CS-382 Programming Tools

Assignment #1 Using the vi Editor

OBJECTIVES

- Entering and exiting the editor
- Entering and exiting command and insert mode
- Inserting and deleting text
- Moving the cursor
- Refreshing the screen

Introduction The vi editor is an editor that is available with all standard Unix and Linux systems. It is a screen-oriented editor that is different from most other editors you might have used, which means that it is not very "user friendly". You must practice and use vi frequently to become proficient and to find vi useful.

There are two major modes in vi, "command mode" and "text entry mode". The editor starts in command mode. You must issue a text entry command to get into text entry mode. Therefore, you must learn some of the fundamental commands such as "**i**"nsert, "**x**"delete, and "**dd**"delete described below. However, first you have to learn to start vi.

Starting the vi Editor To edit a file that will have the name "hello.cpp", you issue the command *vi hello.cpp*. Since hello.cpp does not exist, you will get a blank screen except for a ~ in the first column of each line. The ~ is a symbol for an empty line (which is different from a blank line). The cursor will be in the upper left hand corner of the screen.

Entering insert mode To start typing text into the file, type **i** for insert. Note that this "i" command is not displayed on the screen. That is, there is no change on the screen. In other words, there is no way to look at the screen and tell if you are in insert or command mode (not very friendly, is it). Start inserting by typing the first line of the "hello.cpp" program displayed below. If the characters you type are displayed, then you are in insert mode. Also, note there is no "wrap-around" in this editor; you must press <return> at the end of each line. Practice by typing in the HELLO program.

*** Note that you MUST use the double quote character to begin and end a string value.

```
// your name
// hello.cpp
#include <iostream>
using namespace std;
```

```
int main()
{
cout << "Hello Everyone!" << endl;
return 0;
} // end-main
```

Exiting insert mode Now exit insert mode by pressing the **<Esc>** key. Again, you will notice there is no change in the screen. However, what you type next will be interpreted as commands and not added to the file (since you are now in command mode). If you are in command mode and you press <Esc>, terminal will beep at you (do not worry, this does not mean you made a mistake). This is the only way to determine which mode you are in. Press <Esc> several times. If it beeps, you are in command mode; if it does not beep you were in input mode, but now are in command mode.

Moving the cursor The arrow keys on the keyboard can be used to move the cursor to specific positions in the text. However, you must be in command mode to do this! Get into command mode now (press Esc), and try moving the cursor around. It will only move over text and not past the end of the line. On a PC you may need to make sure the "Num Lock" is off. The cursor keys do not move the cursor in insert mode, so don't use them in insert mode! For an experiment, move the cursor to the end of the “cout” line, and get into insert mode (type i). Then press the right-arrow key. What happened? This is not what you wanted, so let's learn how to delete characters.

Deleting Characters To delete a character get into command mode, and then place the cursor over (on top of) the character to be deleted. Now press **x** and the character under the cursor will disappear and no "x" is displayed). Try x-deleting some of the characters that you already have typed. Note that you may not use "backspace" in command mode, only in insert mode.

Deleting Lines To delete an entire line, get into command mode. Then position the cursor anywhere on the line you want to delete, and type **dd**. The line will be replaced with the @ symbol indicating an empty line. Try this by deleting one or more lines that you have typed in.

Refreshing the screen Sometimes (frequently) the text on the screen will be messy (from inserts and deletes, primarily). The best way to clean up these errors and have the screen accurately reflect the file's contents is to refresh the screen. You do this by pressing **<Ctrl-L>** (the "control" key and the "L" key simultaneously). Press Ctrl-L now, and note what happens on the screen.

Exiting the editor To save changes and exit the editor type **ZZ** (Shift-Z Shift-Z). Notice this command is capitalized. No backup file is kept, so if you do not like these new changes, tough.

To exit the editor but NOT save the changes you made, get to command mode and then type **:q!<Return>** (that is, type colon, "q", exclamation point, and then RETURN). You might use this if you have really messed up the file in vi. This will keep the file as it was before you started editing it.

Laboratory Exercise

Exercise1. Edit the file hello.cpp, and fix it so that it is a correct C++ program and execute it.

Exercise 2: Start vi in preparation for some tutorial exercises; throughout this document, the exercises will use the file, sample. Type:
vi sample RETURN
The window clears and displays the contents of the file, sample. Since it is a new file, it does not contain any text. vi uses the tilde (~) character to indicate lines on the screen beyond the end of the file.

