CMSC 201 Fall 2017
Lab 01 – Hello World

Assignment: Lab 01 – Hello World
Due Date: Friday, September 8th by 8:59:59 PM
Value: 10 points

At UMBC, the GL system is designed to grant students the privileges of running a website, storing files, and even developing applications. GL itself is a Linux-based system that we will interact with using a command line interface (CLI). All major operating systems (Windows, Mac, and Linux) can log into GL.

In order to log into GL, you will need to have a Secure Shell (SSH) client. SSH is a network protocol that connects a client computer to a server securely, without the risk of exposing your password or any other sensitive data to anybody watching the network traffic. It is a leading standard in remote login. So, what does all of that mean? Simply put, SSH allows you to use your home computer to connect to a server (*i.e.*, `gl.umbc.edu`) and issue commands as if you were sitting at that computer. So, once you SSH into GL, you can do everything you would be able to do from a computer on-campus. This client is built into Mac and Linux. For Windows, we recommend downloading the open-source client PuTTY.

In Lab 1, we will be logging onto GL and setting up folders for 201 in your home directory. We'll also create a simple python program, and turn it in using the `submit` command. Finally, you'll send your TA an email with a bit of information about you, so they can get to know you better.

This lab is a way for you to make sure that everything needed to complete the homeworks works for you.

Part 1: Logging onto GL

For Windows Users Only

Step 1A:

Watch this video (https://www.youtube.com/watch?v=uE3VpOQi_GQ) , and download and install the PuTTY program. PuTTY will allow you to log into GL from your machine.

Step 1B:

Watch this video (https://www.youtube.com/watch?v=DH3of3tuBxA) and follow its instructions to log onto GL using PuTTY.

For Mac Users Only

Step 1:

Watch this video (https://www.youtube.com/watch?v=UTU-CTH_xLw) and follow its instructions to log on to GL using the Terminal application already installed on your machine.

It is recommended that you do **not** resize the Terminal window horizontally, as the wider window often leads to very weird behavior with how the text is displayed, as seen in the red text in the screenshot below. (You may make the Terminal window taller or shorter, but do not make it wider.)

```
        place += 1
        password = input("Please enter a password: ")

  # if none of these if conditionals were true, then passwordValid will stillb    u       and        ft
he
      print("Thanks for picking the super-secure password", password)

main()
```

Part 2: GL and your home directory

In this part of the lab, we will log onto GL and create some folders in your home directory. These folders will be where you store your labs and assignments for 201 this semester.

Since GL is a command line interface, we cannot navigate it using our mouse. Instead, we must type in commands to tell it what we want to do. The solid green rectangle (or something similar on your machine) is where our cursor is. After typing in a command, you must hit enter for the system to know you are done.

Step 2:

After logging onto GL in Part 1, you are in your home directory (remember, directory is just another name for folder). We will first make a 201 directory in your home directory. The command we will use to do this is **mkdir**, which stands for "make directory." We also have to tell the system what the name of the directory will be. In this case, the name is "201", so type **mkdir 201** now, then hit enter. Remember that Linux commands are case sensitive!

```
linux3[1]% mkdir 201
linux3[2]%
```

If you make a mistake in naming your folder, you have two options. You can either remove that directory or you can rename the directory. For example, if you accidentally created a folder called "**2001**" you could remove it by typing the command **rmdir 2001** and hit enter. Be very careful that you do not remove the wrong directory. There is no recycle bin or "undo" button.
The safer option would be to rename the directory. To rename the directory, you use the "move" command, or **mv**. To use it, you would type **mv 2001 201**, and hit enter.

```
linux3[2]% mv 2001 201
linux3[3]%
```

The **mv** command would rename our folder from "**2001**" to "**201**" instead. (As another example, if we wanted to rename it from "**2201**" to "**201**" we would use the command **mv 2201 201** to do so.)

