FILE EDITING

MODULE OBJECTIVE

- Modify file content

LESSON OBJECTIVES

- Use the vi editor to modify file content

vi Editor (Visual Editor)

There are a number of choices available in Red Hat Linux for text editing programs. The **vi (Visual) editor** is the default text editor for many system tasks. Vi is a powerful, versatile editing tool. The biggest selling point however is that vi is always available no matter what flavor of UNIX/Linux and no matter if a desktop environment is running or not. There are however more robust editors, and editors that are easier to use. One such example is '**gedit**'. But because of reasons just noted vi is the editor of choice to do systems administration, - its just a matter of time before you find yourself in a situation when this is not available and your only option is vi.

Get used to the basics of the vi editor, starting a session, inserting and deleting text, cursor movement, and exiting a session. There are many commands available with the vi editor and many books and manuals explaining these commands, but until you understand the basics, the books may be confusing.

While in the vi editor, the user can operate in two modes:

Command (Control) mode	Used for cursor movement, text deletion, searching for a text string, environment setting, and exiting the vi session and saving the text to a file. When you enter the vi session you will be in the command mode.
Input (Text) mode	Used to insert text, various commands will put you in the input mode (i, I, a, A, o, O)

If you ever get confused about what mode you're in, press the **ESC** key on your keyboard. Pressing **ESC** always returns you to the command mode (and if you are already in the command mode, it simply beeps to remind you of that fact). Which mode you are in can be SOMEWHAT CONFUSING at first!

>>> This section will cover the commands that can be used to start a session, edit text while in the session, save the text, and exit the session. As you will see there are many commands to accomplish these tasks. Once again, learn the basics, and then build on them, and life with vi will be easier.

Starting vi editor

There are two ways to start the vi editor.

vi newfilename	Starts the editor and creates an empty buffer space to be edited.
vi oldfilename	Starts the editor and copies the file to a buffer space to be edited.

$ **vi testfile**

Display: The screen will clear, the first character on each line will be the tilde(~), and the cursor will be at the top-left corner of the screen. The newfilename will be displayed at the bottom of the screen.

```
_
~
~
~
~
~
~
~
"testfile" [New File]
```

You can insert text, using the text mode (insert is next in this discussion).

$ **vi file1**

Display: This will display the contents of the file1 and allow you to edit this file. Displayed at the bottom of the screen will be the filename, number of lines, and number of characters. With the cursor at the top-left of the screen.

This is a test file.
The name of the file is file1.
You will notice the number of lines in this file will not fill the screen.
Any lines that are not needed for text, will be the tilde(~), to the end of page.
~
~
~
~
"file1" 4 lines, 205 characters

Setting up vi

With the editor open you can use the **:set** command to specify options that change characteristics of your editing environment. Options may also be put in an **.exrc** file in your home directory to be available for all vi sessions without having to set them each time you open a new one.

There are two types of options that can be changed with the :set command: toggle options, which are either on or off, and options that take a numeric or string value (such as the location of a margin or the name of a file).

Toggle options may be on or off by default. To turn a toggle option on, the command is:

:set *option*

To turn a toggle option off, the command is:

:set *nooption*

Some options have a value assigned to them. For example, the window option sets the number of lines shown in the screen's "window". You set values for these options with an equal sign (=).

:set window=20

During a vi session, you can check which options vi is using. The command:

:set all

displays the complete list of options, including options that you have set and defaults that vi has “chosen.”

You can find out the current value of any individual option by name, using the command:

:set *option?*

shows options that you have specifically changed, or set, either in your **.exrc** file or during the current session.

Some options simply make editing operations easier.

:set number

displays a line number at the start of each line. This not only makes it easier to track where you are within a file, but enables you to use vi commands such as

:14,25 co 78

-or-

:14,25 mo 78

-or-

:49,60 w! vifile.tmp

-or-

:33,39 del

The first example **co**pies the text/data on lines 14-25 to line 78 (resulting in lines 78-89 being a duplicate of 14-25). The second example does a **mo**ve instead of a copy, so the lines will not be duplicated. The third examples saves a copy of the text/data on lines 49-60 to a file named vifile.tmp. And the last examples shows deleting the range of lines 33-39.

