SENG265: Software Development Methods (Summer 2019)

Lab 01 – Linux and Command Line

Week of May 13th

Authors: Nirav Galani*

*Built on the works of Amanda Dash and David Johnson

Before we get started…

- Labs will be posted by the Sunday evening before the labs
- Attendance will be taken weekly
- You must submit you attendance through connex. This is the only way to submit attendance. You lab instructor will show you how to submit attendance.
- If you miss your regular lab, you may attend another lab. However, you are urged to attend the lab that you are registered for to avoid crowding of certain labs.
- If for a lab, the number of computers is less than the number of students who have showed up to attend, students registered for that lab will be given priority.
- Do not post your code on GitHub or any other public repositories during the semester
  - We will be using a UVic git repository for managing and submitting assignments during the semester.

This week

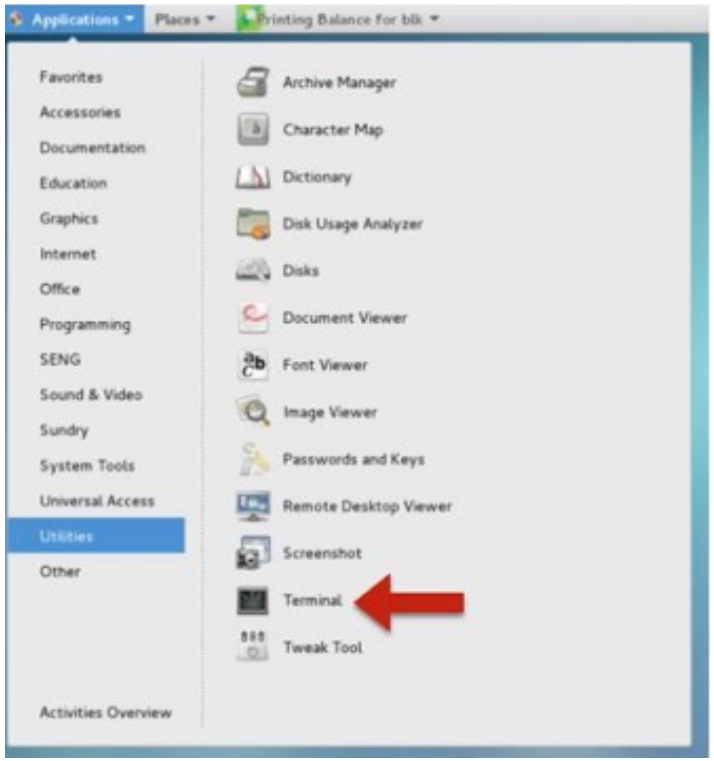
- Working with the command line, basic linux commands
  - cd, pwd
  - mkdir, rm, mv, cp, ls
  - chmod
- Text editor – *Vim*
  - *Opening and closing*
- File permissions
- Remote Connections -- scp, ssh

Before we begin..

- We use a common name convention when talking about commands:
  - <var> : This indicates that var is a variable that is required. You fill in var with your content.
    - grep <search_for> <search_here> → grep bob.txt ~/Desktop
  - [var] : This indicates that var is a variable that is not required.
    - ls [options] → ls –lah
- When you login to a machine (locally or remotely) you will always find yourself in you home directory: /home/<user>
  - At UVic, <user> will be your netlink id
  - To access your home directory you can also use ~ as a shortcut
    cd /home/<user> is the same as cd ~
- On most systems, your command prompt will end in an ($) so we will use that to indicate commands
- REMEMBER: Linux is **case-sensitive**

Accessing the Terminal

- On the lab computers:
  - Applications > Utilities > **Terminal**
- Will Probably be different on your computers
  - If you use Windows may need to install a terminal emulator
  - Although Windows 10 now has a Linux subsystem you can install
  - (These are beyond the scope of this course)



Basic Commands

- The mkdir (make directory) command makes a folder.
  - Usage:
    $ mkdir <new_folder_name>
- Create a folder on your Desktop called seng265 for this course
  - $ mkdir ~/Desktop/seng265
- The rm command deletes a file or folder.
  - Usage:
    $ rm <file_to_delete>
    $ rm -r <directory_to_delete>
  - The -r stands for recursive, which is required to delete folders.
  - Use rm with caution. **There's no undo!**

Basic Commands – File Management

- The mv (move) command moves a file from one place to another
- The cp (copy) copies a file instead.
- Usage:
  $ mv <source_file> <destination_path>
  $ cp <source_file> <destination_path>
- You can also rename a file with mv (but not cp) by giving a new filename, instead of a location.
  - (Be careful: You can accidently rename a file without realizing it if the destination path doesn't exist!)

