Assignment 1

Preface

This assignment is a warm–up programming exercise. The program that you need to write is small, but you'll have to learn a few things to write and run it.

Instructions

For this first assignment you'll write a short C program. This program will have just two lines of output. The first line will contain the username of the user who runs it, along with their userid and home directory, all in a single sentence of the following form.

```
My username is sweiss, my userid is 1220, and my home directory is /data/biocs/b/cshome/sweiss.
```

in which *sweiss* is replaced by the person's username, *1220* would be replaced by their userid, and */data/biocs/b/cshome/sweiss* would be replaced by the absolute pathname of their home directory. ***The output must be exactly in this form, absolutely, without exceptions***!
The second line of output will be one of the following two lines:

```
My shell is  path-to-shell
```

in which *path-to-shell* is replaced by the actual value of the `SHELL` environment variable or

```
The value of my DISPLAY variable is display-variable-value
```

in which *display-variable-value* is replaced by the actual value of the `DISPLAY` environment variable.

Which line is your program supposed to output?

- If your actual userid on our computer network is an even number, your program must output the `SHELL` variable line.
- If it is an odd number, your program must output the `DISPLAY` variable line.

If I ran this program on the server, the session would look like the following:

```
$ hwk1
My username is sweiss, my userid is 1220, and my home directory is /data/biocs/b/cshome/sweiss.
My SHELL is /bin/bash.
```

If my userid were 1221, then the output would be

```
$ hwk1
My username is sweiss, my userid is 1220, and my home directory is /data/biocs/b/cshome/sweiss.
The value of my DISPLAY  variable is localhost:10.0.
```

Error Handling

In this class, you will not receive any credit for a submitted program that can crash. Note the "can" in this sentence. It may not crash when you run it, but that does not mean it can't crash when I run it. This means that every time you call a function, you must check whether it succeeded or failed. If it failed, your program must exit with some error message, such as "The call to foo() failed." It also means that all strings must be NULL-terminated.

Guidance

- To write this program, you'll need to learn how to use a few of the functions and system calls we saw in Chapter 1. I suggest looking at the programs in Chapter 1 of the *demos* directory. There you'll see an example from which you can start.
- The program will need to include all required header files. The man pages for every function and system call it may need show which header files to include.
- What is your userid on our system? Chapter 1 shows you the command for finding it.
- The program has to print out the userid of the user who runs it. There is a function that can get that userid. What is it? A search of the man pages (`apropos` command) can find it for you.
- The program needs to format the output. It's time to learn how to use `printf()`. The `printf()` man page (Section 3 of the man pages, not Section 1) is long but it has everything that you need to know. The demo programs are a good starting point as well.
- Because the program has to print a single sentence with the values of various environment variables, it will need to copy strings. This will be clear when you read Chapter 1 carefully and when you read the appropriate man pages. Functions for copying strings are abundant in C. If you've never used any, you can either:
  - read the C Language Reference Manual, Section 7.24, to learn about the functions that can do this, or
  - search the man pages for a set of candidates, using `apropos`. If you try this, use the `-a` option for "and-ing" the search terms, and try entering "`copy`" and "`string`" together.

Instructions for Submitting the Assignment

1. Use the `submithwk_cs49366` program to submit your program. It can be run on any `cslab` host, so login to a `cslab` host when you're ready to submit. To submit, if your file is named `myhwk.c`, you'd enter

   ```
   $ cd ~
   $ submithwk_cs49366  -t 1  myhwk.c
   ```

   The program will try to copy `myhwk.c` into the directory

   ```
   /data/biocs/b/student.accounts/cs493.66/hwks/hwk1/
   ```

   and if it is successful, it will display the message, "`File hwk1_`*`username`*`.c 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 entering the command

   ```
   ls -l /data/biocs/b/student.accounts/cs493.66/hwks/hwk1
   ```

   and making sure you see a non-empty file named `hwk1_`*`username`*`.c` where *`username`* is your user name and whose date of last modification is the time at which you ran the command.

2. ***You can do step 1 as many times as you want. Newer versions of the file will overwrite older ones.***

Deadline

You have one week to do this assignment. It must be submitted before its ***deadline***, which is ***Thursday, February 8, at 7:00 PM. After that, you will not receive credit for completing it.***

Grading Rubric

This assignment is **8%** of your final grade. The output must be correct and the program must conform to the rules written in the ***Programming Rules*** document on the class's webpage. It must be thoroughly documented.