Character Motion in vi

Getting to a file isn't much good of you can't actually move around in it. You must know the control keys to get you where you want to go. Once you have the text/data displayed, you can navigate through the file with the following keys:

(While in the **control mode**)

h moves you LEFT

j moves you DOWN

k moves you UP

l moves you RIGHT (lower case L)

Ctrl-d moves the screen down a half page

Ctrl-u moves the screen up a half page

Ctrl-B moves the screen backward a full page

Ctrl-f moves the screen forward a full page
*n*G moves the cursor to the *n*th line of file

Home key moves the cursor to first line

<shift> G moves the cursor to the last line

Backspace key will move you one space to the LEFT.

Return key will move you to the Beginning of the next line.

If you try to move in a direction that is not possible (attempting to move up while at the top of the screen), the system will beep at you.

*** **Many systems allow you to use the arrow keys (in place of h,j,k,l)** – THANK YOU. Be aware however that if you attempt to use an arrow key on a system that does not allow that, it may put you into “text mode” (been there, done that !).

Inserting Text

Being able to move around in a file is useful. The real function of an editor, is to enable you to insert or delete information. The vi editor has a special text mode, that must be used to change the contents of a file. There are six different ways to enter the text mode, each having a different function.

i	inserts text before cursor position
I	(capital i) inserts text at the beginning of the line
a	appends text after cursor position
A	appends text at the end of the current line
o	opens a new line below cursor position
O	opens a new line above cursor position

Remember striking these keys will put you into the text mode. ESC will put you into the command mode, if you get confused and need a point of reference. While in vi, striking ESC until the system beeps, ensures you are in the command mode.

Deleting Text

Now that you know how to insert text into a file, you are naturally setting yourself up for the possibility to have to delete text, (possibly what you've just entered, but misspelled).

There are several ways to delete text, with vi. Each command will have a different function. Care should be taken when deleting text, if you are not familiar with vi. Think about what you want to do before doing it.

x	deletes the character at the cursor
dw	deletes from cursor to beginning of the next word
dd	deletes the line containing the cursor
*n*dd	deletes *n* (number of lines) from the cursor
*n*x	deletes n (number of characters) from the cursor
r	replace present character with next typed character
cw	change word with next typed word

Correcting UNwanted Deletions

While deleting text, as well as inserting text, you may find that you did something you didn't want to do. Vi is forgiving in this area, up to a very limited point. Because vi remembers the state of the file prior to the most recent action taken, mistakes can be undone.

u	undo the last command
.	repeat the last command
U	undo all changes on the current line. (Once you move off the line in question, the **U** command is unable to restore it.)

Search Commands

You would think, with all this newfound knowledge on the vi editor, you could go back and dazzle your friends. Well, we not only want to dazzle them, we want to amaze them. Learning to search a file for a certain letter pattern will leave them stunned and wondering, "Is there no end to this UNIX Wizardry?"

/pattern	search for the next occurrence of the pattern
?pattern	search for the preceding occurrence of the pattern

:% s/old string/new string/g global search and replace, similar to sed

n	repeat the last search command given

If you plan to use the vi editor to edit a text file, the search mode will make this process much easier.

Search pattern /pattern

```
$  vi vieditlab

This is a file used in the vi edit lab.
It will not take but a few lines to accomplish this task.
This line will be typed until it hits the right edge of the display.  Once it reaches the
end of the display it will wrap around.


This is the end of the file.
~
~
~
~
~
/it
vieditlab   7 Lines, 251 Characters
```

This search command will find the next occurance of the letter set "it", after the cursor and work it's way to the end of the file.

Search pattern ?pattern

```
$  vi vieditlab

This is a file used in the vi edit lab.
It will not take but a few lines to accomplish this task.
This line will be typed until it hits the right edge of the display.  Once it reaches the
end of the display it will wrap around.

This is the end of the file.
~
~
~
~
~
?is
vieditlab  7 Lines, 251 Characters
```

This search command will find the previous occurance of the letter set "is", prior the cursor and work it's way to the start of the file.

search pattern: :%s/old string/new string/g

This global search and replace is very useful for making repetitive changes. It is very similar to sed and follows the same syntax.

```
    $   vi vieditlab
This is a file used in the vi edit lab.
It will not take but a few lines to accomplish this task.
This is the end of the file.

:%s/vi edit lab/unixintro vi edit lab/g
This is a file used in the unixintro vi edit lab.
It will not take but a few lines to accomplish this task.
This is the end of the file.
```

NOTE: vi search pattern commands are case sensitive, only if the **:set noignore** command has been set.

