CS151 Lab0 Unix, Emacs and Remote Access

All programming assignments and labs in this class must be completed on our Linux server. You can either come to our lab, or connect to our server remotely via an ssh client (see **Remote Connections**). We will not be using an IDE. Instead, you will learn to edit your code with an editor, compile and run your programs at the command-line, as well as organize your program files and folders yourself. Therefore, your primary interface is the Linux shell and it is important that you feel comfortable navigating the Linux system using the command line.

The Unix/Linux operating system consists of three parts: the kernel, the shell and the application programs. The kernel is the heart of the operating system, it allocates processor time and memory, handles file operations and user tasks. The shell acts as an interface between the user and the kernel. The shell is what you are typing to at the prompt after you log in. The following section outlines the basic commands for command-line navigation in the Linux environment. If you are familiar with this from 113 (read it to make sure you really are), you may skip the section.

Linux/Unix basics

- Login
- **`man` (RTFM)**
  - `man <entry>`
  - Displays the manual for a given command
  - Flags you should know
    - –k Search the man pages for the given word.
    - –a Displays all entries for the command instead of the first one (if there are more than one).
- **`ls`**
  - Displays all the files in the directory.
  - Flags:
    - –l displays additional information for each file
    - –a displays all files, files that begin with "." are not displayed automatically
    - –F displays "/" after directories, and "*" after executables
    - –t sort by timestamp instead of alphabetically
- **`mkdir`**
  - `mkdir cs151`
  - You can use this as your primary directory for storing your homework files and stuff; create subdirectories for each of the homeworks and projects.
- **`pwd`**
  - Displays the current working directory
- Understand the path
  - The file system groups all files together in a hierarchical tree structure.
  - The top of the hierarchy is traditionally called the root.
  - Root is denoted by /
  - Any path name that starts with a / is an absolute path name
  - Traverse the path (/home/dxu), e.g. whatever pwd returned:
    - `cd /`
    - `ls`

        - `cd home`
        - `ls`
        - `cd dxu`
    - ~ expansion
        - ~ will be expanded to the absolute path of your home directory. For me, ~/teaching will expand to /home/dxu/teaching
        - ~ followed by a username will expand to that user's home directory. For example, ~dxu will expand to /home/dxu
- **`cd`**
    - `cd cs151`
    - Switches the current working directory. All file names that you enter are relative to the current working directory.
    - `cd ..`
    - cd with no argument (`cd ~`)
- File related commands
    - **`cp`**
        - `cp test.txt` and `long.txt` from `~dxu/handouts/cs151/labs/00`
        - `cp test.txt test2.txt`
        - Copies a file from one location to another.
    - **`mv`**
        - `mv test.txt test3.txt`
        - Moves a file from one location to another. Works across directories.
    - **`cat`**
        - `cat test2.txt`
        - `cat test2.txt test3.txt`
        - Displays multiple files in succession.
    - **`more`** (`less`)
        - `more long.txt`
        - Displays a file in multiple pages determined by terminal size.
    - **`clear`**
        - Clears all text on the screen except for a single shell prompt.
    - **`rm`**
        - `rm test2.txt`
        - **Unix does NOT have undelete!!!**
        - Flags you should know
            - –r Recursive Delete
            - –i Always Confirm
            - –f Force Delete
- & backgrounding
    - Appending an "&" after any command is a request to run the command in the background, and thus frees the current terminal prompt for other tasks. In general, anything that you expect to run for a while should be executed with &. For example, the editor.

The following section introduces you to the editor Emacs, which is my preferred code editor. If you have expertise with another, such as Vim, Atom, VS Code, that is acceptable as well. You may skip this section. You should however go with an editor that supports the Java language and indentation at the minimum.

