**CS 251: Linux & BASH (Inlab)**

- Due: 8:10 PM $9^{th}$ August
- Please write (only if true) the honor code. If you used any source (person or thing) explicitly state it. You can find the honor code on Piazza.

## Overview

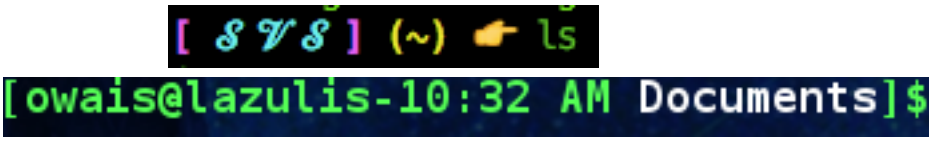
Most of you have Windows 10 installed on your laptops. You've also probably read that Microsoft was adding the functionality of running bash scripts and commands in Windows 10. If you weren't aware of that check out this link. Now who or rather why would someone want to run boring monotone color commands when you have the awesome GUI and Cortona provided by Windows? In the next couple of labs we'll try to introduce you to the power of bash and perhaps reverse the question!!!

## A. Terminally Pretty

Most of your terminal customization that apply only to your account go into the `~/.bashrc` file whereas the global customization that apply to all the users go into `/etc/bash.bashrc`. In fact the `~/.bashrc` file is the second file that is executed when you start a terminal, the first being `/etc/bash.bashrc`, but you usually don't have the permissions to modify `/etc/bash.bashrc` and hence we'll concentrate on playing around with `~/.bashrc`

1. Check out `/etc/bash.bashrc`
2. Check out `~/.bashrc`

We'll start off by modifying the user prompt. For example our user prompts look something like this




The first one is in OS X and the second in Linux. The user prompt is defined by the shell variable *PS1* which is set in `~/.bashrc`. Try using the **echo** command to see what it is currently set to. The following link might come in handy for the next task.

3. Change the value of *PS1* such that the prompt has the following format -
   "user@host - time [currentDirectory]:$"
   Note that the time should be in 24 hr format as HH:MM:SS and currentDirectory should not include the entire path from the home directory but should have only the name of the current directory. Here is the sample for better understanding.


Another common customization done is aliasing. Look here for more details. Usually at times you want to switch between two directories. In that case having a command to go back to the previous directory you visited can come in handy. This can be done using the **cd -**. On the other hand the previous directory you visited is stored in the shell variable OLDPWD

4. Alias **bk** with "cd $OLDPWD".

For the final part of terminal customization, we'll introduce you to ANSII escape sequences. When you type in **ls** you can see that **ls** prettifies its output using colors.

5. [EXTRA CREDIT] Modify `~/.bashrc` such that when you start the terminal you get a greeting that outputs something like


You're greeting must satisfy the following conditions

- It must make use of at least 2 colours
- It must make use of some sort of ASCII Art.

**Create a script** `myBashCustomization.sh` containing the commands used to achieve the sub-tasks 3, 4 and 5 with relevant comments to distinguish the commands for each sub-task.

Important point to note is that bashrc files are not just for configuring the terminal. The two bashrc files also contain a lot of interesting stuff. For example when you type a command with an incorrect spelling, `bash` suggest a possible correction. Or when you hit `tab` bash suggest possible completions of the command. How does bash do this? Answer this in no more than one paragraph in your `readme.txt`.

## B. Grepping Through Proc

A lot of useful system information such as your processor speed, ram size etc can be found by going through the files in the `/proc` directory. For the next set of tasks we'll be focusing on the memory usage.

1. Find which **file** in the `/proc` directory contains information regarding your memory usage.

More particularly we are interested in the Total and Free Memory in the system.

2. Using **grep** isolate the lines showing Total Memory and Free Memory of the system.

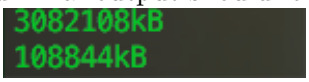
Currently you should have an output that looks something like.

```
MemTotal:        3082108 kB
MemFree:          499332 kB
```

Finally let's get rid of the text so that we are left only with the numbers

3. Pipe the output of **grep** to **tr** with relevant arguments to get rid of unwanted spaces.
4. Pipe the output of **tr** to **cut** with proper arguments so that only the values remain.

Your final output should look like




In your `readme.txt` give the name of the relevant file, and also mention the command you used to get the desired output. We are looking for a single line (the single line command can, and should include usage of pipes) .

## C. Landing on Mars

Many times during your stay at IITB you will find the need to access your CSE account or transfer files from your `mars` account to your local machine (laptop) and vice versa. These set of tasks make you familiar with the powerful features bash provides for doing this quickly. We'll start of by "ssh"ing to your mars account.

1. SSH to your mars account. Mention the command used in the `readme.txt`.

At times it's easy to just drag and drop your files to your mars account instead of using the terminal. Most of the major window managers allow you to access a remote server over SFTP using the following syntax `sftp://user@server`

2. Using Nautilus (File Manager used by Ubuntu), SFTP to mars

Submit a screenshot `sftpGUI.png` of the nautilus window.