Saving Text and Quitting the Editor

As discussed earlier, the editing is taking place in a temporary file. To save your work you must write it to a permanent file.

These are sometimes known as Colon Commands. Typing a colon (:) will move the cursor to the bottom of the text. This will allow you to save the text or exit in several ways.

The following are commands to save your work, as well as, exit the vi editor.

ESC :w	write the current text into the permanent file
ESC :w *filename*	write the current text into a different file *filename*
ESC :q	quit if no changes since last :w
ESC :q!	emphatic form of quit; no changes are written
ESC :wq	write and quit

ADVANCED OPERATIONS…

1) You can use wildcard characters to edit multiple files. The vi editor will stack the filenames that are expanded with the wildcard. To edit the next filename in the stack, use “q” or “w” to exit the current file, then “n” to advance to the next file. To quit all files in the stack, use “q!’.

2) To allow a pattern search to “wrap around” from the bottom of a file back through the top of the file, toggle the “wrapscan” setting from “nowrapscan” to “wrapscan”.

3) Several commands, including vi, allow you to execute a UNIX command with the ! character. Within vi, type “ :! ls “ to escape vi and list files in this directory.

LAB

Your practical exercises for this module:

It is time again to log onto the NWSTC student server (204.227.127.133) **and practice Linux commands.** If you need the instructions again they can be found at the following link:

http://webdev.nwstc.noaa.gov/d.train/linuxinstr.html

A couple of reminders

1. You are encouraged to **EXPERIMENT** in this course and try various commands, so that you **SUCCEED** in the field and subsequent training.

2. DO NOT enter the commands robotically without trying to understand them in the process. Your success at further Linux training and actual work in the field is wholly dependant upon grasping the subject matter in this course.

EXERCISE 1

Starting a vi session and creating a new file

1) Create a file with vi

```
$ vi  testfile1-vi
_
~
~
~
~
~
"testfile1-vi" [New File]
```

When starting a vi session, if the file does not exist, one will be created. You will then enter the vi session in the command/control mode, with the cursor blinking under the first character in the file. Tildes "~" will be displayed down the left side of the screen for unused lines. The bottom of the vi window will show (in this case) "testfile1-vi " [New File].

The previous command created an empty file in the vi session. Before you enter text, we will look at available options and set an option for the current vi session.

Setting vi session environmental options

2) List the vi environmental options.

While in the command mode, (if you are not sure that you are in the command mode, press the ESC key a couple of times), **enter a colon (Shift :)**. This will produce a colon at the bottom of the window.

```
_
~
~
~
```

:set all

*The ":set all" command will display a list of options that have been set and available to be changed. Take note of the "**noshowmode**" option, stating that you will not show the current mode (command/input). We are going to change this. Also note the "**timeout**" option, we will also change this option, but since we are not saving these settings, this option does not matter, it's just for instructional purposes.*

3) Changing an option with the :set command

```
_
~
~
~
```

:set showmode

*Using the ":set showmode" command will change this option from "noshowmode" to "showmode". Use the "**:set all**" command to check this option.*

4) Check to see if option setting worked.

To see if this option now works, **Press the " i " key** to put you in the insert/input mode (additional insert keys are covered in later labs). At the bottom right of the screen, you should see "INPUT MODE", meaning that you are now in the input mode. **Pressing the ESC key** will put you back into the command mode and the "INPUT MODE" will go away.

```
_
~
~
~
"testfile1-vi" [New File]                    INPUT MODE
```

Notice that when in the command mode, it does not show "COMMAND MODE". Hey, I didn't write the program.

5) Setting the option back to original setting

This option setting is for this current vi session only, (if you don't believe this... open another dtterm and start another vi session and see if you get the "INPUT MODE". That is not part of the lab, just proving a point.

If we exit from the vi session, (or close the window), it would be forgotten. But, for this lab, we want to change it from "showmode" to "noshowmode".

```
_
~
~
~
:set noshowmode
```

The ":set noshowmode" command will reset this option. Use the ":set all" command to once again check that the option has been changed.

Although, not part of the lab, the **/home/.exrc** file should be set up to allow the student to create their own **/home/studentx/.exrc** file. Setting up this file will set these options for any vi session that you start (except a session started on a remote machine). **DO THIS ON YOUR OWN TIME, or when all labs have been completed**.

6) Exit this vi session without saving the file.

The " **:q!** " will quit the vi session without writing the file to disk. The " **!** " won't ask for a write before it quits.

