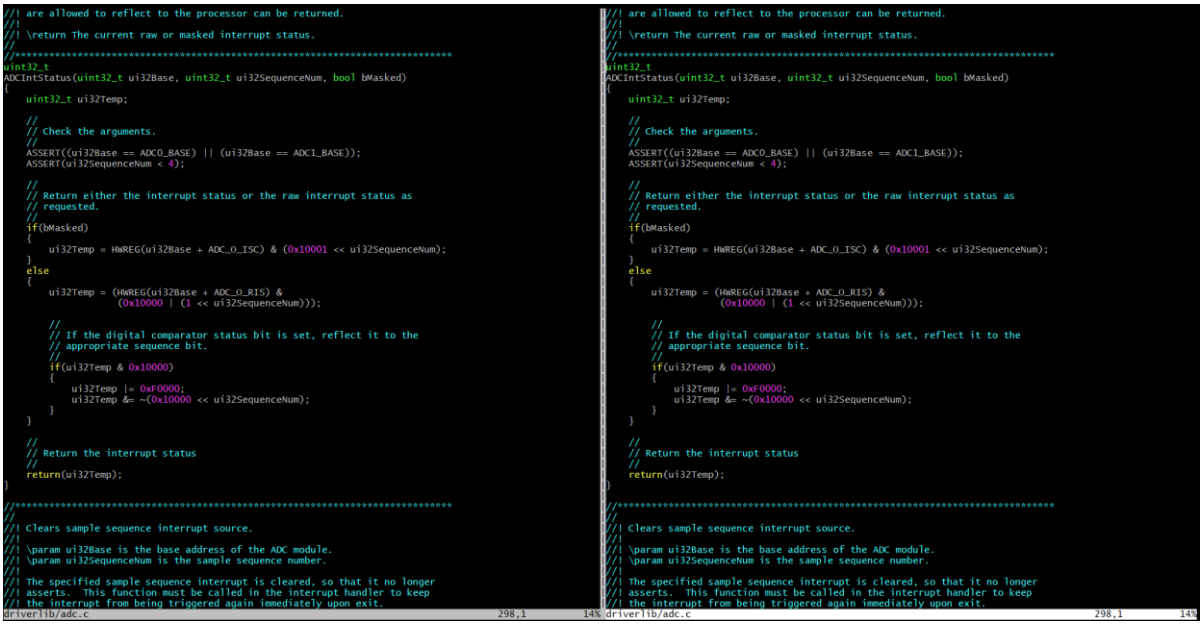
5. **Split-Screens:** Want to look at more files at a time? Vim's got your back! Vim has a split command that enables you to look at more than one files in one vim session. Let's see how can we work with this feature.

   a. **Splitting the screen**: In order to split the screen, just type in the command split in normal mode, i.e. type **:split .** But the same file opens in the split. Wanna open a new one? Just enter **:split <file's path>.** For example- **:split main.c.** Viola! You just split the screen and opened a different file. The short-hand for this command is **:sp <path> .**

b. **Vertical Split:** So, you didn't find the horizontal split comfortable? No problem, you can split your screen vertically too! Just type **:vsplit <path>.** This command too has a short-hand representation, **:vsp <path>.** E.g. **:vsp main.c**



c. **Moving between splits:** The screens are now split and it's natural want to edit the file opened in other split. How to get to that file? Just press **CTRL + w + w.** That'll take you to the other split. Go ahead and edit your file as soon as you see the required split it highlighted.



(Notice the right file is highlighted instead of the left one as in the image before)

d. **Resizing the splits:** Wanna change the size of a particular split? Just go to that split, and type **CTRL + W** followed by **Shift + >** to increase the size. To decrease the size type in: **CTRL + W** followed by **Shift + < .**