Using the file manager when you double click on a file, it's opened (i.e., actioned) using the applications installed **on your system**. The application on the client end. But what if the client is a weak client (aka "thin client" in the industry; think of the client used in the JEE Mains online exam.) Further, you'll probably come across cases where only the remote system has the required program to action on the file. In such cases, the idea of using a file manager doesn't work. Linux comes with a built in solution, the X Window system, and ssh cooperates; you'll need to pass additional arguments when you ssh in such cases, especially applications which have a graphical output.

3. SSH to your mars account. Run gedit on `/tmp/lab3_inlab_C3.txt` using command line such that gedit is run on mars but the GUI of gedit is displayed to you on your local machine.

Mention the commands used for this task in `readme.txt`.

Currently every time you use ssh or scp you are asked to give a password. Now we configure your machine for a password-less entry into your mars account.

4. [EXTRA CREDIT] Set up password-less entry into your mars account

For the purpose of this you need to write a script `configurePasswordLessEntry.sh` which takes your cse_username and the target hostname as the input and configures your machine so that next time you try to remotely login to your mars account from your machine, you are not prompted for a password. The script should be run using the following syntax

**./configurePasswordLessEntry.sh cse_username targetHost**

Suggested commands: `ssh-keygen`,`ssh-copy-id`

## D. Killing A Processes

When you're working with the terminal you'll majorly come across two types of processes.

- Foreground Processes - These type of processes are connected to the terminal and your keyboard. For example in your first lab octave was an example of a foreground process. Usually hitting Ctrl+C can be used to kill such types of processes

- Background Processes - These type of processes are also connected to the terminal but not the keyboard and hence anything you type does not go to the process. You can send a process to the background by hitting Ctrl+Z or using `&` as the last argument.
- DAEMON - When a process is neither connected to a terminal nor the keyboard it's commonly known as a DAEMON (pronounced as demon)

A good introduction to foreground and background processes can be found here.

When it comes to killing processes nothing beats xkill, as long as the application has a GUI. Give it shot.

1. Key in "xkill" in a terminal and click on an open firefox/chrome window.

Killing processes that have a GUI is pretty straight forward. Let's focus on something a tad bit harder...killing a background processes.

2. Write a bash script called `infiniteLoop.sh` that as the name suggests, has an infinite loop
3. Open a terminal and run it in the background

Mention the command used for this task in `readme.txt`.

Now observe that even though the script has to run in an infinite loop you got back the prompt. Does that really mean the process has ended? If the process has not ended.

4. Kill the background process.

A few commands that might come in handy for the above task `ps,kill,pkill`. Finally lets kill a very **strong** process. Run the script `cell.sh`. The process runs in the foreground but hitting Ctrl+C has no effect on it.

5. [EXTRA CREDIT] Kill `cell.sh`

You're probably wondering why not kill all process using the same command that you used to kill `cell.sh`. One of the reasons is that using Ctrl+C gives a process an opportunity to perform some clean up tasks like say saving files, before exiting whereas if you use the method that you used for killing `cell.sh`, this can at times cause problems like loss or corruption of data.

For each of the sub-tasks 3, 4 and 5 explain in the `readme.txt` how you figured out the solution along with the commands used.

## Submission Guidelines

1. When you submit, please document individual percentages such as Student 1: 80%, Student 2:100%, Student 3:10%. In this example, the second student will get full marks (10/10) and the first student will receive 8/10.
2. Do include a readme.txt (telling me whatever you want to tell me). (The reflection essay is not required for inlab, but is required for outlab.) Do include group members (name, roll number), group number, honour code, citations etc.
3. The folder that you submit should contain scripts `myBashCustomization.sh`, `infiniteLoop.sh` and `configurePasswordLessEntry.sh`. It should also include the screenshot `sftpGUI.png`. `readme.txt` should contain answers to questions which asked you for explanations.
4. The folder and its compressed version should both be named `lab03_groupXY_inlab` for example folder should be named `lab03_group07_inlab` and the related `tar.gz` should be named `lab03_group07_inlab.tar.gz`

## How We will Grade You [30 Marks + 10 Extra credits]

Extra credit points are available to you only if you get the basics right.

1. Task A [**7 + 3 Marks**]
    - Sub-task 3 : 3 Marks
    - Sub-task 4 : 2 Marks
    - Sub-task 5 : 3 Marks [EXTRA CREDIT]
    - Reasoning : 2 Marks
2. Task B [**11 Marks**]
    - Sub-task 1 : 2 Marks
    - Sub-task 2 : 3 Marks
    - Sub-task 3 : 3 Marks
    - Sub-task 4 : 3 Marks
3. Task C [**5 + 4 Marks**]
    - Sub-task 1 : 1 Marks
    - Sub-task 2 : 1 Marks
    - Sub-task 3 : 3 Marks
    - Sub-task 4 : 4 Marks [EXTRA CREDIT]
4. Task D [**7 + 3 Marks**]
    - Sub-task 2 : 2 Marks
    - Sub-task 3 : 2 Marks
    - Sub-task 4 : 3 Marks
    - Sub-task 5 : 3 Marks [EXTRA CREDIT]