Lab 2: Source Code Editing
CMSC 340
Due: Mon. Feb. 18 by 11:59:59pm

Both developers and system administrators spend a LOT of time editing text files. Those files might be the source code for a project or configuration files for a server. They might also be documentation or a message to users. A typical workflow for a sysadmin involves opening lots of files over and over, performing a short simple task, and then closing the editor to work from the command line. Those short little tasks add up to a lot of time, so it's important to learn to be efficient in the editor.

Most graphical editors take a long time (relatively speaking) to load and display an editable document. It is also difficult to use a graphical text editor remotely over the network (for example, by SSH). That means that most power users prefer simple terminal editors. The two most popular editors are emacs and vim.

In this lab, you are going to practice some of the more advanced features of vim.

Step 1. Setting Up

Make a folder named Lab2. In that folder, type "git init" to create an initial git repository. Then download the file "Lab2.txt" from the course web site and add it to your project (don't forget to both "git add" and "git commit").

Step 2. Task One: A script

Your first task for this lab is simple to describe, but tricky to carry out. Create a file named "script.vim" that contains a single function named "InsertText". When run, the InsertText function should insert the words "Longwood University" in front of every instance of the words "Computer Science" in the Lab2.txt document. To do this, you should use:

1. A while loop.

2. The "search" function.

3. The "normal!" keyword, which executes its arguments as if the user had typed them while in normal mode.

4. The "execute" keyword, which can be used with "normal" to include control characters and spaces.

5. The "cursor" function, which can position the cursor on a line, giv en the line number.

BE SURE TO CITE ANY SOURCES (especially online source) you use and DO NOT COPY LARGE BLOCKS OF CODE (more than two or three lines) FROM ANYONE.

Step 3. Task Two: Mapping

Map the function you created in Step 2 to the key <F2> so that **when pressed in normal mode** the <F2> key triggers the function.

Step 4: Task Three: Code Indenting

Now download Lab2.cpp from the course web site. Add it to your project. This is some code I wrote a long time ago that is part of a plugin that hooks my calendaring software up to the submit web site. I've deliberately obfuscated the code by munging all the text into a single line. Your job is to re-indent the code so that it follows proper C++ style. For the purposes of this lab, that simply means that every statement should be on its own line and every block should be indented one “vim shiftwidth” more than scope in which it is nested. Try to use as few vim keystrokes as possible. Use “git” to track your changes as you go.

Some things that might be useful to remember:

Every C++ statement in this file ends with either a semicolon or a curly brace.

Not every block is contained within curly braces (for example, an if-statement body which consists of a single line).

You can record a macro in vim by typing “q” followed by a letter, performing a sequence of commands, and then typing “q” again. You can playback a macro by typing @ and then the letter for that macro. You can play it back 100 times by typing “100@” and then the letter for the macro.

Submitting

Make a tarball of your code by typing:

```
cd ..
tar czvpf Lab2.tar.gz Lab2
```

and then log into http://marmorstein.org/~robert/submit/ to turn upload it to my server.