

CSci 430/530 Project 1

Program is due on **Friday, March 11**

Part A:

Objective: Design and implement a **fairer scheduling algorithm** to replace the original one in XINU to solve the starvation problem for low priority process.

We discussed briefly a few different methods for scheduling. Recall:

- ▶ First-Come, First-Served (FCFS)
 - Processes are served in order they come into queue
 - Process runs until complete (it is not preempted)
 - More like batch scheduling
- ▶ Shortest-Job-First (SJF)
 - Processes have a length of time associated with each one
 - The currently waiting process that will take the shortest length of time is selected next.
 - SJF is optimal, gives minimum average waiting time for a set of processes (iff) the length of time is known or at least accurate
 - Problem, not always possible to know or estimate well the length of time for a job.
- ▶ Priority Scheduling
 - A priority is associated with each process
 - CPU is allocated to currently waiting process with highest priority
 - Priority scheduling is equivalent to SJF if higher priority is associated with normally shorter jobs
- ▶ Round Robin (RR)
 - Each process gets a small unit of CPU time, after this time has elapsed the process is preempted and added to the end of the ready queue.

Recall that in XINU, we use a combination of priority and round-robin scheduling. Jobs with different priorities waiting are scheduled according to the highest priority. Jobs of equal priority will work in a round robin fashion since there is a preemption time limit in XINU which causes a job to be forcibly rescheduled which allows for jobs of equal priority to get fair share of processor.

One problem with priority scheduling is that, low priority jobs can be starved out and never execute. You are asked to implement the aging solution, whereby as time progresses the priority of the process gradually increases.