CS4375: Theory of Operating Systems **Instructor: Dr. Deepak Tosh**

Fall 2021 **Due Date:** 10/15/2021, Midnight

Assignment # 3

This homework introduces you to a new tool, **vmstat**, and how it can be used to understand memory, CPU, and I/O usage. Read the associated README and examine the code in **mem.c** before proceeding to the exercises and questions below.

Note: You would need a Linux virtual machine (like Ubuntu) to work on this assignment because "**vmstat**" is a tool available in most Linux OS. If you face any difficulty in setting a linux VM, please reach out to our TA or me.

1. First, open two separate terminal connections to the same machine, so that you can easily run something in one window and the other. Now, in one window, run **vmstat 1**, which shows statistics about machine usage every second. Read the **man** page, the associated **README**, and any other information you need so that you can understand its output. Leave this window running **vmstat** for the rest of the exercises below.

   Now, run the program **mem.c** but with very little memory usage. This can be accomplished by typing **./mem 1** (which uses only 1 MB of memory). How do the CPU usage statistics change when running **mem**? Do the numbers in the **user time** column make sense? How does this change when running more than one instance of mem at once?
2. Let's now start looking at some of the memory statistics while running **mem**. We'll focus on two columns: **swpd** (the amount of virtual memory used) and **free** (the amount of idle memory). Run **./mem 1024** (which allocates 1024 MB) and watch how these values change. Then kill the running program (by typing control-c) and watch again how the values change. What do you notice about the values? In particular, how does the free column change when the program exits? Does the amount of free memory increase by the expected amount when mem exits?
3. We'll next look at the **swap** columns (**si** and **so**), which indicate how much swapping is taking place to and from the disk. Of course, to activate these, you'll need to run **mem** with large amounts of memory. First, examine how much free memory is on your Linux system (for example, by typing **cat /proc/meminfo**; type **man proc** for details on the **/proc** file system and the types of information you can find there). One of the first entries in **/proc/meminfo** is the total amount of memory in your system. Let's assume it's something like 8 GB of memory; if so, start by running mem 4000 (about 4 GB) and watching the swap in/out columns. Do they ever give non-zero values? Then, try with 5000, 6000, etc. What happens to these values as the program enters the second loop

(and beyond), as compared to the first loop? How much data (total) are swapped in and out during the second, third, and subsequent loops? (do the numbers make sense?)

4. Do the same experiments as above, but now watch the other statistics (such as **CPU utilization**, and **block I/O** statistics). How do they change when **mem** is running?
5. Now let's examine performance. Pick an input for mem that comfortably fits in memory (say 4000 if the amount of memory on the system is 8 GB). How long does **loop 0** take (and subsequent **loops 1, 2**, etc.)? Now pick a size comfortably beyond the size of memory (say 12000 again assuming 8 GB of memory).
   - How long do the loops take here?
   - How do the bandwidth numbers compare?
   - How different is performance when constantly swapping versus fitting everything comfortably in memory?
   - Can you make a graph, with the size of memory used by mem on the x-axis, and the bandwidth of accessing said memory on the y-axis?
   - Finally, how does the performance of the first loop compare to that of subsequent loops, for both the case where everything fits in memory and where it doesn't?

Grading:

The assignment will be graded on following items:

1. Completeness and correctness on your responses, explanations, and observations.
2. Inclusion of appropriate evidence (in form of screenshots)
3. Clarity of Report with all the required responses
4. References (including links where you found some sample code.