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

1.1 Fancy IDE Interface to SCC!

It is good to know basic Unix commands and tools describe below that are common to many High Performace Centers, although I have to admit they keep changing!

But now the SCC has very nice interface which overlay this with an extensive IDE. I recommend exploring this for its reflixibity and convenient methods. Make the SCC appear esseentially as application on your laptop! In class and on Slack we can all learn this together.

Go to https://scc-ondemand2.bu.edu/pun/sys/dashboard and start exploring.

If you go to Login Nodes on the top you can choose `>_scc1` and have your Unix shell.

If go to you will get `/projectnb/username` AFTER you have actual make this directory.

You can download to your latptop, drag and drop into CCS etc.

The tab **Interactive Apps** get you to all sort of direct access.

Best probably to firts use the *Good Olde Unix Way* describe next to understand the fancy stuff.

1.2 Good Olde Unix Way

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 it is already there. On a Window machine you can `google` to find lots of ways to put Unix shell on top of your machine. To make life easier this term everyone should also set up `VNC viewer`. This will help with graphics display at the CCS -a very convenient new tool.

Your default shell on `scc` may not be bash, so you should run the command `bash` to switch over to a bash shell. Then your home directory will have the prompt:

```
[username]@scc1 ~]
```

Note all your work should be done in the project directory. To navigate to our project's active directory.

```
cd /projectnb/alg504
```

At `/projectnb/paralg/` you must create your own directory (folder to Window folks!) with your login `username` using the unix command `mkdir username`. Do a `cd username` to go to you working area. This is where all your work should be done in this `project home`: `/projectnb/alg504/[username]`

You should clone the class GitHub in your directory by by running

```
git clone https://github.com/brower/EC504_2022
```

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!) The clone for GitHub will now be in your personal subdirectory `[username]/EC504_2022`. Finally to pass in your coding exercises (e.g. for Homework 1) you must create a directory `[username]/HW1` to store the final solutions. The rest of your files and directory are your personal choices but a nice practice is to create `[username]/HW1_working` directory for this exercise and copy from `HW1_codes` in your GitHub clone

When you have finished the homework your should only copy results `[username]/HW1`. This should include only source code, the makefile and written exercise, figure and output that are part of the homework. You can always clean up the non-sources run `make -k clean`.

This organization as you do each $HW0, HW1, HW2, ...$ will give you nice set of tools in each of the `HW#_working` and final solutions in `HW#` which be useful as we work on later exercises and when when you work on your project.

1.3 INGORE THIS FOR NOW!

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

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/`

The login node scc1.bu.edu is fine for all Homework Exercises in EC504. Maybe some will want to explore this for Projects. You can writen scripts to organize this and I believe the new **SCC OnDemand** interface https://scc-ondemand2.bu.edu/pun/sys/dashboard automates a lot of this. My Spring course EC526 on Parallel Algorithms for HPC, Machine Learning etc is where this larger system is appropriate for projects.

2 Unix Shell, Editors et al

There are lots of `standard` Unix commands and tools. There are essentially infinite resources, 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 worth learning a few unix 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 commands. 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! It is also provided on CCS and with VNC viewer has full graphics capability.

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. For all our Homework we will automate compilaton by running

```
make -k
```

in the directory with your code. After that you can run the progrqm by

```
./hello
```

Sometime we will pass parameters to the compiled code. Indeed this essential the way all unix commands are run. In priciple yous can view the source C code for Unix commands but I have never tried it. These command are have many option so they are probably pretty complex C codes! If you want see all the options for a command, `> man [command]`

Try `> man ls` You really didn't need to know this. I almost always us the command `>ls -al`