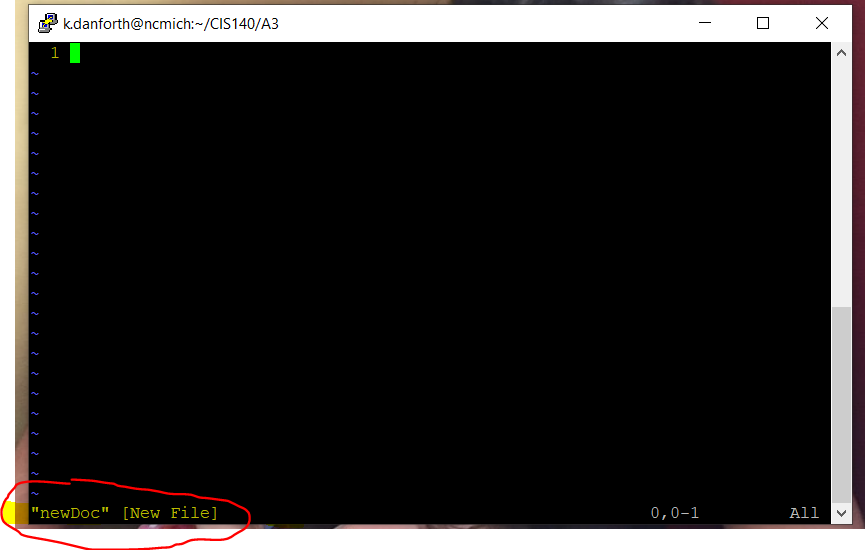
**This tutorial will show you the basics of how to use the VIM text editor from the Linux terminal.**

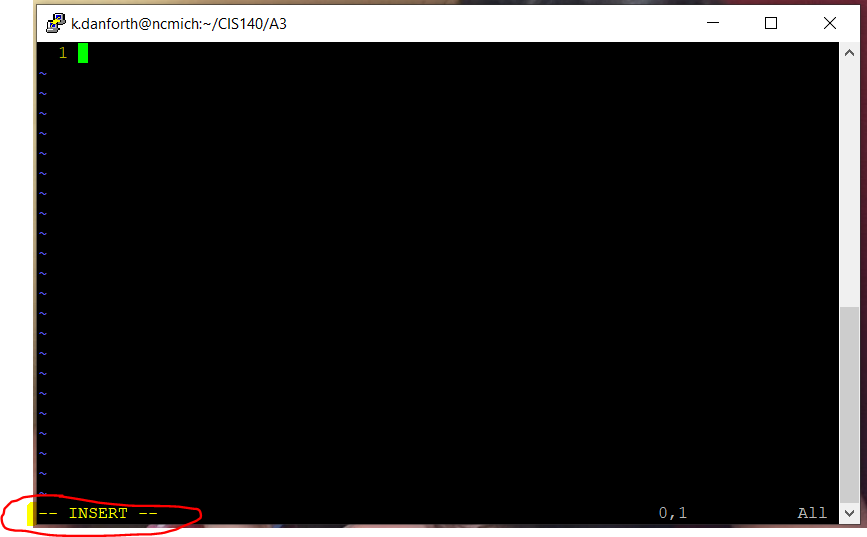
The first command to open the editor is “vim” <filename>. Below is an example of this command. If the file exists in the current directory, vim will open the file in the editor. If the file does not exist, then the file will be created, and it will be opened in the editor. In this example the file will be created.

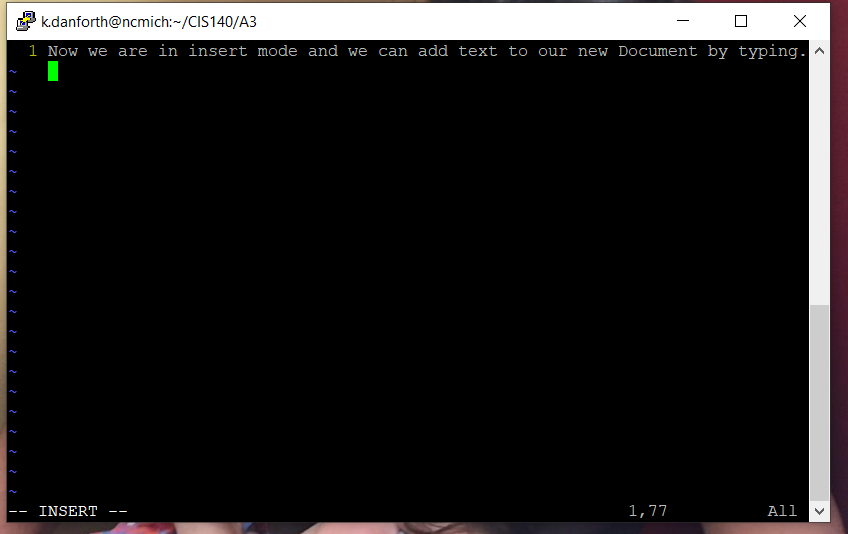
When the editor is opened the terminal will be shown as the picture below.

When a document is first opened in the editor you can see the name of the file in the lower-left corner. The editor has 3 major different modes normal, insert, and visual mode. The visual mode also has another variation visual line mode. When the editor first opens it is in normal mode.

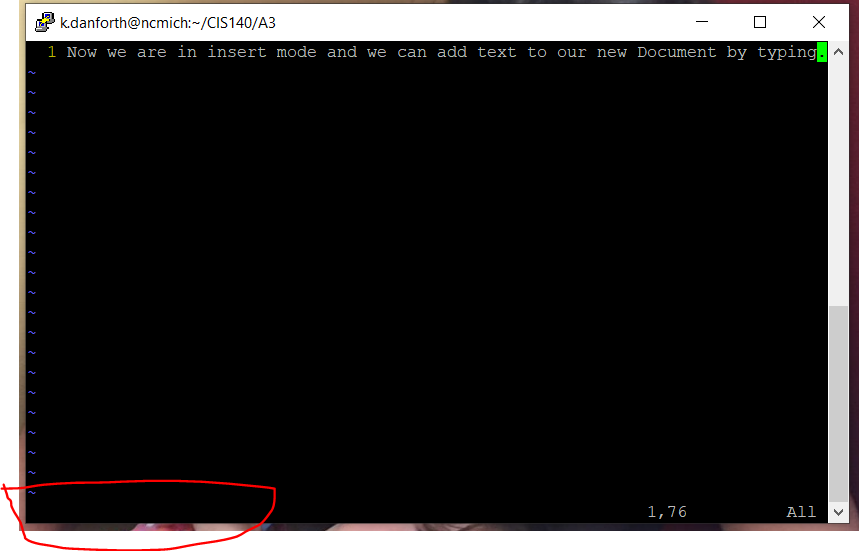
Normal mode is the default mode of the editor although it is not the mode used to insert text. In normal mode, you can use certain keypresses to enter commands to do specific things and move to other modes. Moving the cursor in normal mode is done using different keys “h” one character left, “j” moves one row down, “k” moves one row up, and “l” moves one character right. These movement keys can be combined with numbers in the front to move a certain number of times. From nomal mode you can move into insert mode using keypress “i”.

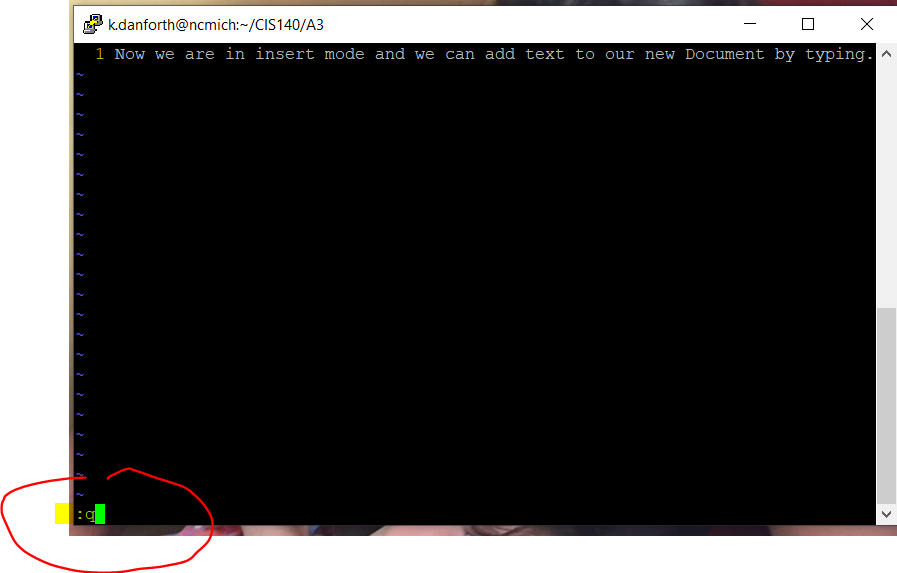
In insert mode is where you can enter text into the document.

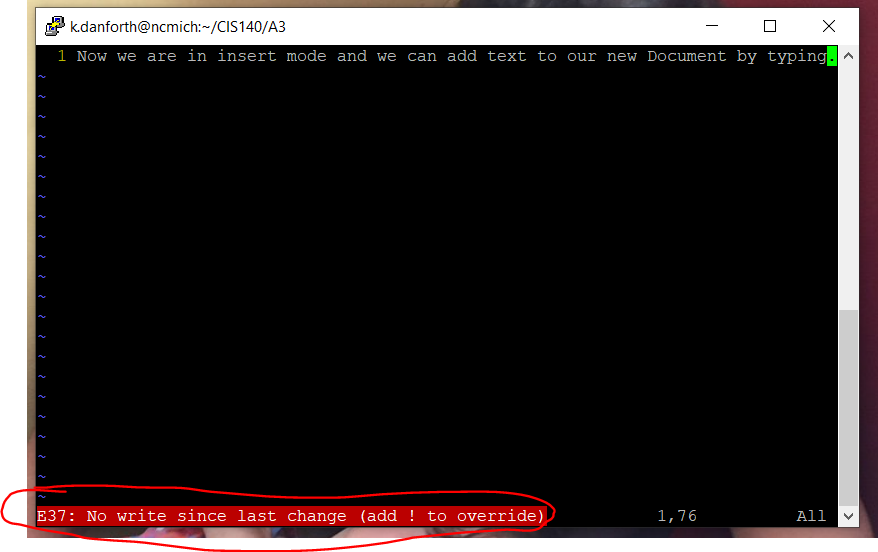
To enter text into the document we will need to enter insert mode. This is done by entering “i” while in normal mode. After “i” is pressed the bottom of the terminal screen should show INSERT indicating that we are in insert mode. As seen above. Now that we are in insert mode we can begin to type on the keyboard and enter letters into the document.

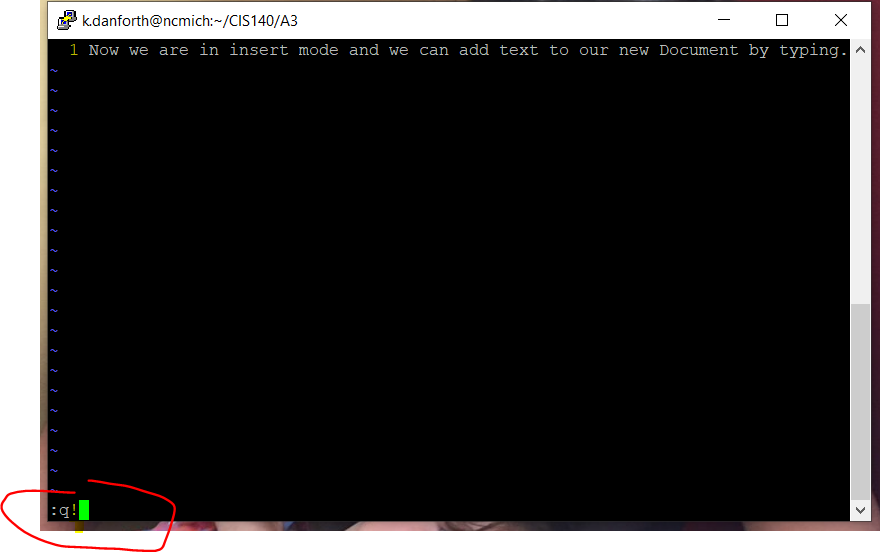


To close the document editor, we must go back into normal mode. This is done by pressing the esc key on the keyboard. Normal mode is indicated by an empty line at the bottom of the terminal screen. You can see this in the next screenshot below. This area is where commands are entered.

