CIS 18A
Introduction to Linux / Unix

Text Editing

De Anza College
Instructor: Clare Nguyen

Part 1

Topics:
- Text editing tools and vim
- Start and end a vim session
- Move the cursor
- Add, delete, and replace text
- Search

Text Editors

- A text editor is a tool used for creating and modifying text files.
- A text file contains basic characters that you can type in from the keyboard: letters, numbers, punctuation marks. A text file does not contain text formatting such as different font sizes, font types, colors, or special formats such as columns, bullets, and tables.
- Text files are most often used with computers: software programs, scripts, log files, error files. On the Windows system, text files have the filename extension of *.txt*
- Because text files are basic character files that are used with computer systems, a text editor has basic functionalities to modify these files, such as add, delete, change, cut, copy, paste. To do more visual formatting such as font color or table formatting, you would need to use a word processor rather than a text editor.
- For this class, we will learn to use a text editor so that we can modify text files.

Text Editing Utilities

- There are a few common text editing utilities available for Linux:
  - vi: (**v**isual **i**nterpreter) the "original" and oldest text editor. It was developed by UC Berkeley student Bill Joy, when Unix was in its early stages. It is now often replaced by vim.
  - vim: (**vi** **im**proved) built on top of vi, with more features. It was created by Bram Mooleanaar for the Amiga computer. This is the default text editor for Linux.
  - pico: (**pi**ne **co**mposition) an editor built for the pine email tool. It was developed at the University of Washington and is most often used with the pine email tool.
  - emacs: (**e**ditor **mac**ro**s**) this editor is as popular as vim. It was originally created by Richard Stallman (of Free Software Foundation fame).
- For this class, we will learn to use vim, sometime pronounced as if it rhymes with 'him'. vim is one of the most useful editor to learn for Linux, because every Linux distributions comes with vi as part of its package. Other editors are not guaranteed to be available without an extra download and install.

Using vim / vi (1 of 3)

- For voyager, the command name for vim is shortened to vi (to make it shorter for the user to type), so the command name vi will be used throughout the lecture notes.
- Because voyager has a command line interface, there are a few things to keep in mind when working with vi:
  - Don't run vi when the utility script is running. Every time you make a change to the text, vi will refresh the screen by sending to the screen many characters. This will make your script output file grow out of control.
  - Don't use the scroll bar on the vi window to scroll up and down. The scroll bar is part of your local GUI system and will not interact with vi, so the scroll bars will not have the effect you want and have been known to confuse students. Use the vi scroll commands instead.

Using vim / vi (2 of 3)

- Since vi is a command line tool, there is no pull down menu for copy and paste, and no mouse to highlight and select. vi has to work with just the basic keys of letters, numbers, punctuation marks.
- If every usable key is a certain text character, how can you tell vi that you want to select and delete a line?
  The answer is in the 2 modes in vi: the *command mode* and the *insert mode.*
- *Command mode*: every key you type is interpreted as a command. Example: r for replace, q for quit.
- *Insert mode*: every key you type is interpreted as a text character that you add to the text file: r means r will appear in the text file, q means q will appear in the text file.
- It is very important that you keep track of which mode you are in, since r can mean 2 different things.

Using vim / vi (3 of 3)

- vi always starts in command mode.
- At any time, to go to command mode, use the escape key.
- From command mode, there are different commands to go to insert mode, as covered in the next slides.
- vi commands are only used when you are in vi. They will not be interpreted correctly by the shell, so don't use them at the shell prompt.
- Disclaimer: just like with other Linux utilities coverage, the following lecture notes on vi is not a complete coverage of all vi commands. They are a set of most commonly used commands to help you get started, and they are commands you're responsible to learn in this class. For more sophisticate use of vi, refer to the man page or the text book.

Command to Get Started

- To run: vi filename
  - If filename is a new name, the new file will be created and then opened.
  - If filename is an existing name, the existing file will be opened.
- Empty lines (lines with nothing on it) appear with a ~ at the beginning of the line.
  - If the file is new, all lines will have ~
  - If the file is smaller than the size of the vi window, the last lines will have ~
  - If the file is larger than the size of the vi window, you see the first part of the file.
- The cursor will typically be at the beginning of the file, or on the line where you last used vi with the file.
  - To open an existing file with the cursor at a particular line:
    vi +*n* *filename* where *n* is the line number you specify.

Commands to Get Out of vi

- To get out of vi, you need to be in command mode so you can use the quit commands.
- To quit when you haven't changed the file :q
  this is useful when you open the file for reading only
- To quit but don't save any change :q!
- To quit and save any change :wq
- The colon : is required in the quit commands. The q is short for **q**uit, the w is short for **w**rite.
- If you've changed the file and use :q, vi will ask you to retype the command with the exclamation point :q!

