**CPE435 - Lab 6: Process Scheduling**

A multiprogramming operating system allows more than one process to be loaded into the executable memory at one time and for the loaded process to share the CPU using time-multiplexing. One of the reasons for using multiprogramming is that the operating system itself is implemented as one or more processes, so there must be a way for the operating system and application processes to share the CPU, The scheduler is responsible for allocating the processor resources to the individual processes, there are different classes of scheduling that can be implemented for this purpose like Round-robin, Priority and FCFS.

1. Round-robin scheduling algorithm is one of the simplest scheduling algorithms, where each process gets a small unit of CPU time (*time quantum*), write a task that implements this scheduling, find the scheduling order, average waiting time for the processes and turn-around time (to each process) when round-robin scheduling is applied.

Hint:

* + **Assume all processes arrive at the same time**
  + **Define your own variables to perform the task**

/\* Process Data Structure \*/

struct process {

int pid; /\* Process ID \*/

int burst; /\* CPU Burst Time \*/

int working; /\* Working time, for round-robin scheduling if working ==burst then process complete it’s job and exit \*/

int t\_round; /\*Time need for a process to complet it's work \*/

};

/\*Function to be implemented: \*/

int read(); //read input from the user

void calc(); //to calculate turn-around of all processes and the ordering

void display(); //to show the order of execution

1. Priority based-scheduling is one of the scheduling algorithm in which processes are allocated to the CPU on the basis of an externally assigned priority. The key to the performance of priority scheduling is in choosing priorities for the processes, write a task to implements this way of scheduling, find the scheduling order, average waiting time for the processes and turn-around time (to each process) when Priority scheduling is applied(assume that a processes are Non-Preemptive)

Hint:

* + **Assume all processes arrive at the same time**
  + **Define your own variables to perform the task**

/\* Process Data Structure \*/

struct process {

int pid; /\* Process ID \*/

int burst; /\* CPU Burst Time \*/

int priority; /\* Priority \*/

int t\_round; /\*Time need for a process to complete it's work \*/

};

/\*Function to be implemented: \*/

int read(); //read input from the user

void calc(); //to calculate turn-around of all processes and the ordering

void display(); //to show the order of execution

Part A: Due Monday Sep 28, 2015 (in lab demonstration).

Part B: Due Wednesday Sep 30, 2015 (in lab demonstration).

Report is due Thursday October 1,2015

Good Luck