To quit we want to enter the command ”:q” in normal mode.

This command will not work in our situation since we have edited the document and the changes will not be saved. So, an error message is shown below.

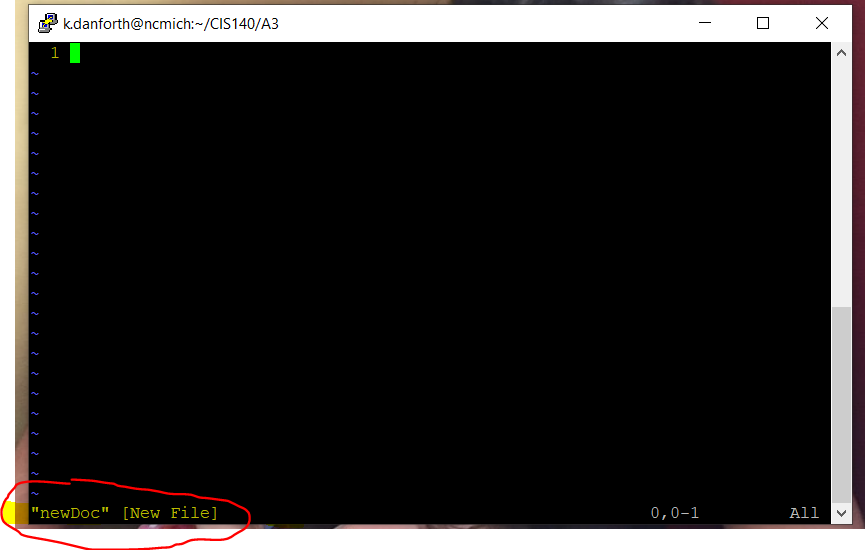
This message shows an option to override ”!”. I will demonstrate this below.

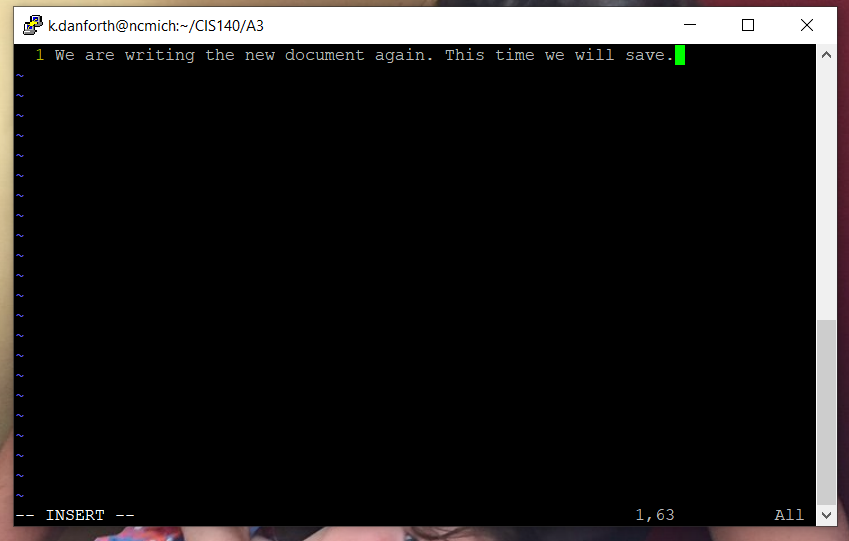
When the command ”:q!” is entered the editor quits and the changes to the file are not saved.

As seen here, in our case, since this was a new file being created for the first time, if we look using the” ls” command you can see that the file was not even created at all.

Let us create our file again using the command “vim” <filename>.

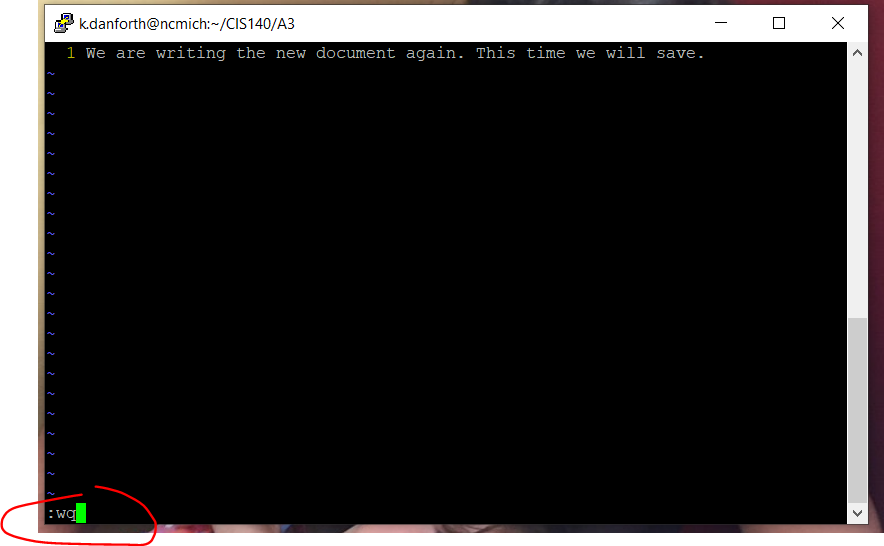


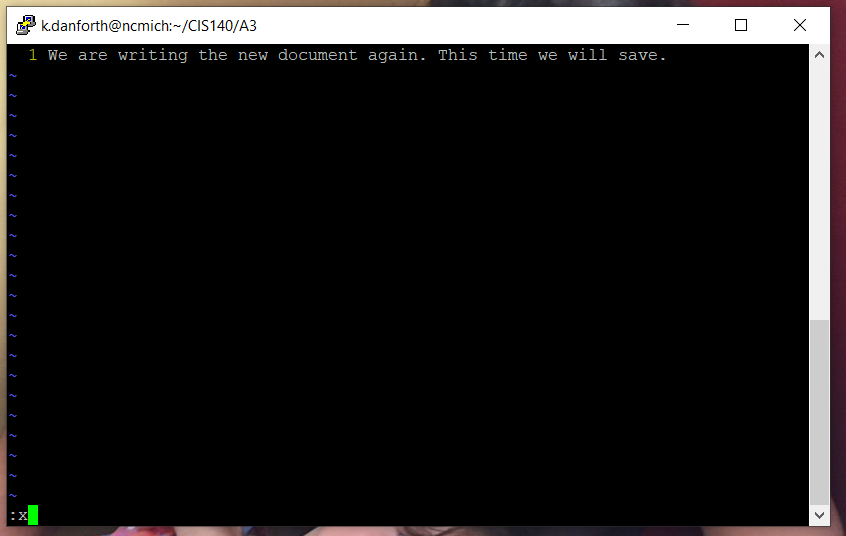
Enter ”i” again to enter insert mode and we will type into the new document. This time we will save the document when we quit.



We will need to enter back into normal mode by pressing the esc key.

Once in normal mode we can type one of two commands to save and quit. We can either use a combo command that uses” w” (for write) to save and” q” to quit. Making the command ”:wq”. (The command” w” can also be used alone to just save the changes to the document without quitting:”: w”).



The other command that can be used to save and quit the editor is ” :x”.

We can now type the” ls” command and see that the file has been created and saved.

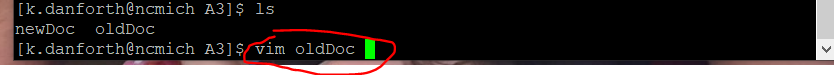
We can use the same ”vim” <filename> to open the file in the editor again.

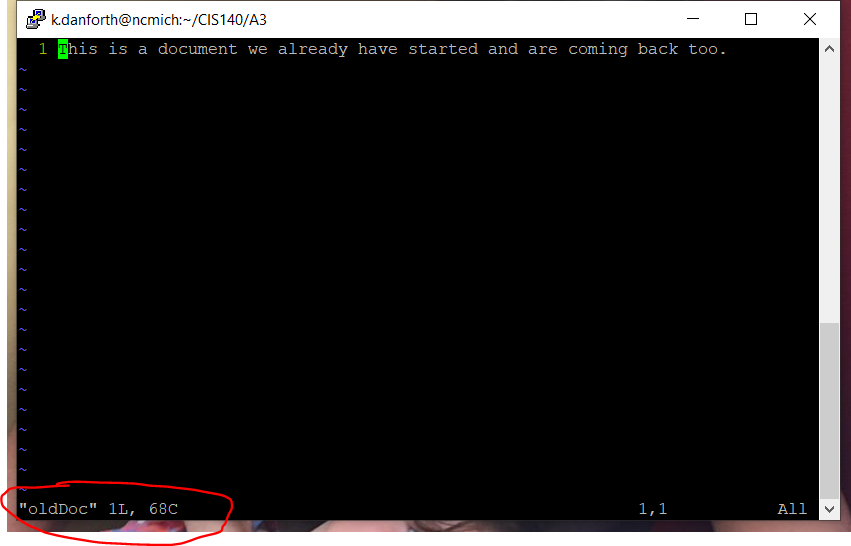


And here we can see the changes to the document were indeed saved.

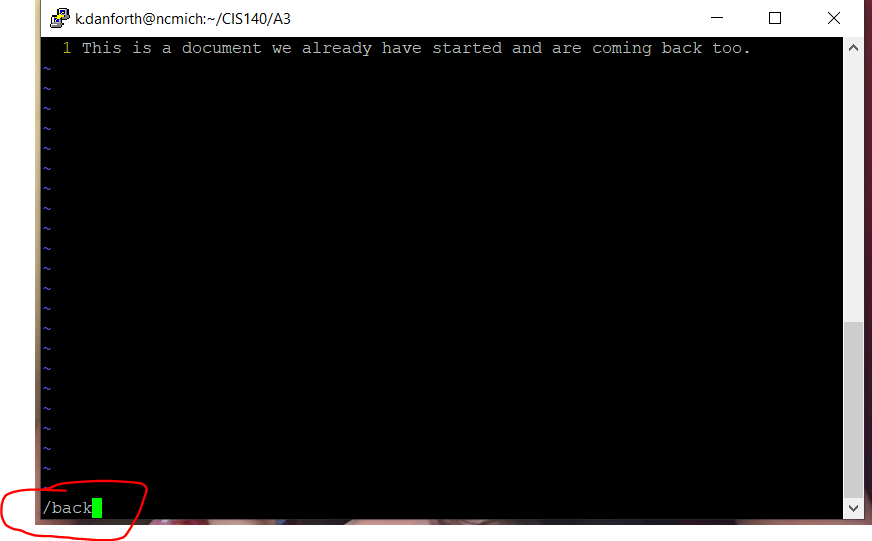


You can see below that there is another document in this folder named “oldDoc”. The same “vim” <filename> command can be used to open any text files that you have access to on the machine.

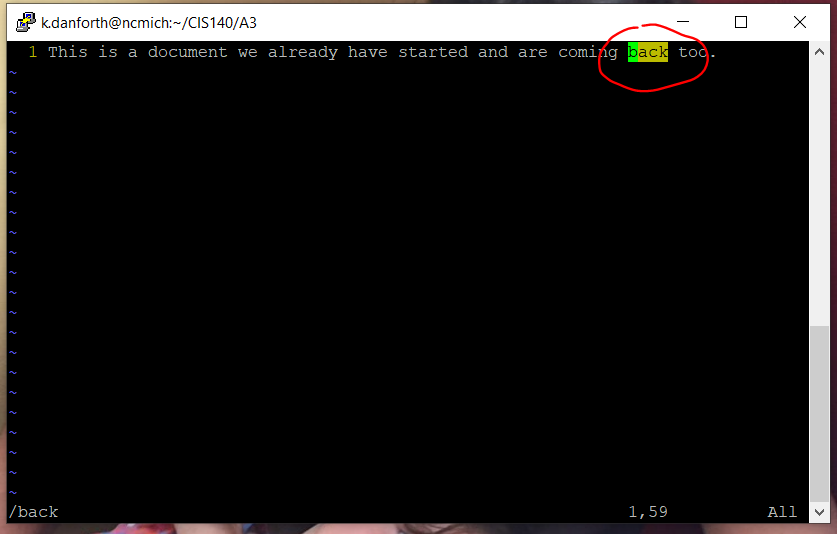


When you open a file that had been created previously you get some more information at the bottom of the terminal window. It gives you the name of the document that is open plus, the line count, and the character count.