Step 2B:

This step is <u>only</u> for students who are <u>retaking</u> CMSC 201. If you are repeating CMSC 201, you likely already have a folder named "201" from last time. You will want to follow the steps below to move your old folder to a new name before creating one for this semester. While you'll still have access to your old assignments, it is *highly* recommended that you *don't* refer back to them, and instead do the work again for this semester.

```
linux3[23]% ls
201
linux3[24]% mv 201 old_201
linux3[25]% mkdir 201
linux3[26]% ls
201  old_201
linux3[27]% █
```

Step 3:

We now want to change our "current directory" to this new 201 directory. (In other words, we are going "inside" the 201 folder.) The command is `cd 201`, which means "change directory to the 201 folder." Type `cd 201` now, then hit enter.

```
linux3[2]% cd 201
linux3[3]% █
```

Step 4:

In order to see the contents (like files and subdirectories) of a directory, we need to use the `ls`, or "list," command. Since the 201 directory was just created, it should be empty. You can check this by typing `ls`, then hitting enter. Nothing will be displayed, which means the directory you're in is empty.

```
linux3[3]% ls
linux3[4]% █
```

Step 5:

We want to create two directories inside the 201 directory: one called Homeworks, and another called Labs. For this step, we again use the **mkdir** command. First type **mkdir Homeworks**, then hit enter. Once you've done that, type **mkdir Labs**, and hit enter again. Finally, you can type **ls** to verify that you now have two directories named "Homeworks" and "Labs" in your 201 folder.

```
linux3[4]% mkdir Homeworks
linux3[5]% mkdir Labs
linux3[6]% ls
Homeworks   Labs
linux3[7]%
```

Step 6:

We will next create a directory called hw1 inside the Homeworks directory. (When you work on your first homework, you will save your Homework 1 files inside this directory.)

First, we need to change our current directory to be inside the Homeworks folder. To do this, type **cd Homeworks** and then hit enter. Next, to create the hw1 folder, type **mkdir hw1**. You can then type **ls** to check that the hw1 directory has been created.

```
linux3[7]% cd Homeworks
linux3[8]% mkdir hw1
linux3[9]% ls
hw1
linux3[10]%
```

At this point in the lab, your GL account has directories that look like this:
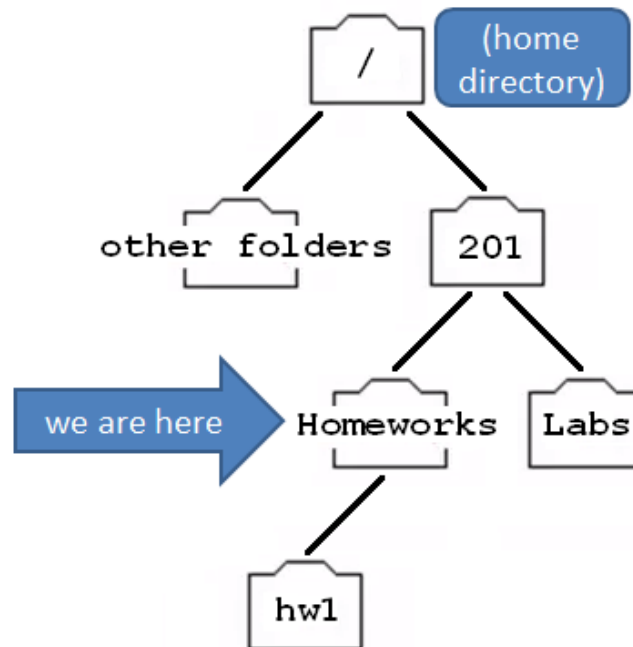
Step 7:

Now we want to create a lab01 subdirectory inside Labs. Before we can do this, we need to move to the Labs directory. First, we go up one "level," back to the 201 directory. To do this, we type `cd ..` ("cd" followed by a space, then two periods). The two periods (`..`) represents the parent directory, or the directory above the current location. The parent directory of Homeworks is 201, so we are now back in the 201 folder.

```
linux3[10]% cd ..
linux3[11]%
```

Step 8:

Let's just double check that we are actually back in the 201 directory. We do this asking GL to "print working directory," using the **pwd** command. Type **pwd** and hit enter. It will show you the full path, including "umbc.edu" and your username. If we look at the end, we can see that we are in the 201 directory, just like we wanted to be.

> We are in the 201 directory, which is inside the home directory.

```
linux3[11]% pwd
/afs/umbc.edu/users/k/k/k38/home/201
linux3[12]%
```

> This part of the path will depend on your username.