EXERCISE 2

Starting a vi session

1) Create a file with vi – in your "front_yard" (directory) create a dog(file) – I mean that is where a dog(file) belongs afterall .

$ **vi dog-file**

```
_
~
~
~
~
~
"dog-file" [New File]
```

When starting a vi session, if the file does not exist, one will be created. You will then enter the vi session in the command/control mode, with the cursor blinking under the first character in the file. Tildes "~" will be displayed down the left side of the screen for unused lines. The bottom of the vi window will show (in this case) "dog-file " [New File].

Adding and Deleting Text and Cursor Movement

2) Adding Text

Once you started the vi session, **Press the " i " key** and insert the following text:

```
This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
This line will be used for the deleting text exercise_
~
"dog-file" [New File]
```

The cursor should be blinking after the last character of your text. This file will be used in later labs, so if you use a different file or different text, just be aware of these differences, in those later labs. As we progress in this lab, we will use other text inserting characters, such as, (A, I, a, O, o), but the main body of text should remain.

3) Cursor Movement

Use the cursor movement keys (h, j, k, l, Shift-G), move around the file until you understand the usage. Did you remember to Press the " ESC " key to go back to command mode? Try the arrow keys and see if they work.

Some keyboards may or may not be mapped for the arrow keys to work. In the case of it not being mapped, you may be put in the INPUT MODE after attempting the arrow keys.

Now that you have mastered cursor movement, let us revisit adding text with the additional characters.

- Place your cursor at the beginning of the last line.

This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
This line will be used for the deleting text exercise
~
"dog-file" [New File]

- **Press the " a " key**, and insert some text, (you might use your name).

This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
T**Scobby-Doo**his line will be used for the deleting text exercise
~
"dog-file" [New File]

Notice the cursor moved one place to the right, went into the insert mode, and entered the text you chose, at the new cursor position.

- **Press the "ESC" key** (command mode), **Then "Shift-A"**, and enter text.

This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
TScooby-Doohis line will be used for the deleting text exercise**Scooby-Snack_**
~
"Dog-file" [New File]

With the "A ", you will append text at the end of the line currently holding the cursor. Lower case "a "appends to cursor location and "A "at the end of the line. The cursor is now after the last character of the line.

- The lower case " i " inserts text at the present cursor location and the " I " capital will insert text at the beginning of the line currently holding the cursor. Try these characters until you are comfortable with the results.

- **Place the cursor** anywhere on the next to the last line of text, **Press the " o " key** to open a new line below the cursor line. Enter some text and Press ESC.

This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
This is the text I entered
TScooby-Doohis line will be used for the deleting text exerciseScooby-Snack
~
"dog-file" [New File]

- **Press the " O " key** and open a line above the cursor line.

This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.

_
This is the text I entered
TScooby-Doohis line will be used for the deleting text exerciseScooby-Snack
~
"dog-file" [New File]

Notice that using the "o and O "will open the new line and place you in the input mode.

As you will see in the vi handout we gave you, or from any number of vi references you might use, there are many options to use with the vi editor. Find the commands you are comfortable with and become proficient with those.

4) Deleting Text

Deleting text can also be accomplish in many ways. We are going to keep this basic and use the " dd " and " x " commands for the lab. You are welcome to try the other commands in this reference material or any vi reference you may have. You must be in the command mode to delete text, <u>You will no longer be told to Press the ESC key</u>.

- Use the " dd " to delete a complete line. If your cursor is not in the blank line, created by the " O " in the last steps, then put it there. **Press the lower case d key twice " dd "**. This will delete the current line.

This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
This is the text I entered
TScooby-Doohis line will be used for the deleting text exerciseScooby-Snack
~
"dog-file" [New File]

- **Place the cursor** at the beginning of the line of text you added and **Press a single " x "** to delete a single character. Do this for 5 or 6 characters. **Press the " u "** lower case, to undo the last change. **Press the capital " U "** to undo all the changes to the current line.

- **Delete any extra text** in your file (with the method of your choice), leaving the basic text as follows:

This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
~
~
"dog-file" [New File]

5) Saving a file and Exiting the vi session

As was explained in lecture, when in a vi session, you are really editing a buffer file. Saving this text to a real file can be accomplished with the " :w " write command and Exiting can be accomplished with the " :q " command. This would be a two step process.