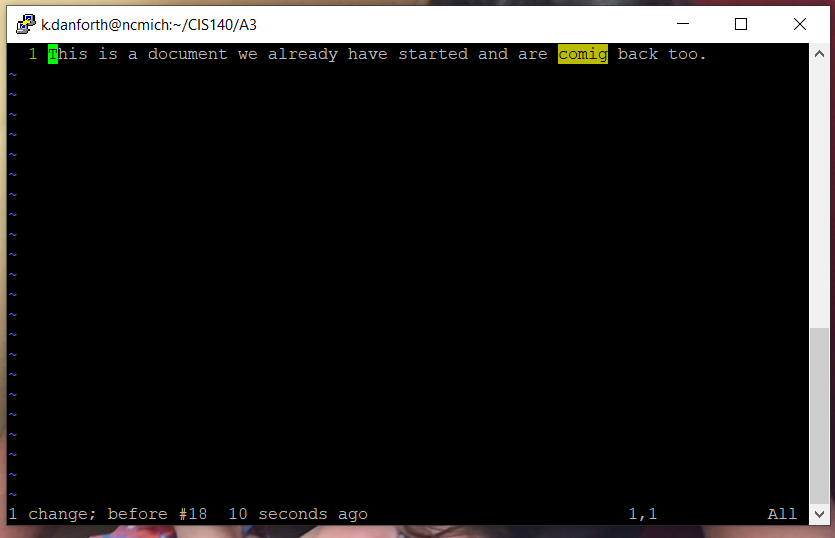
Another useful cammand that can be used from normal mode is the “/<string>” cammand. With this command we can search for specific text in our document. See below we can use this commmand to find the word “back”.



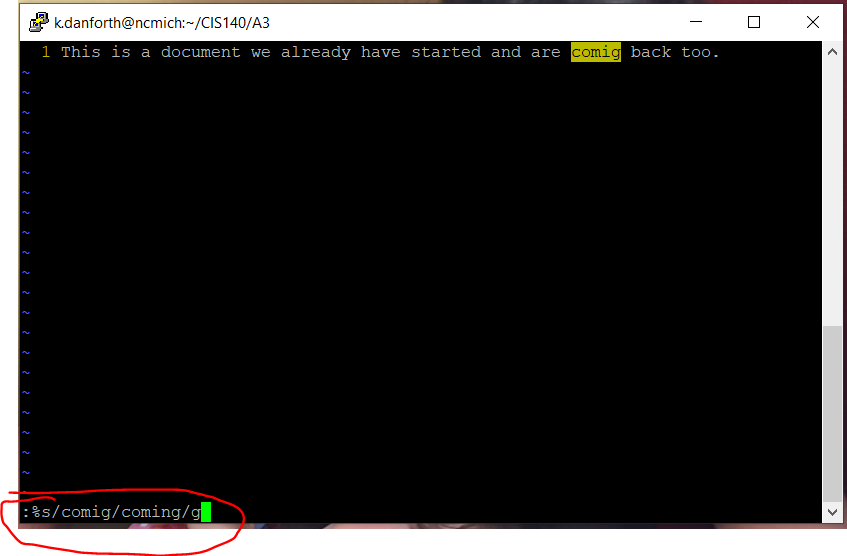
When the word “back” is found it is highlighted.



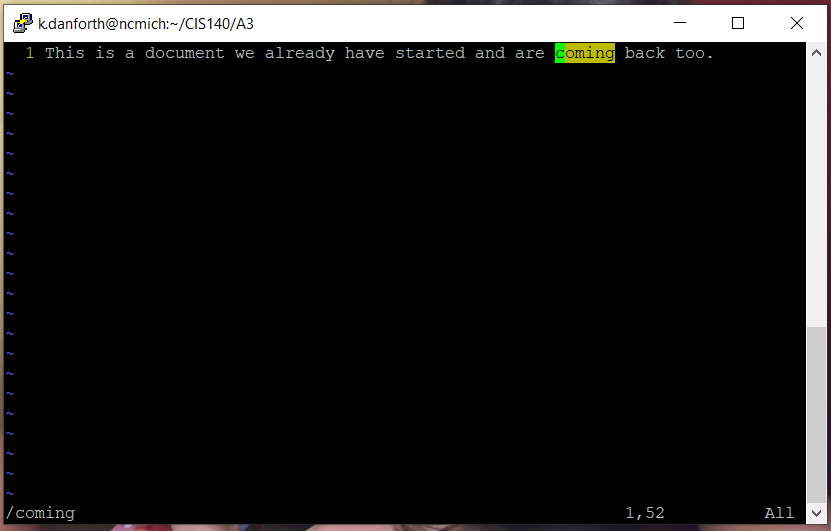
Another useful tool is “replace” which also can be used from normal mode. Replace is perfect for fixing typos or common spelling errors.



Below you can see where we have a typo. We can use the command “:%s/<wordToReplace>/<replacmentWord>/g(for global)>”. This will find the <wordToReplace> and replace it with <replacmentWord>.

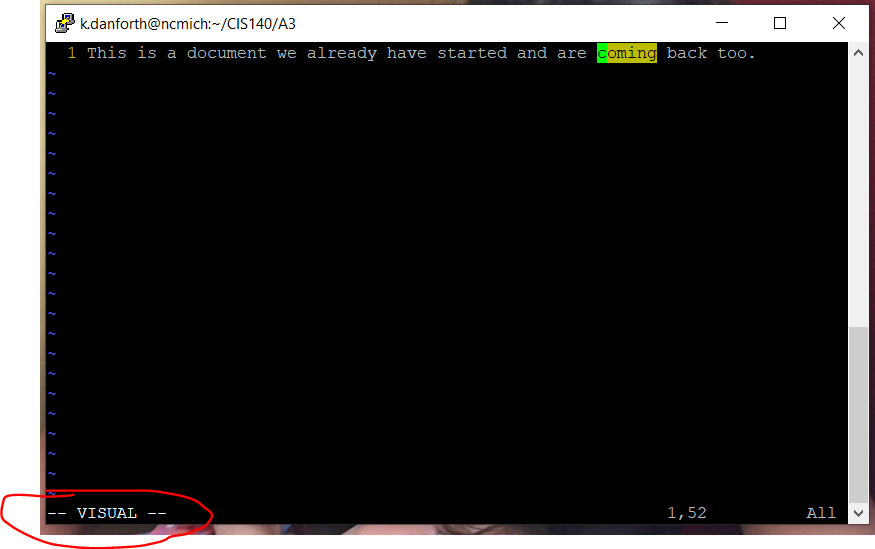


You can see here that “comig” was replaced by “coming”.

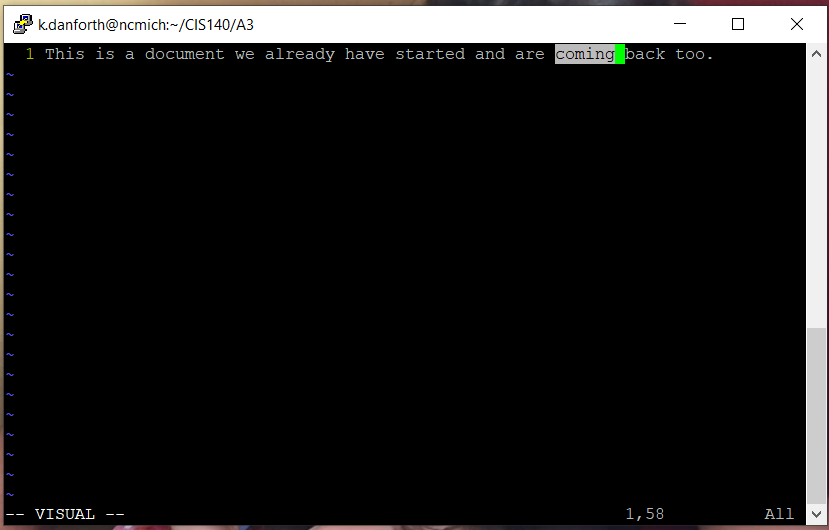
Command”/coming” to highlight the word.

Another way to highlight a word is to enter visual mode the final major mode of the Vim editor.

To access visual mode type “v” from normal mode.

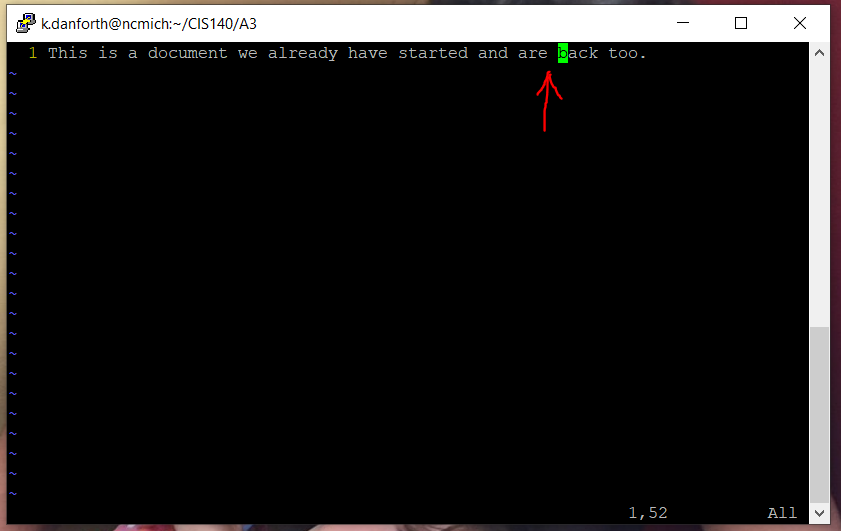
Here at the bottom of of the terminal window you can see – VISUAL -- indicating that we are in visual mode.

Visual mode is mainly used to select and highlight text. You can do this by entering visual mode and moving the cursor. If you enter visual mode with the cursor on the letter “c” of the word “coming” you can use the arrows to move the cursor and highlight in either direction. In our case I will move to the right in order to highlight the word “coming”.



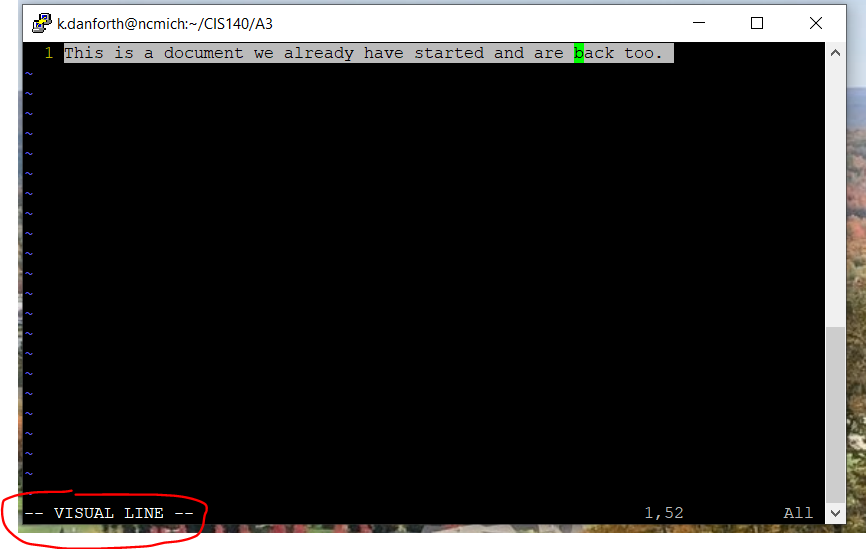
Once Highlighted different key presses can act on the selected text.

We can press “d” for delete. And you will see the word coming has been deleted.



