**Lab 1**

You will be using **syccuxas01.pcc.edu** as your Linux programming environment. You can log on to it using [**PuTTY**](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html). Connect to port 22 via an SSH connection. Your username will be the same as your username on MyPCC and D2L. Your password will be **your pcc G number with a capital G**.

In order to develop software in the Linux command line environment, you will need to learn to use a variety of tools. You should start with these:

**cd** go into a directory

**mkdir** make a directory

**rmdir** remove an empty directory

**rm –r** remove a directory and everything in it  
(be *careful* with this)

**ls** list files and directories in a directory

**cp** copy a file

**mv** move a file or rename a file

**cat** display a text file

**less** display a text file that’s longer than your window

**rm** remove a file

**passwd** change your password

You will need to learn a text editor. There are three possibilities:

**nano** easier to learn for starters, but you should learn one of the following two powerful ones instead.

**vim** a powerful text editor with code-oriented features

**emacs** more powerful than vim, but a steeper learning curve  
(actually the programming language LISP masquerading  
as a text editor)

I will provide support for learning vim. I have never used emacs, so I can’t provide support for that.

**Things to do:**

* Log on to **syccuxas01.pcc.edu** and change your password.
* Get familiar with file system related unix commands
  + Protect your home directory

**chmod 700 ~**

* + Create a directory **cs260** in your home directory:

**mkdir cs260**

**ls**

* + Go inside **cs260:**

**cd cs260**

**ls**

* + Create a directory **lab1**
  + Go inside **lab1**
  + Copy **~lliang/files/hello.cpp** into **lab1** directory:

**cp ~lliang/cs260/labs/lab1/hello.cpp hello.cpp**

**ls**

* Use g++ to compile your hello world program and run it
  + Compile hello.cpp and call the executable myhello

**g++ -o myhello hello.cpp**

If you use c++11 featurs in your program:

**g++ -std=c++11 –o myhello hello.cpp**

If you want to be able to debug the code: -g option

**g++ -g -std=c++11 –o myhello hello.cpp**

* + Run myhello

**myhello** or **./myhello** if . is not in your PATH environment variable

* + Remove the executable

**rm myhello**

* Bundle your lab1 together in a tar ball, email it to yourself and extract from the tar file. Your future projects will be much bigger than one source file, so it’s necessary to create a tar file to bundle everything together for delivery.
  + Create a tar file of lab1 directory

**cd ..** (you should be in the parent directory of lab1)

**tar cvf lab1.tar lab1**

**ls** (you should see lab1.tar)

* + Email it to yourself, check email and save the tar file

**mailx –s “lab1 tar file” –a lab1.tar your-login-name**

You can type the message. A single period ‘.’ at the beginning of a new line will send the email.

To check email:

**mailx**

Typing the message number, e.g. 1, allows you to view the message

**1**

To save the file:

**w message-number** e.g. w 1

If you have an empty message body, the tar file will be saved as message number, e.g. 1 in the current directory.

If you have a message body, it will ask you to accept the file name or to give it a new name.

After viewing the message, it will be saved to mbox in your home directory. In order to view the message again, you need to use:

**mailx –f**

* + Extract from the tar ball and it will rebuild the lab1 directory. You can test this in a temporary directory

**tar xvf lab1.tar**

* Start learning one of the text editors.
  + nano is an easier one to learn
  + For vim, type **vimtutor** at the command line prompt.
  + Edit hello.cpp to make it do fancier things other than just saying “hello”, compile and run it.
* Use gdb, the debugger: the ability to use a debugger is essential for your success for this class. You should create a cheat sheet for the frequently used gdb command today for your own reference.

**g++ –std=c++11 –Wall –g –o exectutable-file \*.cpp**

**gdb executable-file**

[A gdb tutorial](http://www.unknownroad.com/rtfm/gdbtut/gdbtoc.html)

* Create makefile to make building your project easier and efficient. For all labs/projects this term, you are required to have a makefile.

[A makefile tutorial](http://mrbook.org/blog/tutorials/make/)

* Use valgrind to check memory leaks

**valgrind --tool=memcheck --leak-check=full executable-file**

* Lab 1 submission
  + Run the following command when you are in cs260 directory

**ls –R lab1 > lab1.txt**

* + Write a makefile for hello world project and test if it works. Make sure you include clean target

**cat lab1/makefile >> lab1.txt**

* + Create a cheat sheet, and call it gdbCmds, for frequenctly used gdb commands. It should at least include
    - How to start gdb
    - How to run your program
    - How to put in a break point
    - How to step your code
    - How to inspect variables
    - How to list the code
    - How to view the call stack
    - How to examine individual stack frame
  + ftp lab1.txt to your local machine and upload it into lab 1 dropbox