Swarm Software Team Task

Instructions

The following tasks would be done along with your mentors **Pranit Bauva**, **Spandan Kumar Sahu**, **Shivang Agrawal**, **Amegh Bhavsar**, **Anubhav Jain** and **Srishti Sammader**. All of the tasks need to be done in our Gymkhana upper lab and everyone has to bring their own laptops. The rest of the hardware (if required) will be available in the lab. Apart from these, we will have lots of discussion which will help you in doing the tasks. You will have to write a documentation about all the tasks describing what all approaches you took and why you chose one approach over another one. You will have to submit this documentation along with the working codes to successfully clear the task round.

Note: If you face any problem in understanding the topic, don't just copy the code, but instead ask any of the mentor to explain it. We would love to explain you the topics and we will definitely hate it if we see directly copied code from the internet (although we have tried our best that these things won't be directly available online).

Timings: 25th February to 4th March from 8.00pm to 12.00 midnight

Note: You have to submit all of this in a zip file email to **swarmiitkgp@gmail.com** by **5th March 2017 midnight** which is a **hard deadline** and we **won't** consider your submissions after that deadline.

1. Install **Ubuntu 16.04** if you don't have any other Linux distribution.

   Note: You can dual boot also but having a clean installation is always better.

2. Revise all of your Winter workshop contents in Image Processing or Autonomous Robotics.

   Note: If you haven't attended the winter workshop, then it is a **must** to get a good idea of all the concepts taught. Please contact any of the seniors for resources where you can study those.

3. Create a file named **input10000.txt** in which the first line will contain the number of integer values which are going to follow (we will be dealing with 10,000) and the next lines will contain 10,000 randomly generated integers. Make this by creating a C program.

4. Set a environment variable named **THREADS** to some value while invoking the program and print that value on the **standard output stream**.

   Resources:

- Shell variables and Environment Variables
- Getting Environment varibles inside C code

5. Design a merge-sort. Use the **input10000.txt** file which you created from above tasks. Then create similar files for 100000, 1000000, 10000000, 100000000 integer values. Then do a performance analysis of merge sort, make nice graphs and show them in the documentation and also verify that it follows the rule

$$\Theta(n \times \log n)$$

Resources:

- Merge Sort Tutorial - Kent University

6. Design a multi-threaded merge sort. Start a new recursion on a new thread. See the performance benefits and tell whether multi-threading is useful for this purpose.

Note: If you can think, then this **isn't** how you actually do multi-threading. Comment upon this in your documentation. Also write how it should be actually done. You can get a hint from your previous tasks.

Resources:

- Slides on Pthread from Illinois
- MIT's Slides for Multi threading
- POSIX Threads Programming