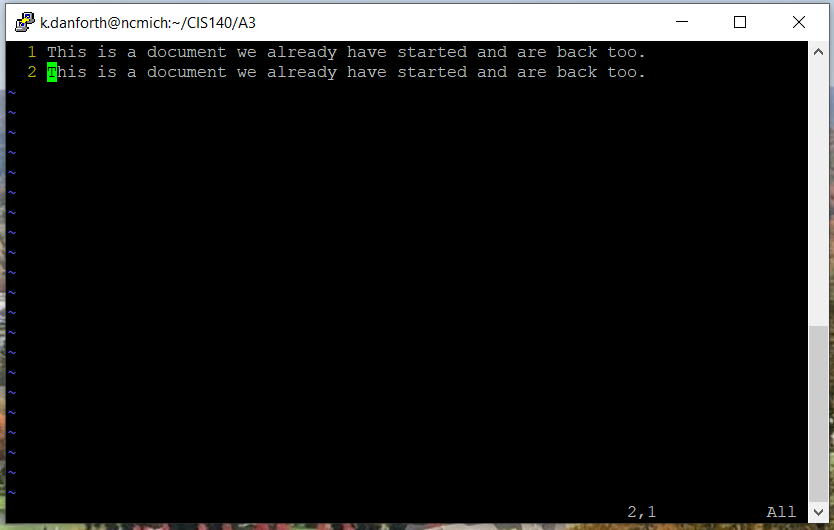
The major mode, visual mode has a minor variation that will select whole lines of text rather than just one character at a time. This mode is called visual line mode and can be entered from normal mode by typing “:V” where the “V” is capitalized.

Here we have entered this command and visual line mode. This action has highlighted the whole line of text that the cursor was located on.

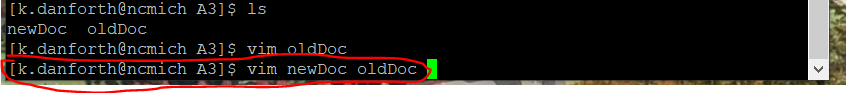


Another keystroke that can be used in visual mode and visual line mode is “y” for yank (copy) and then “p” for paste.

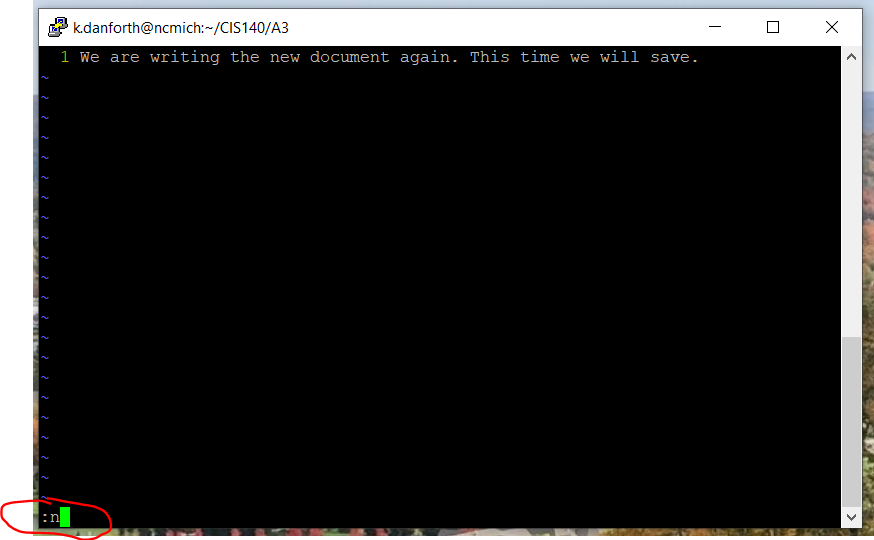
If we use “y” to copy the line we can then use “p” to paste it onto the next line of the document.



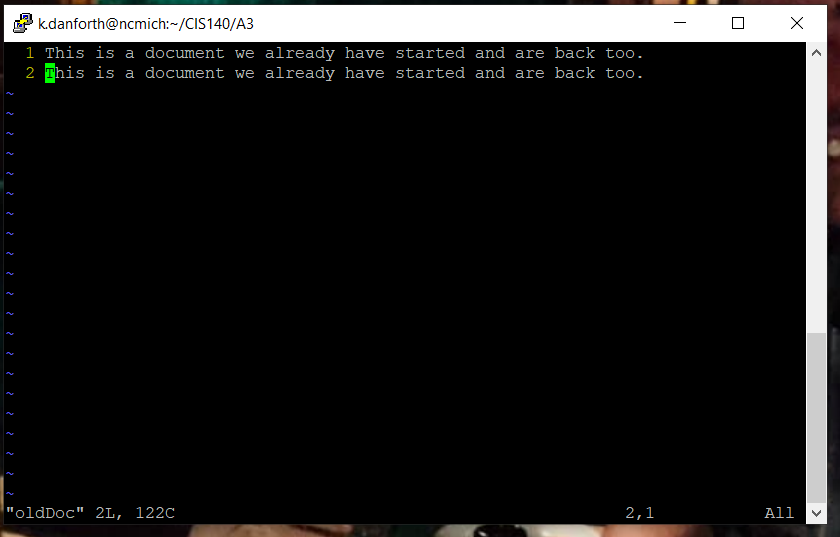
You can also open two docs and move back and forth between them. You can do this by using the command “vim <filename> <filename> … ...” with as many <filename>s as you should need to open.



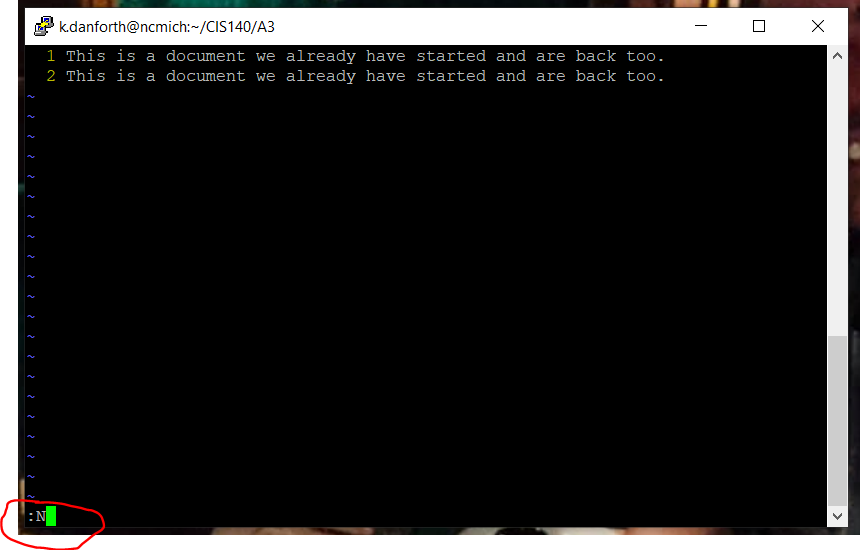
In normal mode you can move to the next doc using command “:n”.



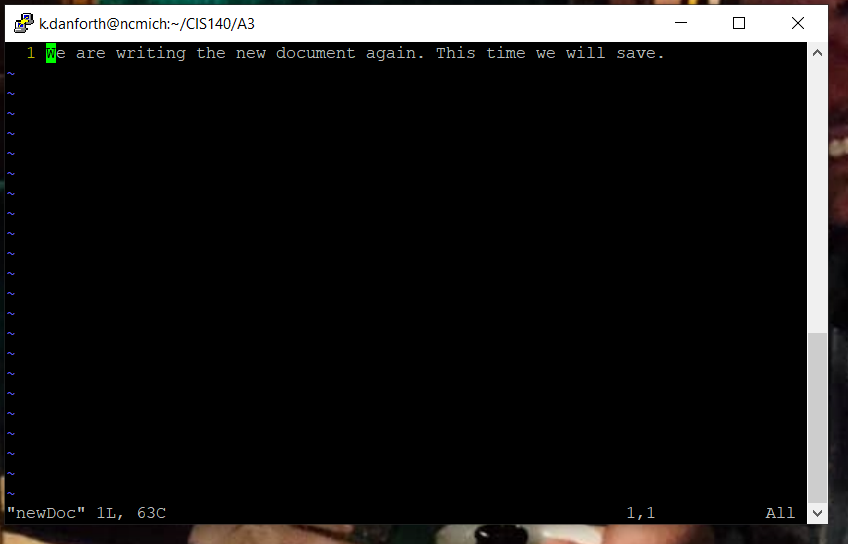
See here we have moved to the next open document in the editor “oldDoc”.



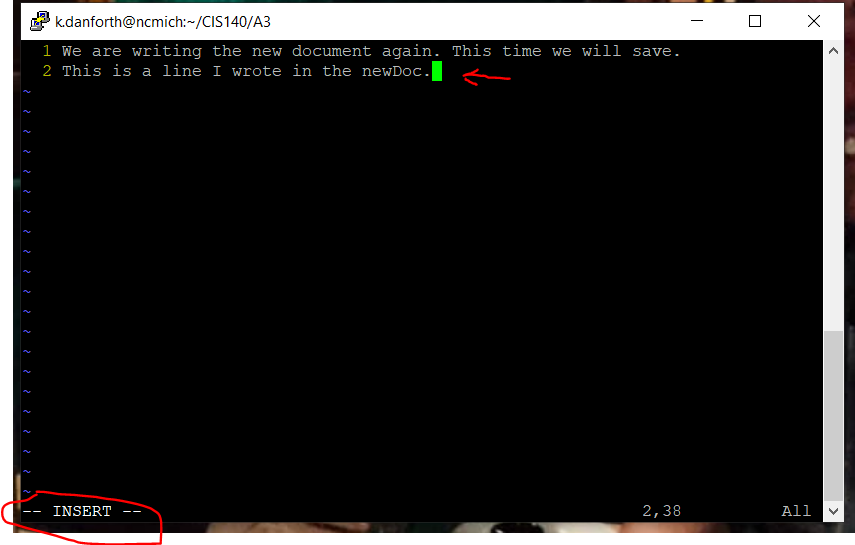
And then you can use command “:N”, where the “N” is capitalized, to go back to the last document.

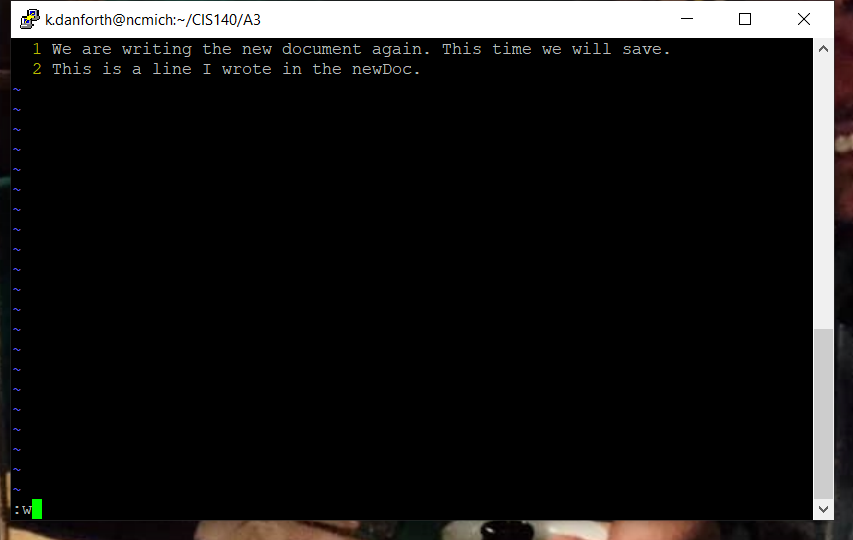


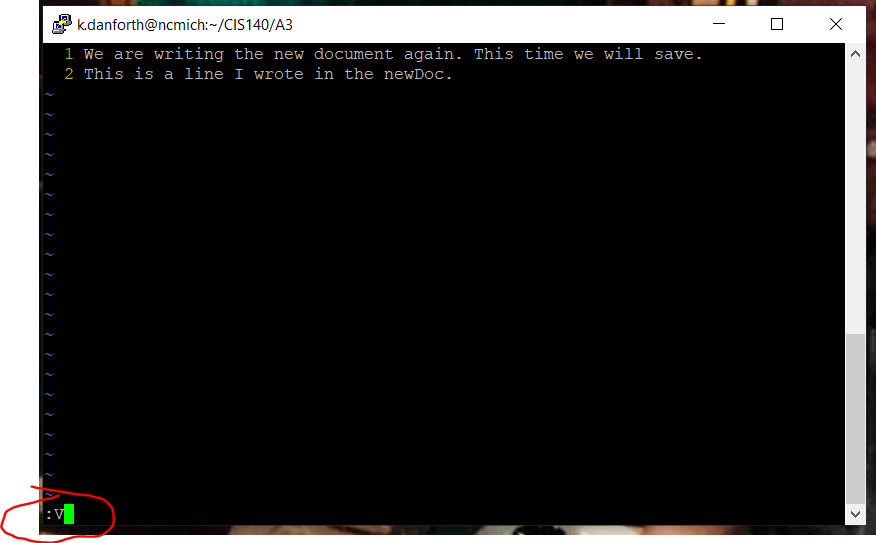
Here we are back to the first open document ”newDoc”.