Exercise 4: To enter input mode, press i
Note that in the bottom right-hand corner, vi indicates that you are in input mode. Then enter the following text, and remember to press RETURN at the end of each line.

We scrambled to strike camp. Water crashed down upon us, far too slow
in foot and hands to do it. "Oh, no," I said.
Our heavens darkened grimly. Thunder echoed overhead and shook the
clouds, even the ground rumbled as each clap loudly exploded.
Glancing back, I saw an ocean rising behind us.
It just wouldn't stop raining.
Our weathered tent was a poor shelter tonight. As a lightning bolt
flashes over the hills, we made out a small cave on the mountainside.
A safe haven, thought I.
Our heavens fell down, but just up ahead lay safety.

After entering this text, press ESC to return to command mode.

Exercise 5: Try using the cursor movement keys to place the cursor on the word "hands." Move the cursor onto the "s." Press **x** to delete it. Now let's delete the word "loudly." Move to the beginning of the word. Although you could press **x** seven times to delete the word and its trailing space, it is quicker to delete it by typing **dw**. Finally, remove the line "It just wouldn't stop raining." Move the cursor anywhere on that line, and type **dd**. The line vanishes.

Exercise 6: Move to the line "first line" and delete it by typing:
dd
Bring back the line by pressing:
u
which undoes the last text change. Press **u** a few more times, and watch what happens.

Exercise 8: Move to the blank line just after "last line." Press:
i
and insert the following text:
I saw everything spinning wildly. RETURN
Press ESC to return to command mode. Now move to the line "earlier last line" and press:

J to join it with the line you just typed.

Exercise 9: In command mode, type:
:w RETURN
to enter the "write" command. After a moment, vi gives you a report at the bottom of its window:
"sample" 13 lines, 539 characters
Don't worry if the line and character numbers differ a bit from yours. Most importantly, you just saved your work. Now you are ready to quit vi. From command mode, type:
:q
Again the colon signals an ex command at the bottom of the vi display. Press RETURN to finish entering the command and to exit vi.

Exercise 10: To display the line numbers, enter the following command:
:set number RETURN
This command will immediately display the line numbers in the left margin of your vi window.

Exercise 11: From command mode, try jumping to line 1, the end of the document, and line 6 by entering:
1G
G
6G
After typing each of these "go to" commands, notice that the cursor jumps to the desired line. You can type a multiple-digit line number, of course. Next, turn off the line numbers by entering:
:set nonumber RETURN
Finally, press CTRL-G and the modeline explain what the outputs are?

Exercise 12: Let's set a few markers and see how they work. Move the cursor to the "first word of first line"," then set a marker there. Type:
mw
Now move over to the word "last word of first line" and set a marker by typing:
ma
Next go to "last word of last line" and type:
ml

Exercise 13: Move to the first line of the document, then jump among your markers by typing the following:
'w
'l
'a
'l
'a
'w
Lastly, try to jump to an undefined marker by typing:
'c
vi just beeps at you because it doesn't know where to jump.

Exercise 14: Now you have finished the exercises in this section. Enter:
:wq RETURN
to write your changes and quit vi**.**

Exercise 15: First start vi, then try these commands. Move to the line 3 after line 11. Now you are ready to copy Type:
:3m11
and press RETURN to finish the command.

Exercise 16: This exercise performs a standard copy-and-paste. It uses the automatic buffer and does not use any markers. Move to the first line, then yank three lines of text by typing:
3yy
Nothing appears to happen, but vi has put the first three lines into its automatic buffer. The message "3 lines yanked" appears at the bottom of the screen. Now move to the last line and paste the lines in place. To do so, just press:
p

Exercise 17: Try a simple search. Type:
/and
and press **n** and **N** a few times to see where the cursor goes (explain).

Exercise 18: Try some of the search strings of the following combinations:
/ \<s.o words starting with s, any letter, then o (shook, slow)
/ \<h.*s\> words starting with h and ending with s with any number of characters in between (heavens, hills)
/o.[rtk] words containing o, any letter, and r, t, or k (bolt, foot, poor, shook).

Exercise 19: Replace the word "and" with "or" in sample file using
: *line1, line2*s*/ oldstring / newstring* command.
You can combine the global and substitute it in the following manner:
:g/*oldstring*/s//*newstring*/g

Exercise 20: Save and exit from vi editor.