The Search for Captain Ctrlaltdel's Treasure

- Many years ago, the nefarious pirate captain Ctrlaltdel decided to hide his treasure in the one place he figured no one would look: a place an old map calls 'Dev/Null Island'.
- This location, amazingly, has been found in UVic's computer system.
- It's up to you to find the treasure... with the power of the command line!
- Download the file 'devnullisland.zip' from Connex, and save it to your seng265 folder you created and extract the file there:
  - $ cd ~/Desktop/seng265 && unzip devnullisland.zip

Cap’n Ctrlaltdel’s Treasure

- Set sail for your desktop by typing:
  - $ cd ~/Desktop
- Then, use cd to go to the *devnullisland* folder. Once you’re there, you can use cd to navigate between the different parts of the island.
- If you get lost, type $ cd .. to retrace your steps.
- Once you find the treasure chest (*chest.txt*), open that chest with cat and reap your reward!

Task (Using terminal)

- **Using terminal**, create the following directory structure inside the seng265 folder created in the earlier slide. Note that all the items with word "Folder" in the name are directories. All the .txt items are files.
  - seng265
    - Lab01
      - Folder1
      - Folder2
      - Folder3
      - Folder4
      - Folder5
      - File1.txt
      - File2.txt
      - File3.txt
      - File4.txt
      - File5.txt
      - All.txt

Task contd.....

- Move File.txt into Folder1, File2.txt into Folder2……..and so on.
- Copy all.txt into all the folders
- Delete File3.txt
- Delete Folder5
- What are the permissions for File2.txt?
- Create a new file abc.txt. What are its permissions?
- Try changing the permissions of user, group and others to File.txt

VIM

- A command-line text editor
- Starting vim:
  - $ vim [filename]
  - $ vim <file1> -O <file2> # open two files side-by-side
  - Creates a blank filename if none are specified
- vim has two modes: Command Mode and Insert Mode.
  - **Command Mode:** you can issue commands, but not type text.
  - **Insert Mode:** you can type text, but not issue commands.

VIM – Command Mode Basics

- In command mode you type functional commands that do things to your document
  - :w - Save your file. The colon is part of the command.
  - :w <filename> - saves to a file of that name.
  - :q - Quit vim. To quit without saving, use :q! Instead.
  - :wq - save and quit.
  - :! <linux command> - Run a linux command from vim
  - i - Enter Insert Mode so you can type in your document
  - Esc – To leave insert mode and enter command mode

Task (Using Terminal)

- Open the file “read.txt” using vim
- Scroll up and down
- Notice that the file contains excerpts from Midnight’s children, but the text is duplicated several times
- Remove all duplicate instances
- Save your changes
- Quit vim
- Use this file to practice using vim

Vim – Other Useful Commands

- 0, $ - Move within a line (0 goes to the start, $ goes to the end)
- Ctrl-f, Ctrl-d - Move the cursor down (whole screen, half screen)
- Ctrl-b, Ctrl-u - Move the cursor up (whole screen, half screen)
- dd - delete a line
- u - undo your last action
- Shift+v [left/right cursor]- highlight text under the curse
- y - Copy (yank) highlighted text and p to paste yanked text
- :sp <file> - Open another file in horizontal split, :vs for vertical split
- :colorscheme <scheme> - Change the vim highlighting color scheme