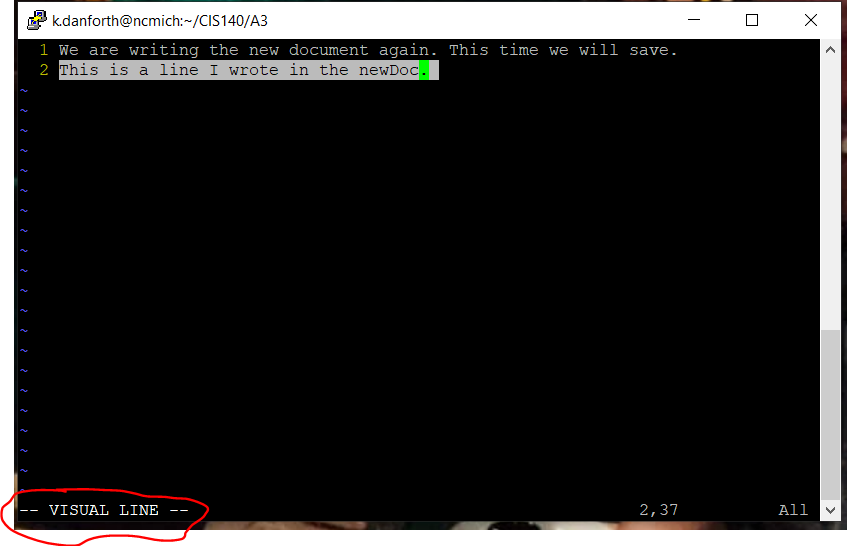
We will add another line to the document “newDoc” by entering “i”, going into insert mode, and then typing on a new line.



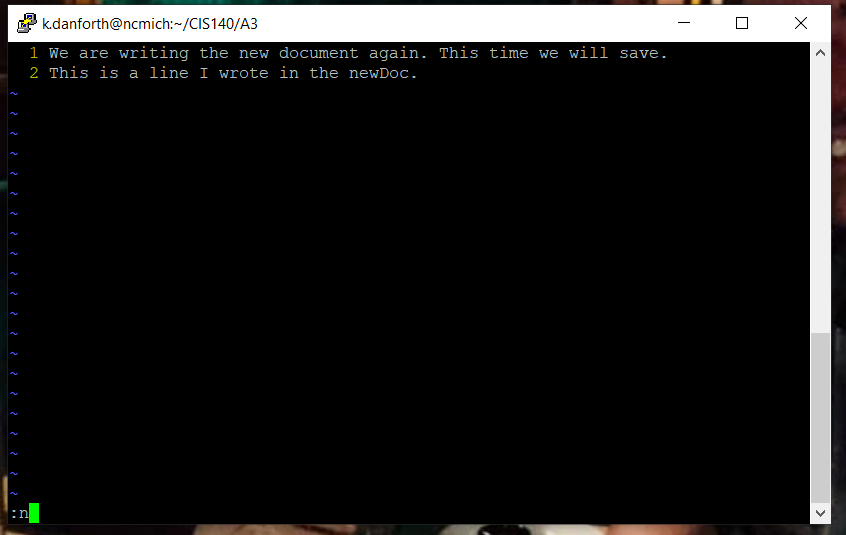
Next we will use the esc key to go back to normal mode then enter “: w” to save and “:V” to go into visual line mode.



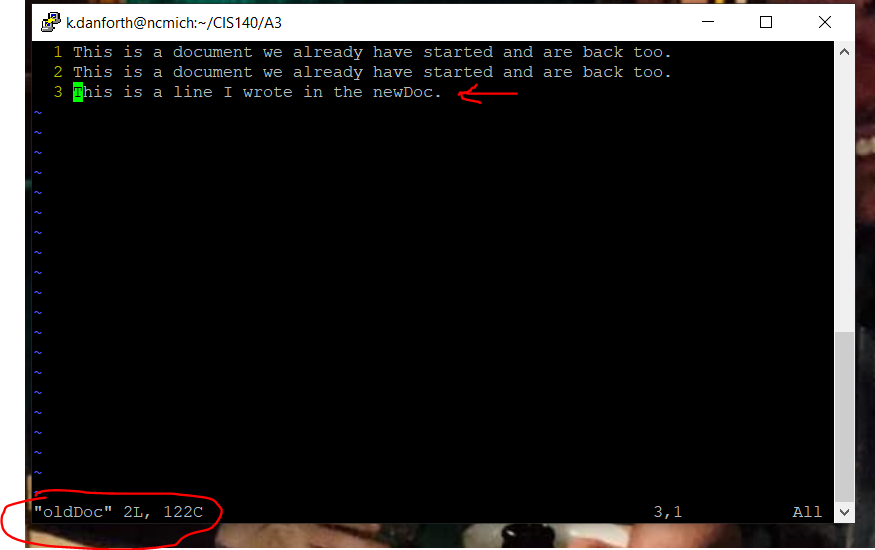
Once we enter visual line mode the line where our cursor was located will be selected.



Press “y” to yank the line, then use esc to go back into normal mode. When in normal mode use “:w” to save the changes and then use “:n” to go to the next open document in the editor, “oldDoc”.



Then press “p” and the line that we had “yanked” from the first open document “newDoc” is not pasted to the second document “oldDoc”.



Now we can enter “:wq” to save and quit the editor.

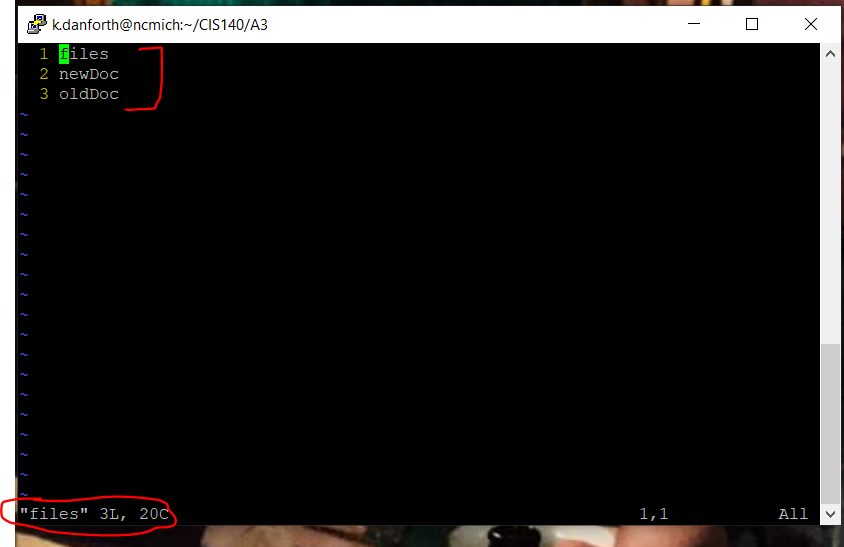
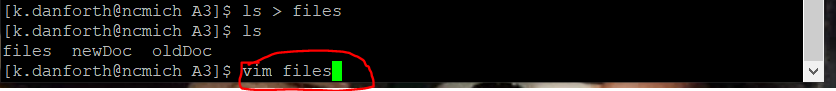
Another useful tool is the ability to insert the output of a Linux command from the command line into a vim file using the redirector. Using the “ls” command and the redirector (>) we can enter the command “ls > <filename>.” In the image below we will create a file called “files” in the current directory if does not exist. If it already does exist it will be overwritten. This file will be filled with the redirected output of the “ls” command of our current directory.



Another lone” ls” command reveals the new file was created.



Command “vim files" will open the file named “files”.

We can now see the text added to the file,” files” was the output from our” ls” command. A list of all the files in our current directory.

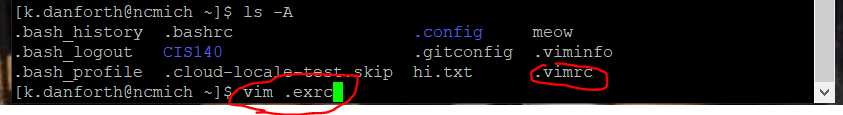
The vim editor does also have a way to set some preferences.

Linux machines have files that have a “rc” at the end of the filename examples are “.vimrc”, “.exrc” or “.bashrc”. The “rc” is derived from the /etc/rc files that are used at the startup of most UNIX systems. This “rc” relates to the phrase run commands and when used as the suffix of a file. That file is usually a run-command start-up file that is run at the startup of a specific program. This “start-up, run-command file” may contain many different commands that can be run at the startup of a program to cause some effect, or to set preferences for the program. These files are normally kept in the home directory.

In our case, the “.vimrc” file is the run-command startup file for our vim editor. We are also going to talk about the “.exrc” file that is the run-command startup file for the vi editor. The vi editor is most standard across Linux machines. Where the vim editor is a more advanced and powerful vi-like editor but is not built-in on all machines. The vim editor will use the ”.exrc” file on startup if the “.vimrc” file is not available as we will see.

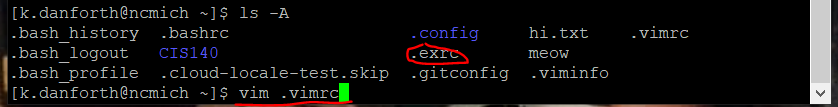
Once we navigate to our home directory using the command “cd ~”.

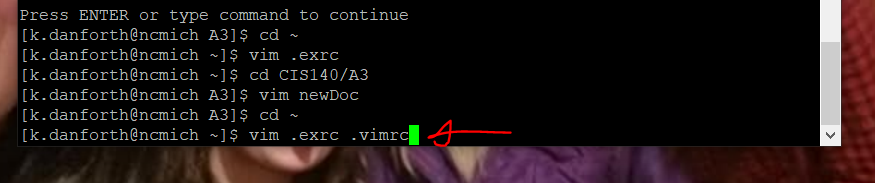
If we use the “ls” command with the “-A” flag, we can see the hidden files included in our home directory.

Here you can see we have a “.vimrc” but no “.exrc”. They are much the same thing and vim will use “.vimrc” first if present but will use “.exrc” if it is present and ”.vimrc” is not present in the home directory.

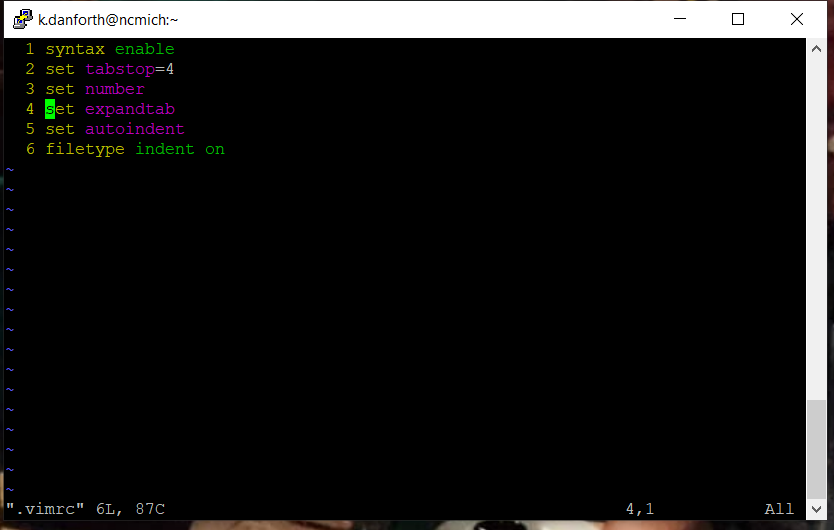
So, to create the “.exrc” file we will use the “vim <filename>” command again where <filename> will be “.exrc”. As seen above.

