CISC106 Summer 2011 Lab01

- This lab an all subsequent labs will be due Sunday at 11:55 PM EDT on Sakai.

Basic UNIX tutorial

1. In this and subsequent courses you will be using UNIX and the command line. This section will give you a brief introduction to some useful UNIX commands. If you do not know how to open a command console to Strauss, ask the TA for help.

2. You should be staring at a prompt, `strauss.udel.edu%`. You can type in commands here to make the computer do things. Type `pwd` and hit enter. You should see something similar to `/home/usra/121/74097` printed below the prompt where you just entered pwd, followed by a new prompt - This is the pattern for entering commands at a command line and reading their output. pwd stands for ”present working directory”; it shows the directory (aka *folder*) you're currently in. The strange path (thing with all the slashes that printed after entering pwd) shown is your *home directory* - it's a folder for you to keep whatever data you want. Don't worry about remembering this path - there's an easy way to always get back to it (as you'll see in a moment.)

3. The command `cd` is used to change the present working directory. Type `cd /` and then hit enter - now you're in another folder. If you enter `pwd` again, you'll now just be greeted with a `/`.

4. Type `ls` and hit enter to get a listing of all the files and folders inside the current folder (present working directory.) There's a lot of junk in this folder! Fortunately, we don't care about any of it. To get back to your home directory, just type `cd` and hit enter. When you give the cd command without specifying a path, it always takes you back to your home directory. If you do an `ls` now, you should see whatever files and folders are in your home directory.

5. Of course, you can make your own folders to store stuff inside of in your home directory. You make new folders with the `mkdir` command. Type `mkdir stuff` and hit enter to get a new folder aptly named 'stuff'. Now you can `cd stuff` to go to this folder. If you type `pwd` now, you should see the path to your home directory with a `/stuff` on the end of it. Type `cd ..` to go back to your home directory. Just like `cd` by itself will always take you back to your home directory, `cd ..` will always take you to the folder that contains the one you're currently in. So if your pwd is `/some/really/good/place`, `cd ..` will change it to `/some/really/good`.

6. You can get rid of folders you don't want with `rmdir`. First, let's do another `ls`. You should now see that you have a 'stuff' folder inside your home directory. Let's get rid of that folder. Enter `rmdir stuff`. You should just be silently given another prompt - but if you do an `ls` again, you'll notice that stuff is no longer among the things in the listing. [1]

7. These are the basic commands you'll need to know to do the rest of this lab. To learn more about the UNIX command line, head on over to this site:

[1] **NB**: `rmdir` will only delete a folder that's empty - if there are any files or other folders inside the folder you want to get rid of, `rmdir` will fail. There *is* a way to delete folders with stuff in them, but if you want to know how to do that I'll let you look it up. Don't say I didn't warn you, though! Once you delete something, it's gone for good!

`http://www.ee.surrey.ac.uk/Teaching/Unix/`

and read through Tutorials One through Six. If this is your first time using UNIX, I strongly urge you to work carefully through each tutorial.

Programming assignment

1. The first thing you should do is make a folder in your home directory called lab01. Next, run firefox and go to `http://www.udel.edu/CIS/106/keffer/11F/labs` There you can download the file 'test.py' (you may have to right-click on it and go to save as.) You should be able to find the lab01 folder you just created (you might have to go back a folder or two to be in your home directory) in the download dialog, so you should save it there. If you go back to the command prompt now and `cd` to lab01, you should be able to do an `ls` and see the test.py file you just downloaded.
2. Now, at the command prompt enter `python-2.7 test.py` [2] If you see the message 'That wasn't so bad, was it?' printed to your console, everything is working right.
3. What you just did was run a very simple python program. Next you're going to create another simple python program. To do that, you'll need to open up a text editor. Enter `gedit&` at the prompt[3] and you should soon be greeted with a window that looks similar to notepad on Windows.
4. Now enter the following lines in gedit:

   ```
   x = 10
   a = 15
   b = 20
   print x
   x = a + b
   print x
   a = 5
   print x
   x = 9
   a = 13
   b = 42
   print x
   ```

   Each of these lines is a *statement* in Python. You'll notice there are two different kinds of statements here - *assignment* statements which assign a *value* (such as 10 to a *variable* (such as x) and *print* statements, which will simply display values to the screen (if you print a variable, the value displayed will be the value assigned to that variable.)
5. Now save this to a file by clicking the save button in gedit. In the save dialog that pops up, you should notice that the "Save in folder:" is the lab01 folder you created (since we ran gedit in this folder.) Name the file `simple.py` and then click save.

[2]The reason for the -2.7 is because there are several different versions of python installed on Strauss. 2.7 is the one we want, though.

[3]if you forget to include the &, you'll lose the prompt and you'll have to quit gedit to get it back!

6. Start a new file (by clicking the new button). In this new file, put in what you think the output of simple.py will be (hint: there are four print statements, so you should have four lines of output.) Don't worry about the accuracy of your guess - you won't be graded on it being right or wrong.

7. Now go back to the command line (you can leave gedit open since you'll be going back to it in a moment) and run simple.py by entering `python-2.7 simple.py`. What did it actually output? Is this different than what you expected it to be? Whether or not it's different or the same as what you expected, can you explain why the output is what it is? Answer these questions in the same file you put your guess in. (You will be graded on the quality of your explaination.) Save this file (name it whatever you'd like) and submit it on Sakai along with simple.py

You should submit simple.py along with the file containing your answers and any other docs required by your TA on Sakai.