Save Commands

- You need to be in command mode to issue these save commands. You stay in command mode after these commands are run.
- To save a file when quitting :wq
- To save a file without quitting :w
  It's recommended that you do this every 5 or 10 minutes when you've been modifying a file. (A good rule of thumb no matter what editor you're using).
- To save a snap shot of the current file to a new file
  :w new_filename
  Note that this does not save the current file in its own filename.
- The colon : is required. The w is for **w**rite

Commands to Move the Cursor (1 of 2)

- You need to be in command mode to issues these move commands. You will stay in command mode after the move.
- The following commands are grouped in order of the distance of a move: from smallest distance to largest distance.
- Move by one character:
  - Left: left arrow or h
  - Right: right arrow or l (lowercase L)
  - Up: up arrow or k
  - Down: down arrow or j

  The j,k,h,l keys are not intuitive for the directions. They were created back at the time when a keyboard had no arrow keys, and these letters are the home keys (and therefore most easily reached) for a 10-finger typist.
- Move by one word (a word is one or more character, surrounded by space):
  - Left: b (for **b**ack)
  - Right: w (for **w**ord)

Commands to Move the Cursor (2 of 2)

- Move *within* a line:
  - To the end: $ ($ often means end or last in Linux)
  - To the front: 0 (number 0)
- Move one screen:
  - To next screen: control-f (f for **f**orward)
  - To previous screen: control-b (b for **b**ackward)
- Move to a line number: *n*G where *n* is the line number
  Numbering starts at 1, there is no space between *n* and G.
  G is for **g**o.
- Move to last line: G
- Move to first line: 1G or gg

Commands to Add Text (1 of 2)

- You need to be in command mode to run the add commands.
- The add commands put you in insert mode. At the bottom of the vi window is the word Insert to remind you that you're in this mode.
- Once you're in insert mode, any character you type is text that is added to the text file.
- When done, you need to use the escape key to get back to the command mode.
- Terminology: the cursor shows where your current position is in the text.
  - The character at the cursor is the current character.
  - The word that contains the cursor is the current word.
  - The line that contains the cursor is the current line.

Commands to Add Text (2 of 2)

These commands add text. Use the one that is shortest to add text at the location you want.

- Add text after the cursor: a (for **a**ppend)
- Add text before the cursor: i (for **i**nsert)
- Add text at the end of the line: A
- Add text before the *first word* in the line: I (uppercase i)
  Note that this command does not add text at the very beginning of the line, if the beginning has space characters. This is useful for programmers who need to keep their code indented.
- Add a new line of text above current line: O (uppercase o)
- Add a new line of text below current line: o (lowercase o)
  (o for **o**pen)

Commands to Delete Text

- You need to be in command mode to run these delete commands. After the commands run, you're still in command mode.
- These commands delete text. Use the one that is shortest to delete text you want.
  - Delete current character: x
  - Delete current word starting from the cursor: dw (for **d**elete **w**ord)
    To delete the entire word, make sure the cursor is at the beginning of the word.
  - Delete current line, starting from the cursor: D (for **d**elete)
  - Remove entire current line, no matter where the cursor is: dd

Commands to Replace Text

- Replace means delete existing text and then add new text in its place.
- The replace command does both steps, saving you from having to delete in one step, then add text in a second step.
- You need to be in command mode to run the replace commands.
- Replace one character: r then new_character
  The new_character can only be one character, then you're automatically back to the command mode.
- Replace one word: cw (for **c**hange **w**ord)
  This command deletes the current word, starting from the cursor, then puts you in insert mode so you can add the replacement text.
- Replace one line: cc
  This command deletes the entire current line, no matter where the cursor is, then puts you in insert mode so you can add the replacement text.

Commands to Search Text

- You need to be in command mode to run the search commands. You stay in command mode after search runs.
- The search commands search the file for an exact match of the text string. The text string can be a single character or multiple words. Spaces count as characters in the text string.
- The search command will show you the first match.
- To search forward (from the cursor toward the end of file) for text_string: /text_string
- To search backward (from the cursor toward the begin of file) for text_string: ?text_string
- Since search will stop at the first match, you can ask for the next match:
  - In the same direction: n (for **n**ext)
  - In the opposite direction: N
- If search reaches the end of the file, it will wrap around and continue the search starting at the beginning of the file. The same works for a backward search.