Step 9:

Now we can create the lab01 folder inside the Labs directory. First, we go into the Labs directory by typing **cd Labs**. Next, we create the lab1 directory by typing **mkdir lab01**. We can verify it was created by typing **ls** to list the contents of the Labs directory. Once it's created, we can move into the new directory by typing **cd lab01**. We can verify that lab01 is now our current directory by typing **pwd**.

```
linux3[12]% cd Labs
linux3[13]% mkdir lab01
linux3[14]% ls
lab01
linux3[15]% cd lab01
linux3[16]% pwd
/afs/umbc.edu/users/k/k/k38/home/201/Labs/lab01
linux3[17]%
```

> We are in the lab1 directory, inside the Labs directory, inside the 201 directory.

> This part of the path will depend on your username.

Part 3: Creating your first Python program

We are now ready to create our first Python program. Using the emacs editor, we will create a "Hello World" program, the traditional first coding project. (Make sure that when you complete this part of the lab, your "present working directory" is lab01!)

Step 10:
Before we can begin, we need to create the file we will edit. We'll use the **touch** command to create a new (empty) file, by giving it the filename. In this case, we want to create a file called "lab1.py", so type **touch lab1.py** now, then hit enter.

```
linux3[17]% touch lab1.py
```

Now that the file has been created, we can open it for editing, by using the emacs editor. By using the command **emacs lab1.py**, we are opening the file lab1.py in the emacs editor. (The emacs editor is similar to Notepad on Windows, or TextEdit on a Mac. It's nothing fancy – it just lets you type and edit files.)

```
linux3[18]% emacs lab1.py
```

The emacs editor will replace your entire window of text, and should look similar to the example below.

```
File Edit Options Buffers Tools Python Help




-UUU:----F1   lab1.py           All(1,0)      (Python)----------
(New file)
```

Step 11:

Let's code our first program! This program will print out a short greeting when we run it. Type in the line of code shown below, but where the code says **FULL NAME HERE**, replace it with your full name.

```
File Edit Options Buffers Tools Python Help
print("Hello World! I am FULL NAME HERE!")

-UUU:**--F1   lab1.py          All(1,42)    (Python)----------
```

It's important that you type in the code exactly as shown. Every character is important, and even small changes can keep the program from working. The only change should be replacing **FULL NAME HERE** with your full name.

Step 12:

Once you are done typing your code into emacs (and have replaced **FULL NAME HERE** with your name), you need to save your work and exit. First, save it by pressing **CTRL+X** and **CTRL+S**. (Press the CTRL and X keys at the same time, then the CTRL and S keys at the same time. You do not need to hold the CTRL key down for the entire command.)

If it worked, you should see a confirmation message at the bottom of your screen, similar to the one below.

```
-UUU:----F1   lab1.py          All(1,42)    (Python)----------
Wrote /afs/umbc.edu/users/k/k/k38/home/201/Labs/lab01/lab1.py
```

Now that you've saved your code, you can exit by pressing **CTRL+X** and **CTRL+C**. You will exit out of emacs, and be back to the command line.

Part 4: Running your first Python program

Step 13:

We will be using Python 3 for all of your assignments. However, GL runs Python 2 by default, so we need to tell it we want to use Python 3 instead. To do this, type in the command **`scl enable python33 bash`** and hit enter.

```
linux3[18]% scl enable python33 bash
bash-4.1$ █
```

You'll know it worked because the prompt changes. You must enable Python 3 every time you want to run a Python program! If your prompt starts with "**`linux`**" as it does in the first line above, you are not running Python 3.

<u>TIP</u>: You are still able to use commands like **`ls`**, **`cd`**, and **`emacs`** when the prompt reads "**`bash`**". (You don't need to exit bash to use GL, or to submit your assignments.)

Step 14:

To run your code, type in **`python lab1.py`** and hit enter. Your program should run as seen below, but with your name instead of **`FULL NAME HERE`**.

```
bash-4.1$ python lab1.py
Hello World! I am FULL NAME HERE!
bash-4.1$ █
```

If your program does not run, but instead gives you an error message, you have at least one error. Just open the file up in emacs again (type **`emacs lab1.py`** after the **`bash-4.1$`** prompt) and take a close look at your code. Make sure it matches the code from Step 11.

