## Exercise: Using Text Editors in Linux

**What You'll Do**

* In this lesson, you'll learn how to use two text editors that are commonly available in \*nix (Unix/Linux) operating systema, as well as Git Bash and the bash terminal in macOS.
* Knowing how to use one or both of these text editors will help you in situations when you don't have access to a graphical text editor such as Visual Studio Code.

**Instructions**

**Why do we need to know this?**

* Visual Studio Code and other text/code editors with a graphical user interface are very handy, of course, but you're not always going to have access to such editors.
* There are times that you will find yourself logged into a machine that has only the bare minimum required software installed on it. Often, this will be a Linux machine, running a Linux distribution such as Ubuntu or Red Hat.
* In the final book of this week, as well as the Week 2 Workshop, you will be asked to log into a virtual server machine on the Amazon cloud, called an EC2 instance. (EC2 is short for Elastic Compute Cloud, Amazon's proprietary cloud composed of virtual machines that can be easily started up or removed according to need.)
* This machine will be running Ubuntu. You won't have Visual Studio Code on this machine! Or Notepad, or TextEdit, or anything like that.
* So what happens if you need to make an edit to a file?
* Two of the most common built-in \*nix (Unix/Linux) text editors are called **vi** and **nano**. (There's also **vim**, **emacs**, and many others, but we'll focus on these two for now.)
* You can typically access both of these from any bash shell, including Git Bash. You will also be able to access both of them from your Amazon EC2 instance, once we reach that point.
* Which one should you use? It's up to you.
* For now, let's learn how to open a file in each of these editors, how to make an edit, and how to save or discard the changes and quit the editor.
* Go ahead and open a bash terminal (through Git Bash, Terminal, or the VS Code integrated terminal).
* Let's use your user's home folder for this exercise, which you can easily get to by entering:

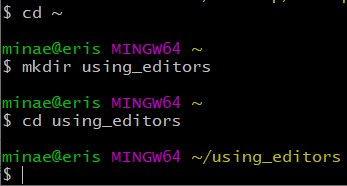
cd ~

* From there, let's make a temporary folder for this exercise:

mkdir using\_editors

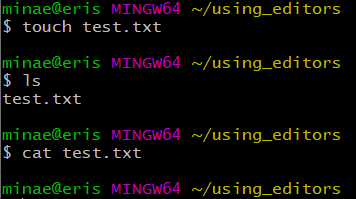
* Then **cd** into this folder:

cd using\_editors



* Let's create a file to play around with. We will do this with the **touch** command.
* If used with a filename that doesn't exist in the current folder, this command will create a blank file with the provided filename.
* Go ahead and enter:

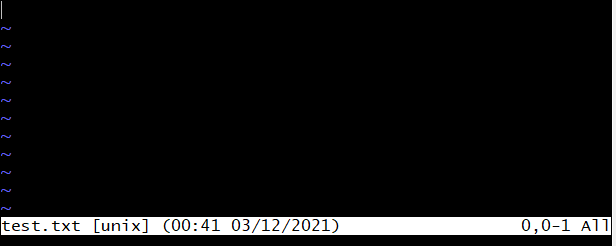
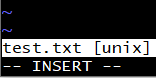
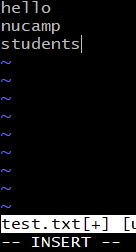
touch test.txt

* You can verify that the file was created by entering the **ls** command.
* You can also take a look at its contents using the **cat** command, which will show nothing because the file is empty:  
  
* Now let's see how we can add and edit content in this file using the **vi** text editor.

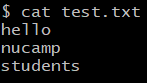
**Using vi**

* To open a file using **vi**, use **vi** followed by the full filename. If the file exists in a different folder than our current location in the terminal, then the path must be provided as well as the filename.
* In this case, the file is in the same folder where we are entering the command, so we only need to provide the filename.
* Enter:

vi test.txt

* The vi text editor will open the file, and you will see a screen like this, with the filename at the bottom:  
  
* There is nothing in the file currently. The ~ lines that you see are actually signifying a lack of any text.
* Now, you can't just start typing stuff right away.
* The vi editor has two modes: **command**mode, and **insert**mode.
* When you first open a file with vi, it opens in command mode.
* If you want to enter text into the file, you have to enter insert mode.
* To enter insert mode, press your **i**key.
* At the bottom of your terminal, you should then see:  
  
* If you don't see **-- INSERT --** at the bottom, then you are in command mode.
* Once in insert mode, you can enter text. Go ahead and type whatever you wish.
* Enter a few lines of text, and notice that you can navigate in the editor using arrow keys.  
  