Another “ls -A” on the home directory shows that the “.exrc” file was created.

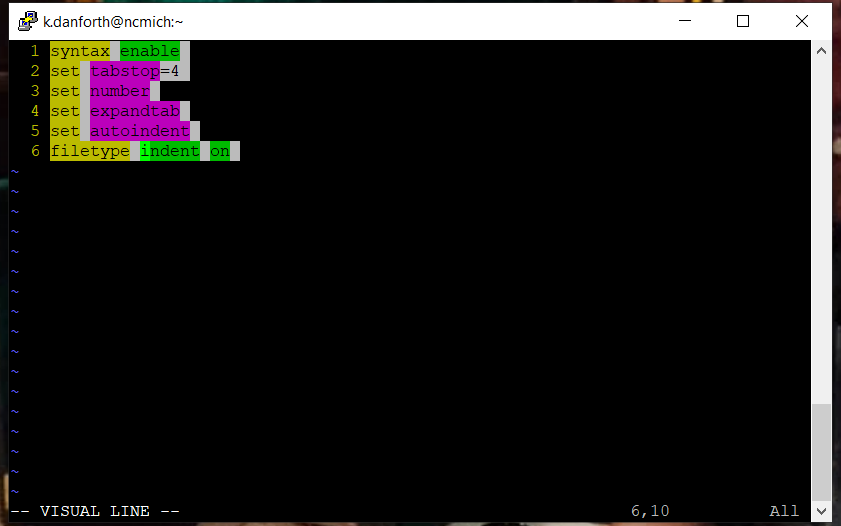


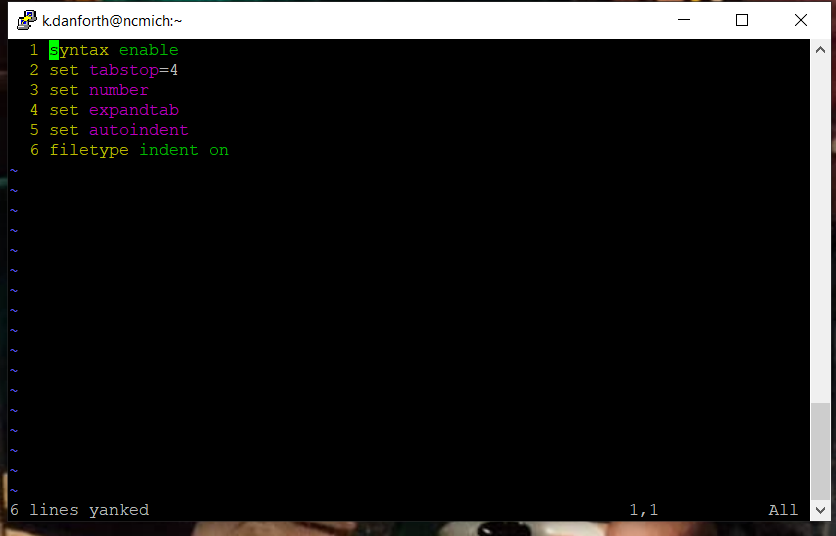
If we run the command” vim .vimrc .exrc” we can start our editor and open both the ”.vimrc” and “.exrc” This way, we can copy the information from “.vimrc” to “.exrc”.

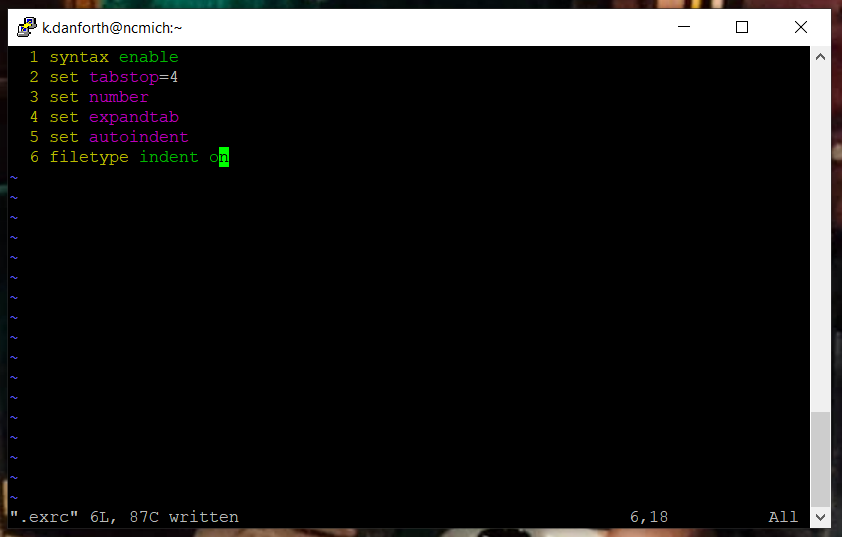
We are now open to the first file (.vimrc).



We will enter visual line mode by using the “: V” command with the cursor on the top line and use the down key to move the cursor down to the bottom of the six lines, highlighting them all.

We will use the” y” command to (yank) copy the six lines. Make sure to also use the”: w” command to save after this as well.

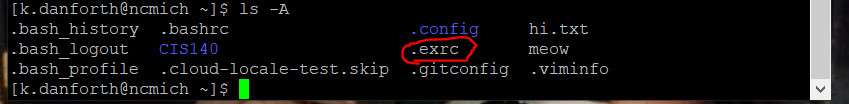
Use the ”:n” command to go to the next open file (.exrc). Then press” p” for past and your ”.exrc” file should be populated as shown below.



Now since the editor will not use the ”.exrc ” file unless there is no ”.vimrc” file present in the home directory we must remove the ”.vimrc” file. We do this with the” rm <filename>" command.



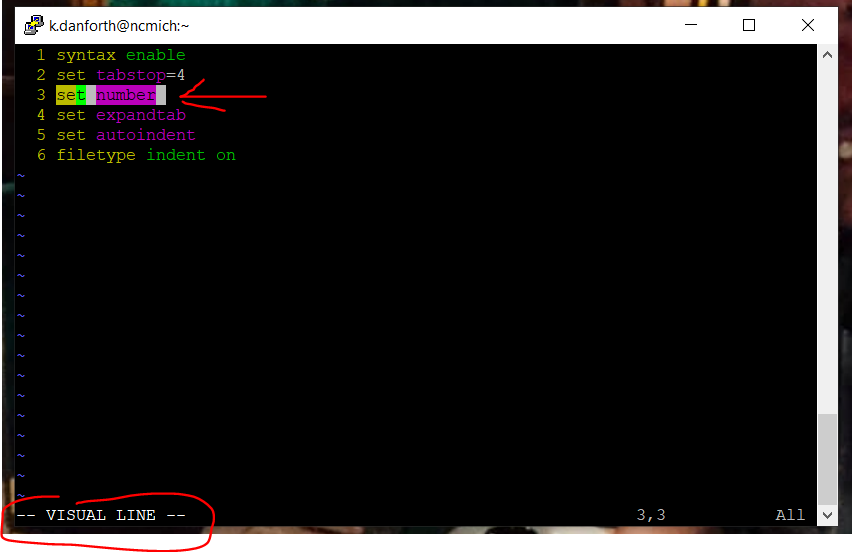
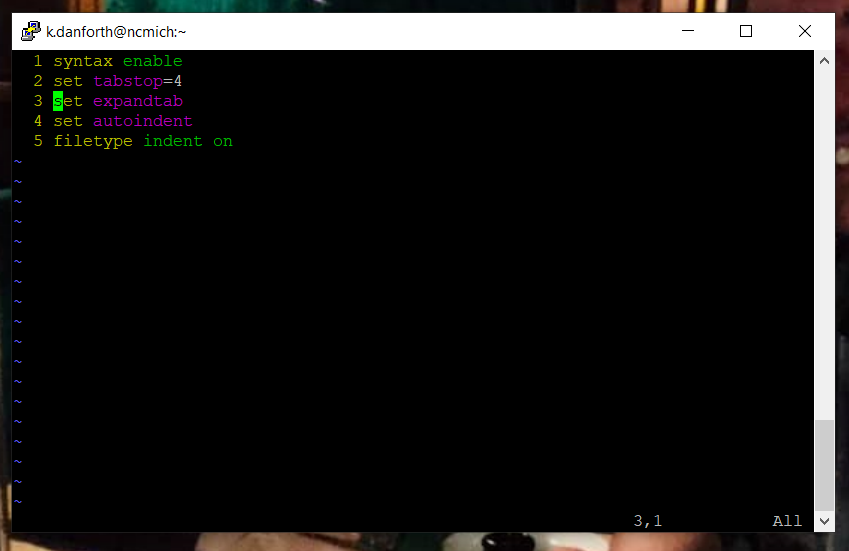
You can see here by using the “ls -A” command it has now been removed.



Then we will open the “.exrc” file in the vim editor using the command “vim <filename>”.



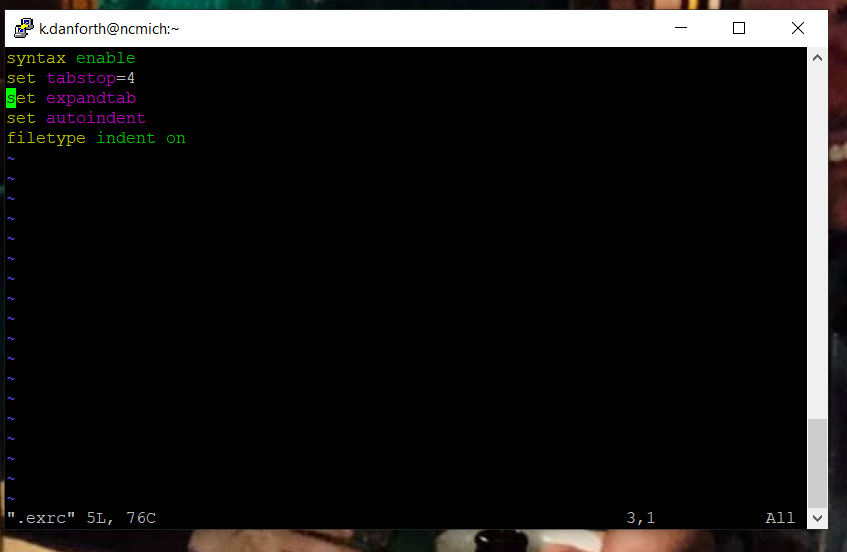
We will edit the file to see what changes are made. With your cursor on line three enter visual line mode using “:V” in normal mode.

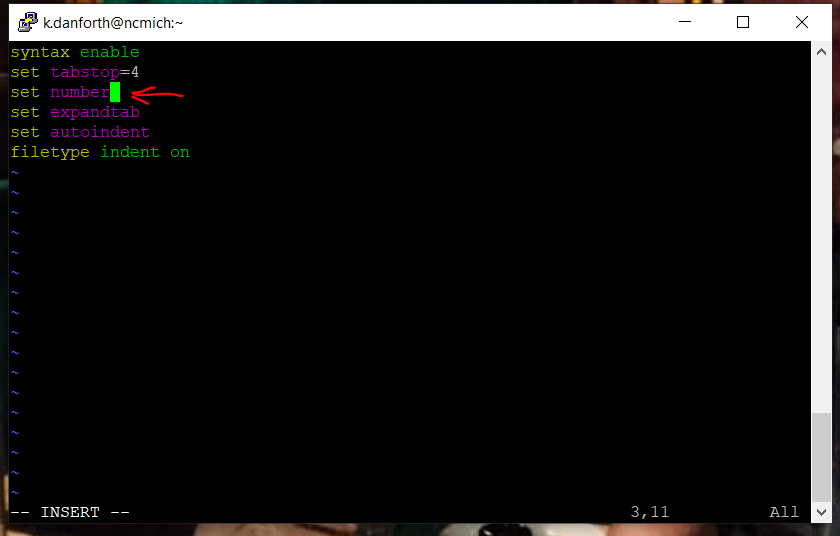
Press” d" to delete the highlighted line that says” set number “. 

