Assignment 4

Due Monday by 11:59pm **Points** 100

Assignment 4: **Process Scheduling**
(http://www.cs.umsl.edu/~sanjiv/classes/cs4760/assignments/ass4.pdf)

My comments to some questions asked by students:

- Child processes will have process control blocks that contain other information about the process to be run. The process control block should be in shared memory.
- The main process will perform the scheduling based on the algorithm specified in the assignment. The bit-vector will help with PID assignment and also to know which process is alive. The message queue will perform as you suggested.
- The round robin queue can have a pointer to the process control block (just the PID will do). The queue will only have the runnable processes.
- The child processes can be described as real-time in the process control block. As we discussed in scheduling, a real-time process should be scheduled if one exists, prior to scheduling any user process.
- In this project, you have four queues. I'll suggest keeping queue 0 at highest priority.
- The real-time processes should all go in queue 0 and will never jump to any other queue. As you have noted, user processes will ***not*** go into queue 0.
- Each queue (including the real-time queue) will be treated as a round-robin queue. However, you will always execute processes in real-time queue first if there is any in there. Make sure that you do not add too many real-time processes to allow running other processes.
- The message itself *must* be a struct that specifically contains a long type for the message type and a char array for the message itself.
- "Otherwise, the process starts to wait for an event that will last for r.s seconds where r and s are random numbers with range [0,3] and [0,1000] respectively, and 3 indicates that the process gets preempted after using p% of its assigned quantum, where p is a random number in the range [1,99]."
  1. Does this mean there are 4 possible states for each process? Or is the waiting and preempt the same?
     - Termination
     - Uses all quantum
     - Waiting for I/O
     - Preempt
  2. If the above answer is yes, then I am a little unsure on how to proceed with the preempt state. Does the process "pretend" to be interrupted and send the state back to the parent for the parent

to decide what to do with the process?

3. Do both the processes waiting for I/O and preempt get placed in the Blocked queue?

   It may not be a bad idea to have four states as you suggested. Only the processes waiting for I/O wil be on blocked queue, preempted processes will be in ready state.

- Normally, the preempted processes should be left at the head of the queue. This implies that they will be handled before any other process at the same priority level till they exhaust their quantum.
- ***The quantum, as described in the project description, appears backward. The higher priority queue should have a longer quantum. I have fixed it in the project description.***