Assignment 1

Preface

This assignment has a simple objective: to prepare you for doing the rest of the assignments and for following along during the lectures. It will also introduce a few concepts to help you to understand why you are asked to do the various tasks described below.

Overview

In this course, we use the ***Computer Science Department Network*** (***CSDN***) for many assignments and in-class activities, as well as for accessing certain programs I have written for the class. The CSDN is the collection of computers named `cslabN` (N=1,2,...,26), as well as the gateway machine, `eniac`, and the file and network services server, `biocs`[1], on which they depend. All of these machines are part of the ***domain*** `cs.hunter.cuny.edu`. This means that their ***fully qualified domain names*** are names like `eniac.cs.hunter.cuny.edu`. and `cslab6.cs.hunter.cuny.edu`.

This class has its own directory in the file system. The ***absolute pathname***[2] of this directory is

```
/data/biocs/b/student.accounts/cs340_sw.
```

Within this directory are subdirectories of various kinds. Among them is a directory named `bin`, which contains executable programs that you can run. (In Unix, *bin* is short for *binaries*, another name for *executables*.) For example, there is a program in `bin` named `submithwk_cs340`. This program is the one you'll use when your homework assignment has to be submitted on the CSDN server. To run this program from any directory in the file system, you would have to type the very long pathname

```
/data/biocs/b/student.accounts/cs340_sw/bin/submithwk_cs340
```

This is a lot to type. Most operating systems, including UNIX/Linux, have a collection of "environment variables" for every user that can be modified to customize the way in which you interact with them. One of them is named `PATH`. The `PATH` environment variable is a colon-separated list of directories within which `bash` looks for the names of commands or executable files that you type on the command line. For example, your might look like this:

```
/usr/local/sbin:/usr/local/bin:/usr/bin
```

To make it easier to run the programs in this directory, you can change your `PATH` environment variable so that you can type just the name of the program on the command line to execute it, instead of having to type that very long pathname for it every time you want to run that program. To do this, you need to modify a file in your home directory named `.bashrc`. In particular, you will make changes that modify the value of the `PATH` environent variable.[3] When you're finished, your `PATH` will be

```
/usr/local/sbin:/usr/local/bin:/usr/bin:/data/biocs/b/student.accounts/cs340_sw/bin
```

By following the instructions below ***exactly*** you will ensure that you have modified your `.bashrc file` so that you are able to run the commands in that `bin` directory just by typing their names.

[1]The name of the server is temporarily changed to CS-sm4.
[2]An absolute pathname of a file is the pathname starting at the root of the file hierarchy, '/'.
[3]For more explanation, you can read `http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci132/slides/Lesson_03.pdf` or for a more detailed explanation, see the Bash Reference Manual at `https://www.gnu.org/software/bash/manual/bash.html` and look up "Bourne Shell Variables".

Deadline

You must complete this assignment before its ***deadline***, which is ***Monday, September 11, at 7:00 PM. After that, you will not receive credit for completing it.***

Instructions

Note: In many disciplines, exactness and precision are important for success. In this class, 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. The syllabus has information about how to get an ***ssh client*** if it is not already installed on your machine. Do not proceed until you have installed or found your ssh client. If you do not remember your password or username, follow the instructions at `http://www.compsci.hunter.cuny.edu/~csdir/`. From my desktop computer, which runs Linux, I would open a terminal window and type

   ```
   $ ssh sweiss@eniac.cs.hunter.cuny.edu
   ```

   because `ssh` is a command in Linux that runs the ssh program. If you cannot login for some reason, this is the first problem you have to solve. Everyone in the class has an account on the network and therefore has a username and password. If you have not logged in for a long time, your home directory was removed or your account was archived and you will be unsuccessful at logging in. Contact our system administrator by sending email to `cstechsp@hunter.cuny.edu` if you cannot login. ***Do not expect an immediate reply!*** *Do not send email to any other address*!

2. When you login successfully to `eniac`, ssh to `cslab12`.

   ```
   $ ssh cslab12
   ```

   You can not do this assignment on `any computer other than cslab12.`

   `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.

3. Copy the file `/data/biocs/b/student.accounts/cs340_sw/bin/bash_code` to your home directory using the command

   ```
   $ cp /data/biocs/b/student.accounts/cs340_sw/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.

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 using the following command:

```
$ 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. 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`".

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. The program named `submithwk_cs340` in the directory `/data/biocs/b/student.accounts/cs340_sw/bin` can be run only on `cslab12` at this time. You must run this program to submit your new version of the `.bashrc` file. This command makes a copy of your file in the correct directory with the correct name and permissions. ***In general you will use this command to submit all assignments given in this class***.

8. The `submithwk_cs340` command normally requires two arguments: the number of the assignment (1 in this case) and the filename of your homework file. With just two arguments, it expects the file to be a plain text file.

   Therefore, to submit this assignment, you do two things: navigate to your home directory and run the command:

```
$ cd ~
$ submithwk_cs340  -t  1  .bashrc
```

   The program will copy your `.bashrc` file into the directory

```
/data/biocs/b/student.accounts/cs340_sw/hwks/hwk1/
```

   and if it is successful, it will display the message, "`File hwk1_`*`username`*` successfully submitted.`"

   where *`username`* is your username. You will not be able to read this file, nor will anyone else except for me. But you can double-check that the command succeeded by typing the command

```
ls -l /data/biocs/b/student.accounts/cs340_sw/hwks/hwk1
```

and making sure you see a non-empty file named `hwk1_username` where `username` is your user name and whose date of last modification is the time at which you ran the command. The size of that file should be **exactly the same** as the size of your `.bashrc` file, otherwise you did something wrong.

9. ***You can do step 8 as many times as you want. Newer versions of the file will overwrite older ones.***

Grading Rubric

This assignment is 1% of your final grade. Not much, you are thinking perhaps, but if you do not do it you will not be able to do some of the remaining assignments. You will receive full credit if the file you submit has exactly the lines shown in the above instructions and has been submitted by the assignment **deadline**. If the lines do not match exactly, the script that checks them will fail and you will not receive any credit. If it is not submitted by the deadline, you receive no credit.