* To save the changes and exit from the editor, you must first exit insert mode and return to command mode.
* To do this, press your **Esc**key.
* You should see the **-- INSERT --** text at the bottom of the editor disappear.
* Next, type this command to **write** (save) and **quit** from the editor, making sure that the first character you type is a colon, then press Enter:

:wq

* This should return you to the bash command prompt.
* Try the **cat** command again on the file, and you should see the text you entered:  
  
* Let's open the file one more time using vi:

vi test.txt

* Go ahead and make an update to the file contents. Remember, you have to type **i**to enter insert mode first, before you can make any updates.
* Now, let's say you've changed your mind and you wish to discard the updates you have made.
* To do this, press your **Esc** key, then type the command:

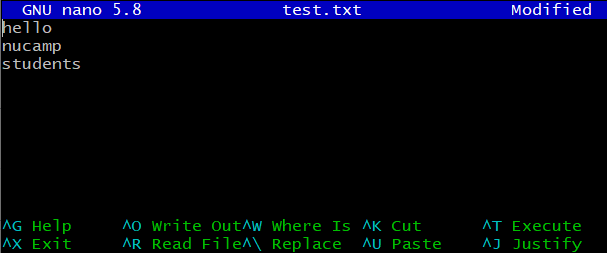
:q!

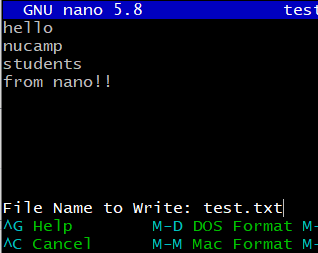
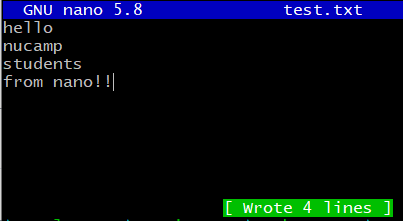
* And press Enter. This will quit from the vi editor without writing (saving) your changes.
* If you **cat test.txt** one more time, you should be able to see that the changes you made were *not*saved.

**Using nano**

* The GNU nano editor is another option that is very commonly available on any \*nix machine.
* Use it to open the test.txt file:

nano test.txt



* The vi editor is what you might call more "old school". It's been around since 1976!
* GNU nano is quite young in comparison, having been first released in 1999.
* While there are die-hard vi adherents, nano would be considered more user-friendly by most.
* As you can see, common commands in nano are shown at the bottom of the editor. The ^ symbol stands for Ctrl.
* You can start entering or editing text right away in nano, without needing to enter a special insert mode.
* Go ahead and add or update the existing text in the file.
* When you're ready to save your changes, enter the key combination: **Ctrl-o** (the letter o, not the number 0)
* You will see a prompt asking you for the File Name to Write, followed by the name of the current file:  
  
* If you wanted to save a copy of the file with your changes but leave the original file unchanged, this is where you could enter a different filename.
* For now, press enter to update the original file. You should see a message showing the total number of lines written to the file:  
  
* To exit from the nano editor, type **Ctrl-x**.
* As long as you have not made any changes since the last time you saved using Ctrl-o, this should return you immediately to the bash prompt.
* If you enter **cat test.txt** now, you should be able to see the changes you made.
* Once again, let's see how we can discard changes. Open the file using nano once again:

nano test.txt

* Make a small change to the file.
* Then enter Ctrl-x.
* You will see the following prompt:  
  
* Enter **n**. This will return you to the bash prompt without saving the change you made.
* You can confirm that the change you made was not saved, using the **cat test.txt** command.
* What you have learned here is enough to make simple updates to files when needed.
* You can explore the links in the additional resources to find out more about both editors.

**Delete the using\_editors/ folder (optional)**

* Feel free to delete the **using\_editors/** folder in your home folder when you're finished. We will not use it further in this course, though you can certainly keep it for further experimentation with editors.
* To delete the folder from the command line, use the following commands:
* Move into the parent folder of the **using\_editors/** folder with:

cd ..

* Then use the **rm**(remove) command on the **using\_editors/** folder along with the **-r** flag, which is required when deleting a folder (the r stands for *recursive*, meaning it will delete not only the folder but what is inside it):

rm -r using\_editors