

## Preparation for Working in This Class

## Overview

This class requires you to work on the computers of the *Computer Science Department Network* (CSDN). The source code I provide, the live demonstrations that I do in class, and the projects that you'll have to write, all of these take place on those computers.

Unless you are a transfer student who is new to Hunter College this semester, or someone taking a class on an e-permit from another college, you are expected to be familiar with the use of the labs, remotely as well as in-person. Before this class starts, you need to make sure that you can login to your CSDN account. Anyone who cannot, whether it's because they don't remember their password, or it's been archived or locked because of inactivity, should follow the instructions on the webpage listed in the syllabus: https: //www.cs.hunter.cuny.edu/~csdir/ to fix this problem. In fact, everyone should read this page and make sure they know what procedures to follow in cse of problems with their accounts. If you avoided doing this for your first few years at Hunter, you are now at a disadvantage.

## Specific Instructions

Your first assignment in this class, to be completed before the start of the second class meeting, is to complete the following steps.

- Make sure that you can remotely login successfully to eniac using ssh and then into any of the 26 different cslab hosts. If you do not have an ssh client installed on your computing device, install it and configure it. If you do not even know what this means, you've got your work cut out for yourself.
- Make sure that you know how to edit files in the command line, using either vim, nano, pico, or emacs, the standard editors found in Unix systems.
- Edit your .bashrc file (e.g., vim .bashrc) so that the PATH variable includes the directory /data/biocs/b/student.accounts/cs493.66/bin

The instructions below explain how to modify the PATH variable..

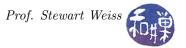
- Do not plan on editing files off line in a Windows application and uploading them. This is a Linux-based class and your files will not be compatible. The exception is that you can install a Linux subsystem on your device and edit files there.
- If you do not know the basics of using **bash**, then read one of the tutorials listed on the course webpage or find one on-line.

## Instructions For Modifying Your PATH Environment Variable

You must follow instructions *exactly* otherwise you will run into problems. You must complete the following instructions for this assignment. In these instructions, the dollar sign '\$' in the commands is the prompt character displayed by the shell; you do not type it. It is there to indicate that you are typing on the command line. Your own prompt might look different — there might be characters preceding it.

- 1. Using any *ssh client* on your computing device, remotely login to eniac.cs.hunter.cuny.edu using your CSDN username and password.
- 2. When you login successfully to eniac, ssh to any cslab host, for example:

\$ ssh cslab10



eniac is just a gateway host. It is the only host in the network visible to the "outside world" of the internet, and it is not a work machine. It exists just as a means to control who is allowed to use the computers in our network. The reason that you need to work on cslab10 right now is that only a few machines have the libraries to which this program links.

3. Copy the file /data/biocs/b/student.accounts/cs49366/bin/bash\_code to your home directory using the command

\$ cp /data/biocs/b/student.accounts/cs493.66/bin/bash\_code ~

The tilde "~" is another name for your home directory. To verify that you copied it correctly, enter the command

\$ ls ~/

You should see a file named bash\_code. If not, you did not copy it correctly. Try again. That file has a bash function in it named pathmunge, and a few lines after it that modify the PATH variable. It looks like this:

```
# Use pathmunge to avoid duplications and put paths in the best position
pathmunge () {
   if ! echo $PATH | /bin/egrep -q "(^|:)$1($|:)" ; then
       # the directory to be added is not already in PATH. Eureka!!!
       if [ "$2" = "after" ] ; then
          PATH=$PATH:$1
       else
          PATH=$1:$PATH
       fi
   fi
}
# Modify PATH variable to include path to cs493.66/bin directory
if [ -e /data/biocs/b/student.accounts/cs493.66/bin ] ; then
   pathmunge /data/biocs/b/student.accounts/cs493.66/bin after
    export PATH
fi
```

4. Save a copy of your existing .bashrc file to a new file named .bashrc\_old, just in case you make a mistake with the remaining instructions:

\$ cp ~/.bashrc ~/.bashrc\_old

5. Now, modify your .bashrc file by appending the contents of the file you copied in step 3 to the end of the .bashrc file. You can either type the contents of that file into your .bashrc file after its last line, or you can just append that file using the cat command, as follows:

```
$ cat ~/bash_code >> ~/.bashrc
```

If you know how to use an editor such as nano or vi, you can edit the .bashrc file instead. *Do not attempt to do this in any other way*. Do not download the file and edit it on your own machine. Do not copy and paste code from this document into your file. Unless you use the above command, it may not do what it is supposed to do in the end. Do not do it more than once. If you need to do it again, first delete the .bashrc file, copy .bashrc\_old to .bashrc and try again.

**bash** is both a command line interpreter and a programming language. The file that you append has lines that define a **bash** function named **pathmunge()**. In English, to *munge* is to transform or mix up data. This function has two parameters; the first is a directory pathname and the second is an optional word. The optional word "after" may be used as its second argument. **pathmunge()** checks



if the directory is already in the PATH variable and if it is, it does nothing. If it isn't, then it either appends the directory to the PATH or prepends to the PATH depending on whether the word "after" is supplied. For example, if the current value of PATH is the string "/bin:/usr/bin:/usr/local/bin:", and I type

pathmunge ~/bin after

then the new value of PATH will be the string "/bin:/usr/bin:/usr/local/bin: ~/bin".

If you already have pathmunge() in your .bashrc file, you only have to add the lines that call it. Following the definition of the pathmunge() function are lines that add a new directory to your PATH variable so that any executable in that directory can be run just by typing its name without a leading path. The directory that gets appended is the one that I use for class-specific commands. If I wanted to append another directory such as the current working directory, whose name is a single dot '.', to the PATH variable, I would add the following line at the end of my .bashrc file:

pathmunge . after

6. Once you have modified the .bashrc file, you must run the following command to tell the bash program to reread its contents:

source ~/.bashrc

7. Your PATH variable is now modified so that you can run commands in the class's directory, /data/biocs/b/student.accounts/cs493.66/bin without having to enter a lot of text.