CSE 301: Operating Systems
Homework 1
(due Noon August 15)

Instructions

1. The deadline is a hard one. The form upload will close at sharp noon August 17. There will be no extensions.
2. You have to type the assignment using a word processing engine, create a pdf and upload on the form. Please note that only pdf files will be accepted.
3. Name the submission as {branch}_{roll_number}_{name}.pdf
4. All code examples must be put up as **secret gists** and linked in the created pdf. Again, only secret gists. Not the public ones.
5. Any instances of cheating/plagiarism will not be tolerated at all.
6. Cite all the pertinent references in IEEE format.
7. The least count of grading would be 0.5 marks.
8. Some suggestions for plotting - WolframAlpha, Academo.Org, Geogebra, Matploltib, GNUplot, Matlab, Octave
9. You can find the course VM on the course web page. The root password is: 1234

1. You are a student studying operating systems at IITGN. You hate the fact that the VM provided to you in the course does not have your username as the main user! Create a new user correspoding to your unique id, which consists of two parts - your first name and a number between 0 and 12. Obtain the latter by doing the following : $(2^{\texttt{last threedigits of your roll number}})\%13$. Capture a screenshot showing the newly created user on the command line. [**One mark**]. Seems too easy?[1]

[1] Oh, okay. This will not be graded. But, you want to have fun, no? Can you solve this

2. You are a system admin at PalajRocks. Your company provides students remote access to terminal with preset programming environment. Your friend told you about fork bomb. You're really worried. You do not want smart IITGN CSE and EE students to cause system crashes! You want to create a fool-proof system to prevent denial-of-service by a fork bomb. For the first-cut, you choose to do some tests on a VM (provided by course TAs), since you do not wish to crash your system[2]. You have the following tasks:

   (a) Write a C program which creates a new process via fork() every second [**One mark**]

   (b) Capture the increase in number of processes over time (using ps). You could do it manually (inefficient) by taking a screenshot every second. Could you automate it (and create a gif?) If you create a gif, upload it on a publicly accessible location and share the URL [**One mark**]

   (c) Ok. You need to stop the C program soon enough! Do it right away!

   (d) Figure out a mechanim by which you can cap the number of processes a user can generate. Let's keep it to 100 for now. Capture a screenshot of the command you run or the file you edit. [**One mark**]

   (e) Re-run the C program. Show that you're able to cap the processes to 100 and prevent a denial-of-service! [**One mark**]

3. You love command line arguments! Write a C program `bmi.c`, where you ask for a person's height (in meters) and weight (in pounds). An example instantiation of the program run is: `./bmi.out 1.8 200`. The output would be the BMI of the person upto two decimal places[3]. Ok, this was easy. Now, modify the program such that you create a child process. Now, access the command line arguments inside the child process and do the computation there and print out the BMI. Can you access the command line arguments of the parent process in the child process? Why or why not? [**One mark**]

4. You are the rockstar kernel developer at SebKhidki corporation. Yeah, your CEO could not think of a better name. Your boss is a great person. She wants to optimise the CPU scheduler. She says that instead of just optimising for average response time (RT) or for average turnaround time (TT), we wants you to minimise SKT[4], which is given by:

---

efficiecntly? Study modular exponentiation. Maybe you will realise that you do not need to compute the power of 2 and then compute the remainder.

[2]Does the problem sound too unrealistic? Think about CodeChef servers. Imagine if they do not have safety mechanisms in place. Someone could write a simple C fork-bomb!

[3]For the curious ones, BMI is not universally accepted as an indicator for health! Also, the BMI ranges for defining obesity can vary across countries.

[4]SebKhidki Time

$$SKT = \alpha \times RT^2 + \beta \times TT$$

Now, for the first cut she makes your job easier (or, does it become harder?). She tells you that the three variables (α, β, η) each can only take integer values from {0,1,$\cdots$100}

SebKhidki has a specific format for maintaining information about processes. For each process, there is a tuple containing the following fields <Process ID, Arrival Time, Length of Job, Pre-emptible or not?>

For testing purposes, you are given the following list of processes:

(a) <1, 0, 10, Non pre-emptible>
(b) <2, 0, 20, Non pre-emptible>
(c) <3, 0, 30, Non pre-emptible>

So, you have 3 processes, none of which can be interrupted.

(a) Find out SKT for longest job first and shortest job first as a function of α and β. Plot SKT for both these approaches as a function of α and β **[One mark]**
(b) Let us now assume that the three loads are pre-emptible. Each context switch takes τ timeunits. What is SKT as a function of α, β and τ for shortest time to completion algorithm? How would the SKT change if instead of <1, 0, 10, Pre-emptible> we had <1, 10, 10, Pre-emptible>? **[One mark]**
(c) What is SKT as a function of α, β and τ and η if we run round-robin scheduling with time slice of η. Load is: <1, 0, 10, Pre-emptible>, <2, 0, 20, Pre-emptible>, <3, 0, 30, Non pre-emptible>. Fix $\alpha = 10$, $\beta = 5$ and $\tau = 5$ and then plot SKT of round-robin as a function of η. What do you observe? **[One mark]**
(d) What is the default scheduler used in Linux (since v2.6.23)? Read the entire Wiki page to learn more! **[One mark]**

Not for grades

1. Can you measure the time taken for a context switch? Would `lmbench` be the right tool for this? What else can `lmbench` give you? Maybe, this is a project idea you want to explore?
2. Study real time scheduling from the paper entitled "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment". Why is "schedulability" such a challenge? Imagine being a NASA engnineer. Would schedulability matter to you now?
3. You might like to read the following inspiring article - In their own words: Unix pioneers remember the good times