1 Accessing SCC

Your coding exercises will be done by using the Shared Computer Cluster (SCC) at the Massachusetts Green High Performance Computer Center (MGHPCC). The MGHPCC operates as a joint venture between Boston University, Harvard University, the Massachusetts Institute of Technology, Northeastern University, and the University of Massachusetts. http://www.mghpcc.org

To access `scc`, open a terminal and run:

```
ssh ~[username]@scc1.bu.edu
```

You are in a Unix shell now. You might want to have a unix shell on your laptop. For a Mac and Window 10 it is already there. Otherwise there a lot of options to put one on top of you machine!

My default shell on `scc` isn't bash, so I normally run the command `bash` to switch over to a bash shell. Before running `bash`, the command prompt looks like:

```
scc1:~ %
```

On the other hand, the default bash command prompt includes your username:

```
[username]@scc1 \textasciitilde]
```

Good! Next, let's navigate to our project's active directory.

```
cd /projectnb/alg504/
```

At `/projectnb/alg504/` you must create your own directory (folder to Window folks!) with your kerberos username using the unix command `mkdir username`. Then change directory with `cd username`. Do a `cd username` to go to you working area. This is where all your work should be. You will probably want to clone the class GitHub in your directory by by running

```
git clone  https://brower@github.com/brower/EC504
```

This way you have all the codes and instructions for the class. You can check status of the GitHub by `git status` and update your copy by `git pull`. This is ALL the GitHub commands you should use.! (Please don't add or remove anything to the class GitHub!)

To pass in your coding exercises (e.g. for Homework 1) you must create a directory `mkdir HW1` to to store the final solutions. The rest of your files and directory are your personal choices. To practice this take the `InClassTestCodeExercise` example on GitHub put the solution into HW0. This should only

have source code. To clean up the non-sources run `make -k clean`. This way I can practice finding your solutions before there is one due. `InClassTestCodeExercise` has two codes with useful stuff. In each director to compile and run just type `make -k .` (see GitHub `VerySimpleMake` for example of makefile examples)

If you'd like to play around with compiling and running code, you can log into an interactive shell by running:

```
qrsh -l h_rt=1:00:00 -P alg504
```

You will probably want to clone the class GitHub in your top level by running

That'll give you a one core interactive shell for one hour. Of course, it should be pretty clear how to try to get it for longer than an hour... but keep in mind, a longer interactive shell may not be available! There's a wonderful collection of information on BU's IS&T website for running jobs on SCC, if you're interested: `http://www.bu.edu/tech/support/research/system-usage/running-jobs/`

2 Unix Shell, Editors et al

There are lots of `standard` Unix commands and tools. There essential infinite but you only need a few for this course. All new big machines for high performance computing, big data, machine learning run on Unix machines so it is work learning a few tricks. This link had more than enough!

https://tjhsst.edu/ dhyatt/superap/unixcmd.html

Editors: I recommend using **Emacs**. Again you need only a few comnads. You can get it on you laptop with fancy menus to get used to it and learn the few necessary steps: Loading file, editing a file and saving it all that is absolutely necessary! Of you may prefer other editors on you laptop.

Graphics: It is fun and occasionally required to graph performance curves. A standard Unix tool is `gnuplot`

http://lowrank.net/gnuplot/intro/basic-e.html

Again the simplest few commands are enough. Pipe out raw number in columns and your all set to use *gnuplot*. In class instructions will l be given on very basics of *gnuplot*. Of course you can pass the output from your code to your favorite plotting tool and do a lot of fancy things if that is your desire. I tend to use `Mathematica` which is very fancy symbolic programming environment with beautiful interactive graphics and it is free software to any BU students, staff and faculty!

2.1 Hello World

Let's start with a simple hello world code!

```
#include <stdio.h>
```

```
int main(int argc, char** argv)
{
   printf("Hello world!\n");
   return 0;
}
```

Let's say we saved this to a file `hello.c`. It could be compiled by:

```
g++ hello.c -o hello
```

Okay, good, we did the obvious.