Part 2

Topics:

- Substitute text
- Copy and paste, cut and paste
- Move existing text
- Repeat, undo, and redo
- Administrative commands

Commands to Substitute Text (1 of 2)

- The command to search *and* replace text is called the substitute command. It will search for a text string, and when it finds a match, will remove the text string and add the replacement text string.
- You need to be in command mode to run the substitute command, and you stay in command mode after substitute runs.
- Format: :addresss/search_text/replacement_text/g
  where address and /g are optional, but the colon : and everything else is required
- Example :s/linux/LINUX
  - substitute will find the first match of linux on the current line and replace it with LINUX.
  - If there is no match, then nothing will change on the current line.
  - If there are multiple linux on the current line, only the first one will get substituted.
- To substitute multiple instances of a text string on the same line, you need to use the option /g (for **g**lobal).
  - Example :s/linux/LINUX/g will substitute all instances of linux on the current line with LINUX

Commands to Substitute Text (2 of 2)

- If there is no address field, the substitute command will work on the current line only.
- The address field specifies which line(s) substitute will work with:

Address	Example	Explanation
Range of lines	:2,8s/linux/LINUX	substitute from lines 2 to 8, inclusive
Last line	:$s/linux/LINUX	substitute on last line
Current line	:s/linux/LINUX	substitute on current
Entire file	:%s/linux/LINUX	substitute on entire file
Range calculation	:.,.+2s/linux/LINUX :.-10,.+8s/linux/LINUX	substitute from current line (note the use of . for current line) to 2 lines after current line substitute from 10 lines before current line to 8 lines after current line

Commands to Copy and Paste

- Copy and paste takes 3 steps, just like with any other editor: copy the text, move the cursor to the appropriate location, paste the text at the cursor.
- You need to be in command mode to run the copy and paste commands, and you stay in command mode after they run.
- To copy text:
  - Copy current word: yw (for **y**ank **w**ord)
  - Copy current line: yy
- The copy command copies the text into a temporary buffer. If there is already text in the buffer, it will be overwritten.
- To move the cursor, use any of the move commands.
- To paste text:
  - Before the cursor: P (for **p**ut, uppercase)
  - After the cursor: p (lowercase)
- The text copied into the buffer can be pasted over and over again, as long as the buffer is not overwritten.

Commands to Cut and Paste

- Cut and paste takes 3 steps, just like with any other editor: cut the text, move the cursor to the appropriate location, paste the text at the cursor.
- You need to be in command mode to run the cut and paste commands, and you stay in command mode after they run.
- To cut text:
  - Cut current word: dw (same command to delete word)
  - Cut current line: dd
- The delete command copies the text into a temporary buffer. If there is already text in the buffer, it will be overwritten.
- To move the cursor, use any of the move commands.
- To paste text:
  - Before the cursor: P (for **p**ut, uppercase)
  - After the cursor: p (lowercase)
- The text in the buffer can be pasted over and over again, as long as the temporary buffer is not overwritten.

Commands to Move Existing Text

- You need to be in command mode for these commands and you stay in command mode after they run.
- Command to **j**oin 2 lines of text: J (uppercase)
  J will take the line below the current line and append it to the current line.
- Command to insert text from another file :r filename
  - Note the : in front, r is for **r**ead.
  - This command will copy the file filename, and insert the copied text after the current line.

Commands to Repeat An Action

- You need to be in command mode to run these commands and you stay in command mode after they run.
- To repeat the previous command: . (a period or dot)
  This command will not repeat a move, search, or substitute command
- To have vi run a command more than once: nCommand
  where n is the number of time you want vi to run Command

Example:	dd	delete the current line
	8dd	run delete line command 8 times, which results in deleting 8 lines, starting from the current line
	j	move cursor down 1 line
	20j	move cursor down 20 lines

Commands to Undo and Redo

- You need to be in command mode to run these commands and you stay in command mode after they run.
- To undo: u

  You can keep undoing until the last time you save the file
- To redo: control-r

Administrative Commands

- To see the status of the current file, such as the filename, how many lines of text, etc. :f (for **f**ile)
- To temporarily bring up a **sh**ell :sh
  - At the shell prompt you can run any shell command.
  - When done, type exit at the shell prompt and to get back to the vi screen that you were at.
- Line numbering:
  - To show line numbers for the lines in the text file :set number
  - To show no line number :set nonumber
  - Note that the line number shown is for the display window of vi only. No line number is added to the text file itself.
  - The default is no line number. Setting or not setting the line number applies only to the current vi session.
  - To save the line numbering permanently, use vi to create a text file named .exrc. Add in this file the line set number

    and then save the file. Line numbering will take place in the next vi session.

Swap File

- When you use vi to open any text file, for example the file fileA, vi creates a temporary file with the name .fileA.swp. This called a swap file.
- It is this swap file that you change if you make any change to fileA.
- When you quit out of vi, vi copies .fileA.swp and overwrites the original fileA, and then deletes .fileA.swp
- If you did not quit out of vi by using a quit command, then vi does not know to overwrite the original file and delete the swap file, so the swap file remains in your directory.
- This means that when you use vi to open fileA the next time, vi will alert you that the swap file exists. At this point, you have a choice to use the swap file or to use the original version. If you do not want to be alerted again about the swap file, make sure you delete it.

Next stop: Directories