Remote Logins

- The ssh (secure shell) is a command that lets you log into another computer.
- The scp (secure copy) lets you transfer files between computers
- Usage:
  - $ ssh <username>@<address_of_computer_to_log_into>
  - $ scp [-r] <username>@<address>:</path/to/file(source)> <dl_here(destination)>
  - $ scp [-r] </path/to/file (source)> <username>@<address>:<dl_here(destination)>
- Assignments must work on lab computers with the SENG265 environment enabled
  - $ setSENG265
  - Before submitting it is suggested that you ssh into a lab computer and test compiling and running your code
- Windows 10 now has SSH functionality in PowerShell
  - Https://www.howtogeek.com/336775/how-to-enable-and-use-windows-10s-built-in-ssh-commands/

Remote Logins

- Helpful to find out information about the computer you're ssh'd into
  - (but also works on local computer)
- Let's find out some info about this computer!
  - $ who # Users currently logged in
  - $ cat /etc/issue.net # What Linux distribution is this?
  - $ gcc --version # GCC version
  - $ python --version # default python version
- To leave, use exit or logout

Remote logins

- Remotely login to ugls.ece.uvic.ca using ssh
  - ssh <netlink username>@ugls.ece.uvic.ca
  - You will be asked for a password. Enter you netlink password
  - Explore the directory structure using the commands learned in this lab.

REFERENCE SLIDES

- The slides following this are for reference
- You will find the information in these slides and other information in the lecture slides

LINUX

- Linux is a free, open sources operating system

Command Line Interface (CLI)

- Any computer interface where the user enters textual commands and gets textual responses is a CLI
- We are using Terminal
- The CLI is just a tool to take input and output in a form that humans can understand. Another program is needed to process the input and provide output.

• Shell

- Shell takes the input from the terminal, interprets it as commands, runs the programs needed and sends the output back to the terminal.

Basic Commands

- When you're in the command line, you are always in a folder called the ***Working Directory***. It is like being in a folder in a graphical interface. You can view your current directory with the **p**rint **w**orking **d**irectory (pwd) command
  - Usage:
    $ pwd → /home/username (if in the home directory)
- To change your current directory use the cd (**c**hange **d**irectory)command
  - Usage:
    $ cd <destination/directory>
  - Example:
    $ cd Desktop/seng265/a1

cd tips

- The special name .. refers to the parent directory of the current directory.
  - cd.. changes the current directory to the parent directory
- The special name . refers to the current directory
- The special name ~ is your home directory
  - cd ~ changes the current directory to your home directory

Basic Commands

- Absolute Paths
  - A path that starts with / is an absolute path,
    e.g cd /home/davidjo/Desktop/seng265
  - The path starts from the top level of the computer's hard drive.
- Relative Paths
  - If the path does not start with / then it is relative
    e.g cd Desktop/seng265
  - The path starts from the current working directory (and path must exist in the working directory)

Basic Commands

- The ls command displays the contents of the current directory
  - Usage:
    - $ ls
- Other options for ls can also be used.
  - Try ls -l
  - ls -al gives more information in a long list and even shows hidden files. Try it.
  - In future labs, we will see how to filter and search through the results.

Basic Commands – File Management

- The cat (concatenate) command displays a file's text.
  - Usage:
    $ cat <file_to_display>
  - Try:
    $ cat <file1> <file2>

- The touch command creates a new blank file
  - Usage:
    $ touch <new_file_name>

File permissions

- Permissions can be set for
  - User “u” [rwx------]
  - Group “g” [---rwx---]
  - Other “o” [------rwx]
  - All “a”
- Try ls -l and you will see permissions for each
- There are 4 types of permissions
  - Read “r”
  - Write “w”
  - Execute “x”
  - None “-”

File permissions

- Create an new file test.txt
  - $ touch test.txt
- What are the permission for this file.
- Permissions can be changed using chmod
  - $ chmod <permission_criteria><file_or_foldername>
  - Permission_criteria is in the form of {user/group/other/all}{+/-(+ to add and – to remove)}{type_type_of_permission(r/w/x/-)}.
  - Example
    - $ chmod u+x test.txt → add execute permission to test.txt for user.
    - $ chmod o-r test.txt → remove read permission to test.txt for other.