Then use the esc key to go back to normal mode then enter the ”:wq” to save and quit the editor.

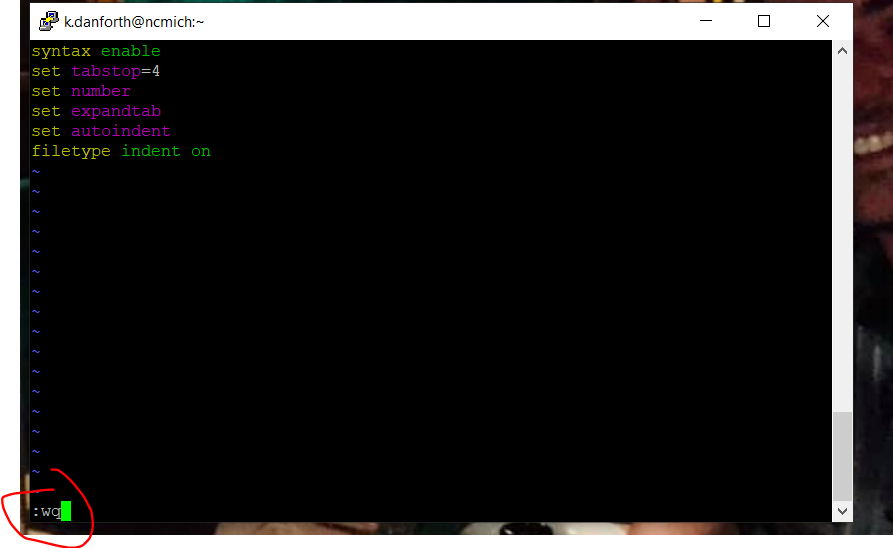
After that open the ”.exrc” file again by using the ”vim <filename>” command.

This will now show us that the line numbers are missing.

Now enter insert mode by typing ”i”. Then move the cursor to the second line and press enter to create a new, third line and then type ”set number” to reset the line numbers in our ”.exrc” file.

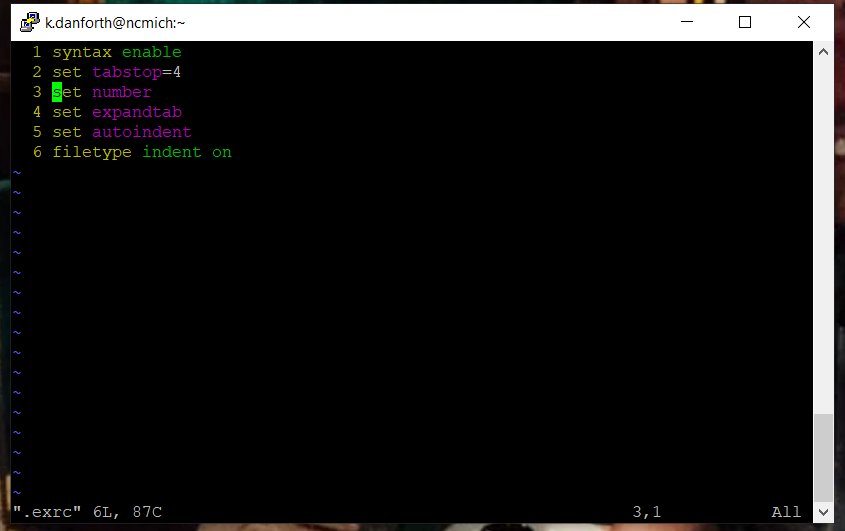


Now use the esc key to enter back into normal mode and use the “:wq” command to save and quit the editor.



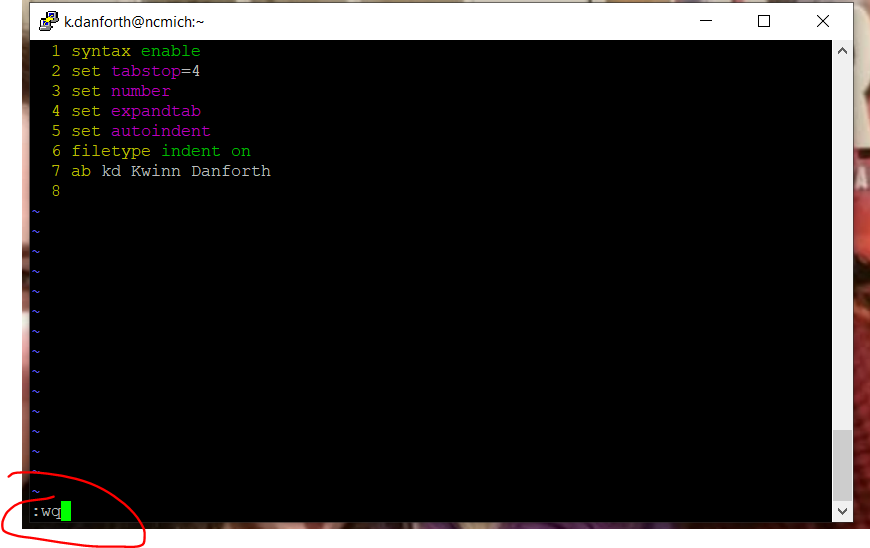
After that open the ”.exrc” file again by using the ”vim <filename>” command.

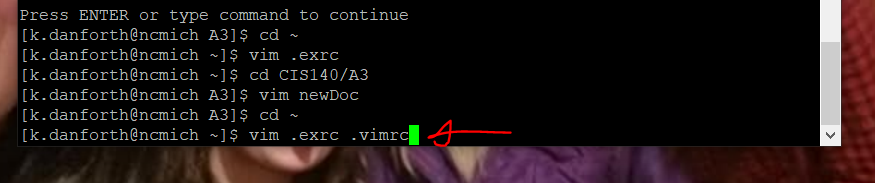
As you can see the line numbers have been returned to the left side of the terminal in the editor.



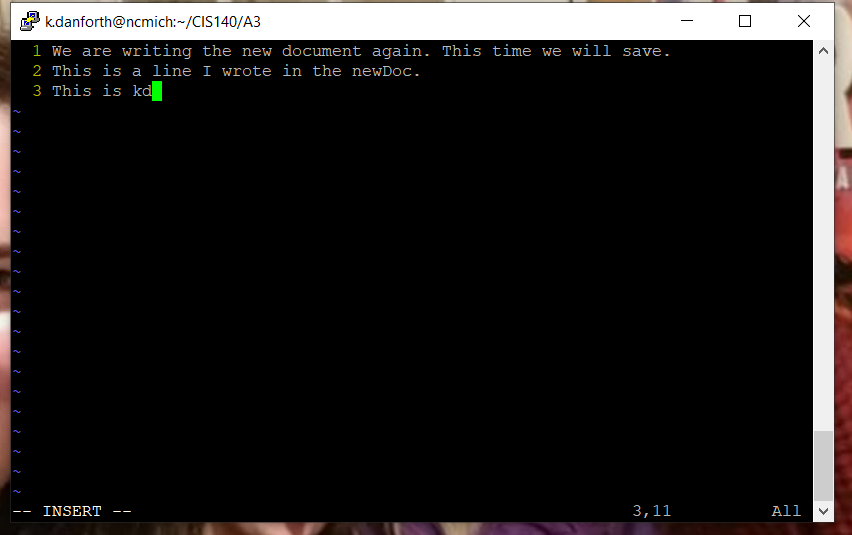
We can also add other commands to the rc file. Here we will add an abbreviation for our initials so we will be able to type our whole name by just typing our initials.

Enter insert mode by typing “i” and then move the cursor to the last line of “.exrc” and press enter to add a new line. On this line type “ab <initials> <fistname lastname>” as seen below. This command adds the abbreviation where ab is the command for abbreviation, <initials> is the abbreviated string and <firstname lastname> will be the long-hand version of the phrase. After the new line has been typed press esc to enter normal mode and enter the “:wq" command to save and quit.

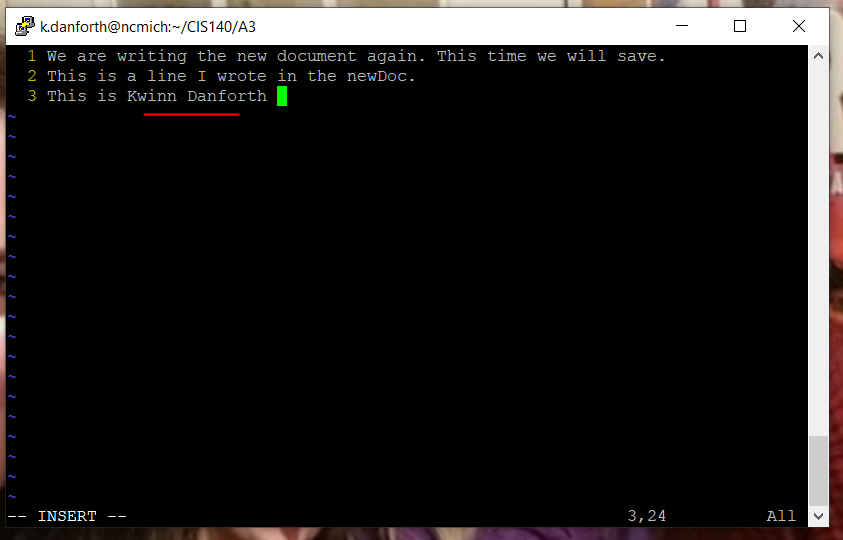
Now we will navigate back to our directory that contains the ”newDoc” file that we created earlier to test our abbreviation. We will start the vim editor by entering the ”vim <filename>” command and open our ”newDoc” file.



We will enter insert mode by typing “i” and on a new line we will begin to type “This is <our initials>”.

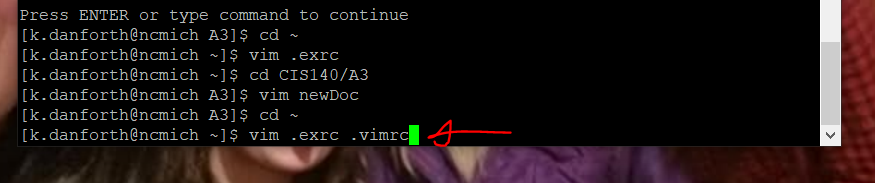


Once we hit the space bar after the last letter of our initials the editor will change the initials to our whole name. This can be useful for creating shortcuts for making typing faster.

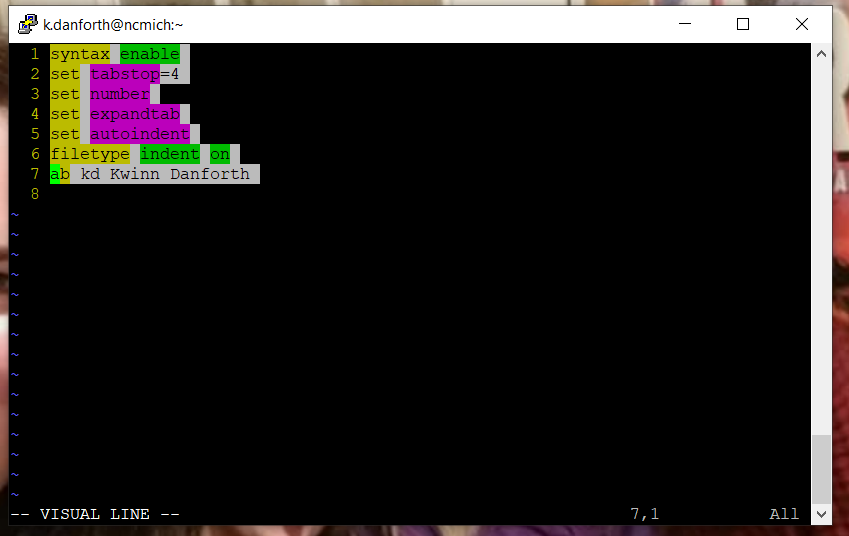


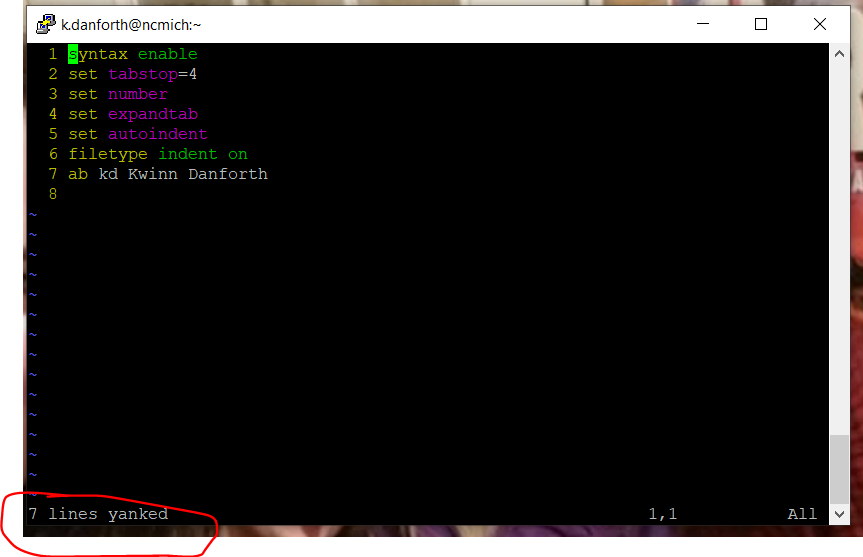
There are many other commands and preferences that can be set that as well can be kept and used in these run-command startup files. The file that is the most versatile with vim, the editor we are using, is the “.vimrc” file. So, I am going to create this file again. Copy the information from “.exrc” into it and then remove the “.exrc” file since I will no longer need it.