In this case we will accomplish this with one command:

Enter the " :wq " to write and quit in the same command.

This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
~
~
"dog-file" 4 Lines, ###Characters
$

Notice that the line at the bottom of the window has changed, stating the number of lines and the number of characters. This data has now been written to a real disk file named "dog-file".

EXERCISE 3

Searching for a text string in an existing file

1) Start vi with an existing file. While in the command mode, text strings can be searched for by using the " / " forward from current cursor location or " ? " backwards from the current cursor location.

- **Press the " / " key** and notice that at the bottom of the window will be a " / ". **Enter the text string** that you are searching for, and **press ENTER**.

$ **vi dog-file**
This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
~
~
/dog

This is a file named **d**og-file.
Many dogs stay in dog houses.
Some dogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
~
~
/dog

The "/dog "command will search for the first occurrence of the string "dog" after the current cursor location. If you strike a ***lower case "n "****, you will repeat that search and the cursor would continue to find the next text string "dog ".* **Strike the " n "** a few times.

- **Press the "? " key** and notice at the bottom of the window will be a " ? ". **Enter the text string** that you are searching for, and **press ENTER**.

This is a file named dog-file.
Many dogs stay in dog houses.
Some dogs hunt, some **d**ogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
~
~
?dog

The "?dog "command will search for the first occurrence of the string "dog"prior to the current cursor location. If you strike a ***lower case "n "****, you will repeat that search and the cursor would continue to find the next text string "dog ".* ***Strike the "n "****a few times.*

This is a file named dog-file.
Many dogs stay in dog houses.
Some **d**ogs hunt, some dogs are for show, and some dogs are just pets.
If you buy a dog for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in. ~
~
?dog

Search and replace old text string with new text string

2) Searching and replacing text can be done in a couple of ways. Using the " :%s/old/new " substitute command, will replace the first occurrence of the string in each line. Using the " :%s/old/new/g " with the global option, will replace all occurrences of the text string throughout the text file.

- Replace the string " dog " with " cat " using the substitute " **:%s/dog/cat/** " command

$ **vi dog-file**
This is a file named **dog**-file.
Many **dog**s stay in **dog** houses.
Some **dog**s hunt, some **dog**s are for show, and some **dog**s are just pets.
If you buy a **dog** for a pet, you should research the breed of **dog** to find
what type of family atmosphere the **dog** is comfortable in.
~
:%s/dog/cat/

This is a file named **cat**-file.
Many **cat**s stay in dog houses.
Some **cat**s hunt, some dogs are for show, and some dogs are just pets.
If you buy a **cat** for a pet, you should research the breed of dog to find
what type of family atmosphere the dog is comfortable in.
~

Notice that in any line with multiple occurrences of the text string "dog ", only the first was replaced.

- Replace the string " dog " with " cat " using the global " **:%s/dog/cat/g** " command

$ **vi dog-file**
This is a file named **dog**-file.
Many **dog**s stay in **dog** houses.
Some **dog**s hunt, some **dog**s are for show, and some **dog**s are just pets.
If you buy a **dog** for a pet, you should research the breed of **dog** to find
what type of family atmosphere the **dog** is comfortable in.
~
:%s/dog/cat/g
This is a file named **cat**-file.
Many **cat**s stay in **cat** houses.
Some **cat**s hunt, some **cat**s are for show, and some **cat**s are just pets.
If you buy a **cat** for a pet, you should research the breed of **cat** to find
what type of family atmosphere the **cat** is comfortable in.
~
"dog-file" 4 lines, ###characters

Notice that all occurrences of the string "dog "were replaced with "cat ".

Saving file to a different name

Saving a file to a different name may be done for many reasons, one of which would be to save the changes you just made and while retaining a copy of the original file. This can be accomplished with the " :w new-filename " command.

- Write the file to a different filename, using the " :w new-filename " command

$ **vi dog-file**
This is a file named dog-file.
.....
what type of family atmosphere the dog is comfortable in.
~
:%s/dog/cat/g
This is a file named cat-file.
.....
what type of family atmosphere the cat is comfortable in.
~
:w cat-file

:q Exit the vi session

Note: If you will be attending the NWSTC course LINUX for WFOs/RFCs, you will be editing a lot of configuration files in lab; and thus it may be beneficial to take additional time here to practice more vi with your dog-file or other files.

End

This is the end of this module. At this time you should proceed to the final evaluation exercises (6-final-eval.pdf).