CST2555
Operating Systems & Computer Networks
Week 4

1 Overview

Unless stated all exercises are to take place in the terminal, the whole point of these exercises is to get use to CLI, not GUI.

vim is a Unix text editor that's included in Linux. It's known for being fast and efficient, because it's a small application that can run in a terminal (although it also has a graphical interface), but mostly because it can be controlled entirely with the keyboard with no need for menus or a mouse. We can create and edit text files using vim. In Bash programming we will use vim to create a file and write a program.

2 Install vim

For Linux systems **vim** will need installing. By default, Linux systems have **vi** installed, and some have **vim**. If you don't have vim installed type the following:

- Update your Linux using: sudo apt-get update
- Install vim: sudo apt-get install vim
- Type: vim
- Return to CLI using: :q

3 How to use vim

- Start Vim from a terminal by typing vim.
- Press **i** to enter insert text mode. When in insert mode, all you can do is type text into your document. There are no commands in insert mode.
- Press **Esc** to enter normal mode, used for commands.
- In normal mode, you can move your cursor with h (left), j (down), k (up), and l (right). It might help to remember that j is down by equating it, visually, with a Down arrow.
- To exit vim, type :wq if you want to save your work or :q! to discard unsaved changes.
- If there is no name for the file, then use this command in normal mode:
  - :wq filename.txt

4 Create a file with vim

- Type this command in Terminal: vim test1.dat
- Insert two lines of data and save it as mentioned above.
- Return to CLI and check if the file exists.
- See the content of the file using cat command.

5 vim

Open a browser and go to Vim Tutorial and complete the tutorial. Using this, you will be familiar with basics of vim.

https://www.openvim.com/

6 history

Complete the following:

- Open a terminal
- Type history and push enter
- Back up the entry using the write switch -w
  - history -w history.txt
- Check if the file exists
- Check its content
- Clear the history using the clear switch -c
- Type history and push enter
- There should be nothing displayed (except history command)
- Restore the history log by using the read switch -r
- Display the last 10 commands in the history log
  - history | tail -n 10
- Type history and push enter. From the list execute a benign command via the numbered entry
  - history | !number
- Using the grep command, pipe the output of history into grep, then search for all the commands in the history log that include the word, “history”.
  - history | (can you remember form week 3?!)

7 echo (command to display)

Complete the following:

- Open terminal
- From the terminal display **Hello World** using the echo command
- From the terminal display Hello World with each word on separate lines using the echo command
  - echo -e 'Hello \nWorld'
- Create a file using the echo command combined with the redirect command
- Check if the file exists and then see its content

8 mv (Move)

Complete the following:

- Open a terminal
- Going back to Week 2, clone the 1000Files again (if you don't have it already).

```
git clone https://github.com/iangmitchell/thousandFiles
```

- untar the tarball (see week 2 notes for help)
- `cd 1000Files`
- `cp 0/1/* .`
- Check the copied files.
- Move `010.txt` to its original location and make a backup in 0/1
  - mv -b 010.txt 0/1

- Move 011.txt to its original location (without backup)
- Move `012.txt` to its original location and force an overwrite
  - mv -f 012.txt 0/1
- Check if these commands worked, using ll command
- Move `013.txt` to its original location and prompt for an overwrite
  - mv -i 013.txt 0/1
- Move *.txt remaining text files to their original location and only overwrite if the source file is newer than the destination file
  - mv -u *.txt 0/1
- use ll to check the files in 0/1

9 File Permissions

Complete the following:

- Open a terminal and go to the home directory.
- Create a file: `touch aFile.dat`
- Open the file using vim and type **ls** and save the file
- Create another file: `touch bFile.dat` and type ls via vim
- Inspect the file: ll
- What are the differences in these files compared with other files?
- Execute file: ./aFile.dat
- What is the problem?

- Change the file so it's executable for the user.
  - chmod u+x aFile.dat
- Execute the file
- What is the output? Why?
- Type ll
- Now, what is the difference between aFile.dat and bFile.dat
- Do the same for bFile.dat and execute it.

10 Basic commands in vim

Commands	**Description**
i	Insert mode
esc	Normal mode
v	Virtual mode
:q!	Close vim without saving
:wq	Close vim and save
/text	Search a text in the file
:set number	Show the lines in the file
y	Copy (first go to virtual mode and select the word/line)
p	Paste
0	Beginning of line
$	End of line
u	Undo
Ctrl+r	Redo

References

[1] Ellen Siever, Stephen Figgins, and Aaron Weber. *Linux in a Nutshell*. O'Reilly.