Emacs

- Buffer-based editor
- Actually a fully functional LISP interpreter, but most people use it for editing purposes mostly. EXTREMELY POWERFUL; can do things like run a shell, do compilation internally, etc.
- To start emacs: `emacs <filename> &` (if filename does not exist, it will be created). Per above, the & is not absolutely necessary, but if you don't use it, you will find that you can no longer do anything else in the terminal that you started Emacs in.
- Emacs keyboard commands (you can use the pull-down menu too, but it'll be slower)
    - Syntax for commands
        - C-<letter> means "Control + <letter>"
        - M-<letter> means "Escape, followed by letter"
    - C-x C-s
        - Saves the current document
    - C-g
        - Cancel any mistaken command
    - C-_
        - Undo the last action.
    - C-k
        - Cuts the current line (starting from current cursor position).
    - C-spacebar
        - Sets the mark for the beginning of a text region
    - C-w
        - Cuts text from the last mark set to the current cursor. Note: in Unix, blocking text automatically copies it to the clipboard.
    - C-y
        - Paste. Can also be done with middle click.
    - C-a
        - Works just like the home key in Windows.
    - C-e
        - Works just like the end key in Windows.
    - M-x global-font-lock-mode
        - Colors your text for easy reading, similar to that in other editors.
    - M-x recover-file
        - If Emacs crashes, a "#" file will be created. Use this command to restore that file.
    - C-h t
        - Bring up the Emacs tutorial. Feel free to read this in addition to the Emacs guide sheet for other commands. Read until you can't stand it anymore. No one every finishes.

    - C-x C-c
        - Quits Emacs, will give opportunity to save.
- Unix FAQ
    - http://www.faqs.org/faqs/unix-faq/faq/part1/preamble.html
- History of Unix

    - http://www.bell-labs.com/history/unix/
- Emacs HOWTO:
    - http://www.tldp.org/HOWTO/Emacs-Beginner-HOWTO.html
- Emacs tutorial (yet another)
    - http://web.nwe.ufl.edu/writing/help/others/emacs/tutorial/index.html
- **Google and use key words "unix tutorial", "emacs tutorial"**

Look in `~dxu/handouts/cs151/labs/00` for code and data files related to this lab.

Exercise 1:
In your home directory, make directory `cs151`, under `cs151`, make directory `lab00`.
```
cd ~/cs151/lab00/
```

1. Use emacs (or your editor of choice) to create your own `HelloWorld.java`.
2. Compile and run.
3. Navigate back up to `~/cs151`
4. Make a new directory, say `lectures` there and copy the first set of lecture notes from my handouts directory to `lectures`

Exercise 2:
1. Copy the file `~dxu/handouts/cs151/labs/00/Lab00.java`.
2. Compile and run
3. Run with the correct command-line arguments so that it prints the following:
   ```
   Hi, CS151
   How are you?
   ```
4. Now modify the program so that when run with correct arguments, it prints:
   ```
   Hi, CS151 Students
   How are you?
   ```

Exercise 3 (only applies to those learning Emacs):
1. Make the following changes to your copy of the file `Lab00.java`:
   a) Delete the first two `System.out` lines (`C-k`)
   b) Now yank them back (`C-y`)
   c) Delete them again using set region then cut (`C-spacebar`, `C-w`)
   d) Now paste them back to the end of the file (`C-y`)
   e) Save and quit

Remote connections (not needed if you are in the lab)
- Terminal (mac)
- OpenSSH (windows 10)
- ssh
    - Connect to goldengate.cs.brynmawr.edu

Exercise 4:
Note that this is only necessary if you choose to not come to the lab to do your assignments. In that case you MUST make sure that you have a functional remote connection to our server that will allow you to complete your work. Here are a few options that you can choose from. Make sure one of them is working for you.

1. `ssh username@goldengate.cs.brynmawr.edu` – simply execute this command from a shell. This limits you to working in a terminal window, which is restrictive. The advantage is you are directly on the server, with access to all your files and there is no need to copy anything back and forth (as opposed to 3 below), and there is no set up on the local machine (as opposed to 2)
2. `ssh -X username@goldengate.cs.brynmawr.edu` – the additional flag "`-X`" sets up X-forwarding, which means you should be able to get all the windows as you normally would sitting in front of a lab machine. However, in order for this to work properly, an X server must also be installed on your local machine, which might require more set up (I tested for macs, installing XQuartz is all that is required. You are on your own with a Windows machine). If everything is working correctly, after login, you should be able to type commands into the server terminal as if you are sitting in front of a lab machine, but anything that pops a new window will show it on your local machine. For example, "`xterm&`" will start new server terminal windows on your local machine and "`emacs&`" will start a new emacs window similarly. "`code&`" to start VS code remotely.
3. Use "`scp`" (or some other ftp client) to copy all the files from the server to your own machine, program there and then copy them back – if you go this way, you must compile and run all your work on goldengate before you submit to make sure that they will execute properly on our server. Of course, you must also install Java on your local machine as well, along with your editor of choice.