So, after I navigate to my home directory. I will open both the “.exrc” file and the “.vimrc” that I am going to create in the Vim editor by using the “vim <filename> <filename>” command.

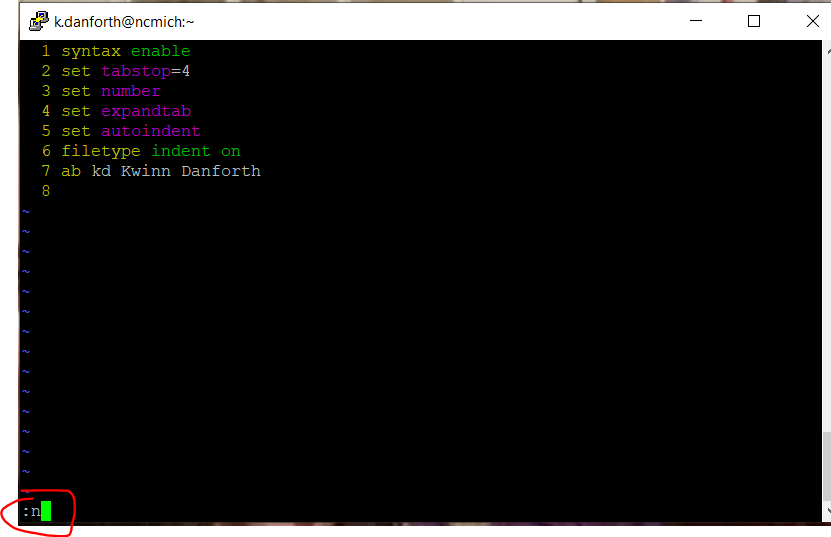


While viewing the “.exrc” file with my cursor in the top line. I will enter visual line mode using the command “:V". Then using the down arrow, I will highlight all the lines and press “y” for “yank” to copy all the lines.

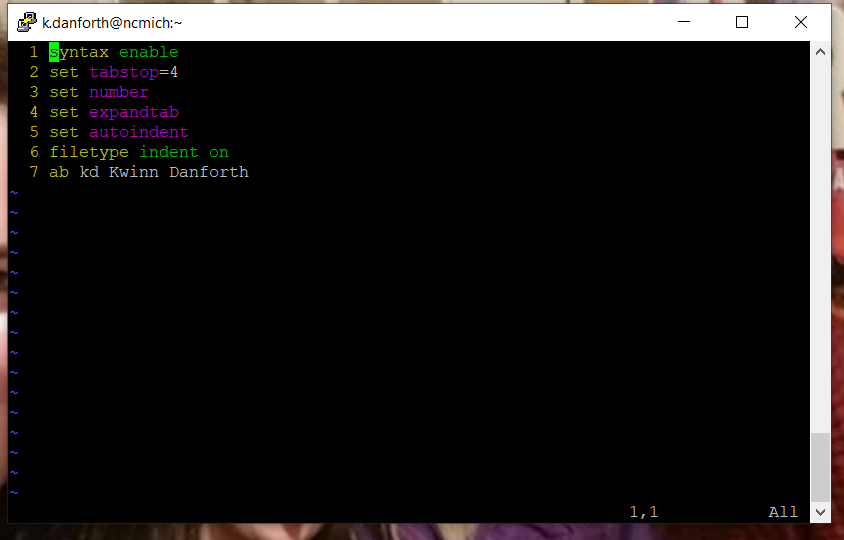




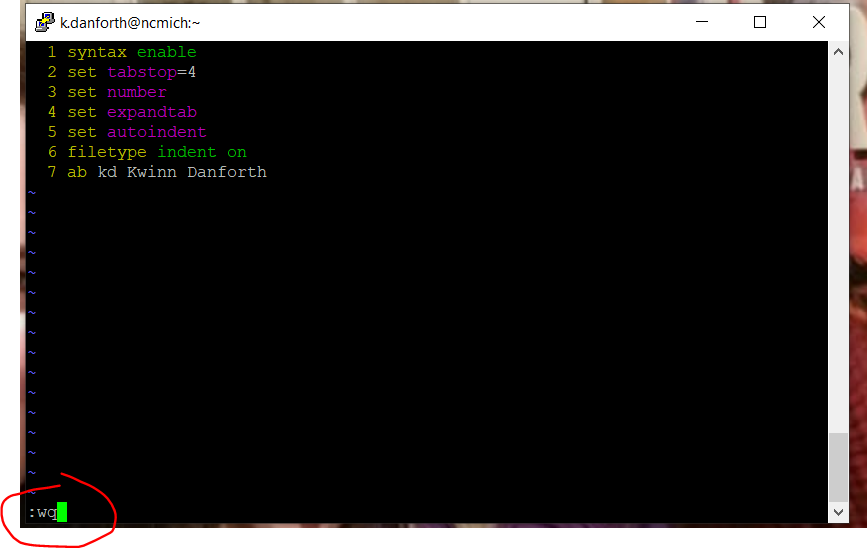
I will then use “:w” to save and “:n” to go to the next open file (.vimrc).



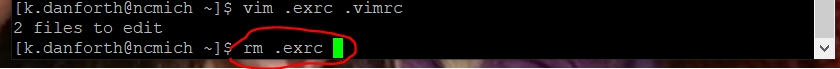
In the newly created “.vimrc” file enter “p” to paste the contents of the clipboard into the “.vimrc" file.



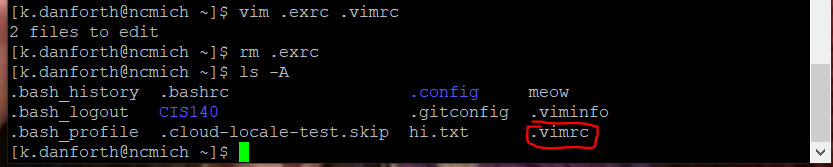
I enter the command “:wq” to save and quit.



Now I need to remove the “.exrc” file using the “rm <filename>" command.



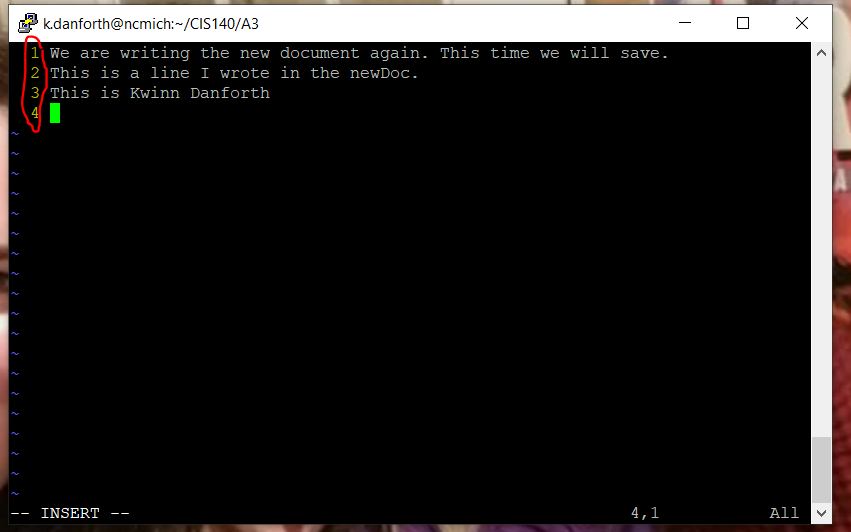
Here we can see that only the ”.vimrc” file is left in the home directory.



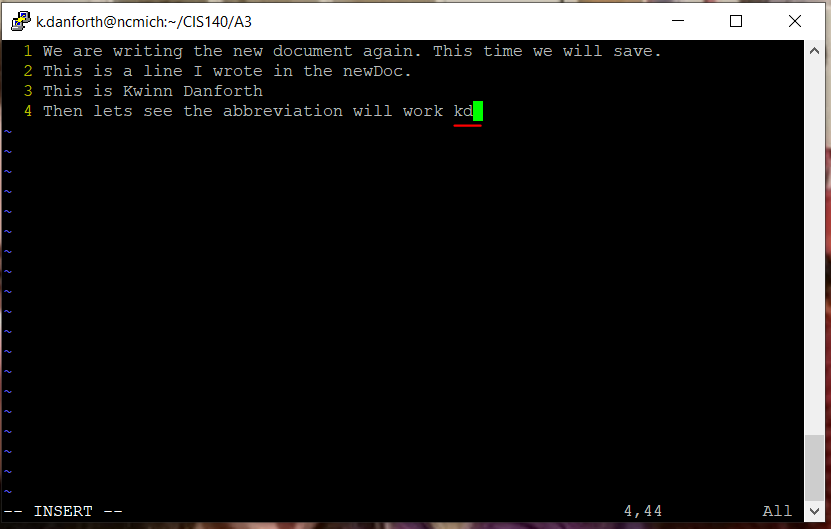
Now I navigate back to the directory with the “newDoc” file we had created and open it in the Vi editor again using the “vim <filename>” command.



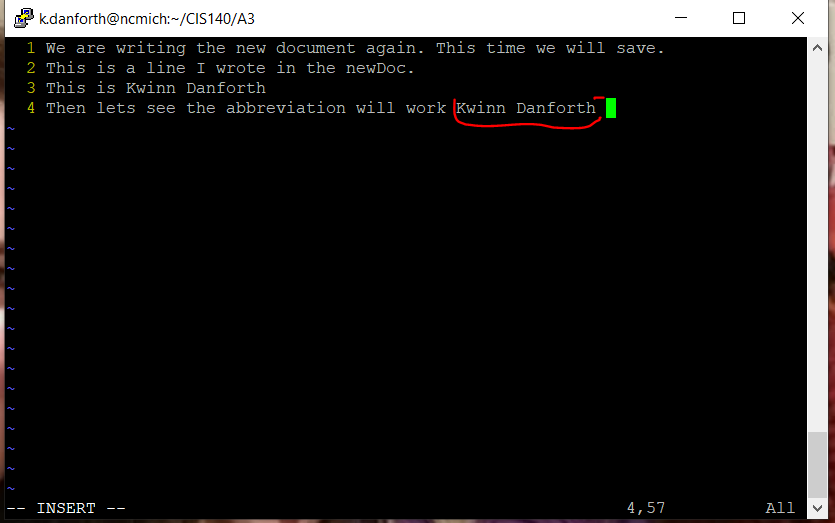
Here we can already see that the line numbers are working. Press “i” to enter insert mode and test our abbreviation again.



When we type our initials again...



And after the space, look at that, it works.



This shows how both the “.exrc” and the “vimrc” files can be used to do the same things with the vim editor. Although I am sure that there are commands that will work for vim editor within the “.vimrc” file that may not work in the vi editor within the “.exrc” file since the vim editor is said to be more advanced.

That is all for this tutorial but the vim editor does have much more to offer.