1 Introduction

This note outlines a course on editing and browsing code effectively. Following are the objectives of this course

1. Start proper way of typing if not already doing.
2. Get expertise in any one editor. Note: this course is based on vim, but if you know emacs editor very well, then contact your mentor to customize this course for you.
3. Get awareness of coding style
4. Get expertise in any one tool used for browsing code, any one tool used for correcting indentation and any one tool for comparing and merging code.
5. Develop the habit of writing code only in the correct style even for test code.

2 Resources

1. We will use vimtutor utility for learning vim editor
2. Following url can be referred for an introduction to coding style

   https://www.kernel.org/doc/html/v4.10/process/coding-style.html
3. Following quick tutorial can be used to get familiar with cscope code browsing utility with vim

   http://cscope.sourceforge.net/cscope_vim_tutorial.html
4. Following url can be referred for installing and using gnu indent utility

   https://www.gnu.org/software/indent/
5. Following url can be used to get meld compare and merge utility

   http://meldmerge.org/

3 Course outline

1. If you already don't know how to type properly ("touch typing"), you shall start now. You don't have to type without looking at the keyboard, but you shall use the correct finger for each key. This is not much difficult. You do some half day practice with the 'asdf' line. Then understand the proper finger for each key (except the function keys and special keys) and then just force yourself to only this way from now onwards. Within a week or so you shall be at speed. For all exercises in this course and after this course you shall be typing the proper way only.
2. Use vimtutor to study basic and advanced editing features of vim.
3. Study #2 mentioned in Resources above properly. Understand each rule and rationale behind the same.
4. Install cscope and study its use using #3 mentioned in Resources above. Get some code base and browse through it using cscope features in vim

5. Install gnu indent and study how to write rules file. Use #4 mentioned in Resources above
6. Install and learn how to use meld. Use #5 mentioned in Resource above

4 Exercises

4.1 Vim exercises

1. Similar to .bashrc file for bash, you can have ~/.vimrc in your home directory to add your preferences when vim starts. Normally there is a vimrc template available in /etc path. Find this one and copy it as .vimrc into your home directory. Add the following preferences to this file
   a) set tab stop to 4
   b) set shift width to 4
   c) enable expanding tabs as spaces
   d) enable auto indentation
2. Create a C file with some 4 dummy functions. Keep some nested if loops inside the function. For variables and arguments use multi-letter names. Save this file as a.c
3. Exercise following movements using most appropriate movement commands.
   a) Go to end of file, then go to start of file, go to starting brace of the first function, then go to the end brace, go back to the starting brace, go to the start of the line of function prototype, go to starting parenthesis, go to first argument, go to last letter of first argument
4. put markers at start of each of the functions. Move from one marker to another.
5. With a.c open in vim, use horizontal split to open a new file named b.c. Now we need to copy first and third functions to b.c. Copy each block to separate records. Go to b.c and paste both records one after other. Save and close b.c.
6. use shift left command to make all lines of function 1 and 2 start in the first column. Then use the ‘=’ command to correct the indentation
7. Do the following search / replace operations
   a) search for the function prototype line using regular expression syntax
   b) search and replace all ‘int’ with ‘unsigned int’ only in lines between 6 and 20
8. In vim we can record a sequence of commands and then replay that. Use this feature to do the following. Go to each function and define a new variable. Replay this three times so that the variable is added in all four functions
9. open b.c using vertical split. Copy functions 2 and 4 to b.c. save b.c. close both a.c and b.c.
10. Use vimdiff to find the difference between a.c and b.c. Make b.c same as a.c using minimum typing.
11. Find how to do the following

a) show line numbers

b) show a vertical line at 80 column, so that we know when we cross 80 column limit.

c) Execute a shell command and come back

d) Symbol to refer to the file being edited in the command line

e) Compile the current file from vim command line itself.

f) Enable spell check

4.2 Gnu indent exercise

1. Create or get a gnu indent rule file that conforms to the Linux kernel coding style. Make the following changes

   a) indentation is 4 columns

   b) tabs shall be replaced with spaces

2. In a.c created in the vim exercise, open it in vim and corrupt the indentation by shifting lines left and right randomly. Run gnu-indent and fix the indentation.