Tutorial 3 : CPU Scheduling

August 27, 2019

Objective :

- Tutorial 3 is intended to help you learn some of the CPU scheduling algorithms discussed in class by implementing and simulating their performance.

Instructions:

- You are expected to run all the sample codes provided. It will help you understand diiferent ways of implementing the scheduler and compare the performance of the different implementations
- *fcfs.c* : Program for first come, first served (FCFS scheduling algorithm. FIFO simply queues processes in the order that they arrive in the ready queue. In this, the process that comes first will be executed first and next process starts only after the previous gets fully executed.
- *fcfs_1.c* : Program for FCFS CPU Scheduling with different arrival times. FIFO simply queues processes in the order they arrive in the ready queue. Here, the process that comes first will be executed first and next process will start only after the previous gets fully executed.
- *sjf.c* : Program for Shortest Job First (or SJF) CPU Scheduling (Non-preemptive). The scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN is a non-preemptive algorithm.
  Basic Idea : Sort all the processes in increasing order according to burst time.Then simply, apply FCFS
- *sjf_preeemptive.c* : Program for Shortest Job First (SJF) scheduling (Pre-emptive). In this scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute
  Basic Idea : Traverse until all process gets completely executed.
  a) Find process with minimum remaining time at every single time lap.
  b) Reduce its time by 1.
  c) Check if its remaining time becomes 0
  d) Increment the counter of process completion.
  e) Completion time of current process = current_time +1;
  e) Calculate waiting time for each completed process.
  wt[i]= Completion time - arrival_time-burst_time
  f)Increment time lap by one.

- *rr.c* : Program for Round Robin scheduling. Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.
  Basic Idea :
  Create an array rem_bt[] to keep track of remaining burst time of processes. This array is initially a copy of bt[] (burst times array)

  Create another array wt[] to store waiting times of processes. Initialize this array as 0.
  Initialize time : t = 0
  Keep traversing the all processes while all processes are not done.

  ```
  Do following for i'th process if it is not done yet.
  If rem_bt[i] > quantum
  (i) t = t + quantum
  (ii) bt_rem[i] -= quantum;
  Else // Last cycle for this process
  (i) t = t + bt_rem[i];
  (ii) wt[i] = t - bt[i]
  (ii) bt_rem[i] = 0;
  ```