For example, here's the error if the quotes are outside of the parentheses:

```
bash-4.1$ python lab1.py
  File "lab1.py", line 1
    print"(Hello World! I am FULL NAME HERE!)"
                                              ^
SyntaxError: invalid syntax
bash-4.1$ █
```

Part 4: Submitting your first Python program

Step 15:
Watch this video (https://www.youtube.com/watch?v=r0otsJZ1ry0) for an explanation of how GL works. Follow the instructions in the video to submit your lab1.py file.

Step 16:
After submitting the file (by following the instructions in the video from the previous step), we can verify that it went through by using the **submitls** command. The command needs to know the name of the class and the name of the assignment – these should be the same as for the **submit** command you just ran: cs201 for the class, and LAB1 for the assignment.

```
bash-4.1$ submitls cs201 LAB1
total 21
drwx------    2 emcgov1 rpc  2048 Aug 29 15:27 .
drwx------  507 emcgov1 rpc 18432 Aug 27 15:01 ..
-rw-r--r--    1 k38     rpc    22 Aug 29 15:27 lab1.py
bash-4.1$
```

On the far right of the last line, we can see that the file lab1.py was submitted successfully. The other two shown ("**.**" and "**..**") are not important for now.

Step 17:
To exit out of bash, and back to the linux prompt, type **exit** and hit enter. (You will need to re-enable Python 3 again if you want to run another Python program.) To exit from GL, type the **exit** command again and hit enter.

```
bash-4.1$ exit
exit
linux3[19]% exit
```

This will log you out of GL and close your terminal window.

Part 5: Emailing your TA

Step 18:

Use your @umbc.edu email account to send your TA an email, with a subject line of "CMSC 201 Intro -- Section ## -- YOUR_UMBC_ID". Inside, your email should contain the following information.

> My preferred name (or nickname): FILL_OUT_HERE
> My pronouns: FILL_OUT_HERE
> My major: FILL_OUT _HERE
> My programming experience: FILL_OUT _HERE
> Fun fact about me: FILL_OUT _HERE

For more information about "my pronouns" and what that means, click here. For your programming experience, it's fine if you simply put down "none."

You can find out who your TA is under the "Sections" page on the course website. Match the section number from your CMSC 201 discussion section (you can find it on your schedule) to the section number in the table. Make sure that the time and place also match the discussion in your schedule!

Part 6: Wrapping Up

Step 19:

Watch this video (https://www.youtube.com/watch?v=ci42iOOtTRI) for an explanation of the "life cycle" of remote programming on GL.

Step 20:

Finally, let's discuss how the labs will be graded in this course.

There will be 13 labs over the course of the semester; your best 10 scores will be used to compute your lab average. To receive credit for attending the lab and completing the assignment, your work must be verified by your TA.

You must attend your assigned lab section and the lab assignment must be completed **during** the assigned lab time (not before).

During the lab, your TA will explain the lab assignment, provide assistance as needed, and record your successful completion of the assignment.

Lab assignments are graded on a scale from 0 to 8 at the discretion of the TAs.

Score	Description
8	Successfully completed the lab assignment
6	Made a good attempt to complete the lab assignment
3	Attended the lab, but made little or no effort to complete the assignment
0	Did not attend the lab, or arrived more than 20 minutes late

If you made a good attempt to complete a lab assignment within the allotted time, but were unable to finish it, you may complete the assignment on your own and show it to your TA to have the grade changed from a 6 to a 7. This must be done within one week of the original lab date.

To request that a lab grade be changed from a 6 to a 7, you must show the completed assignment to your TA *before* your next lab session. For the final lab of the semester, you must show the completed lab to your TA no later than the last day of classes.

Labs are worth a total of 10 points each. The final two points are earned for each lab by completing the Pre Lab quiz, which will be available on Blackboard starting the Friday before the lab (at 10 AM), and closing on Mondays at 10 AM. The Pre Lab quiz ensures that you have a basic understanding of the material that will be needed for lab the following week.

Special thanks to James Gerity who wrote a GL primer on his website:
http://userpages.umbc.edu/~jg5/logon-howto.html

Mac logon instructions based on a guide from Rackspace:
https://support.rackspace.com/how-to/connecting-to-linux-from-mac-os-x-by-using-terminal/