# CPE 460 Laboratory 4: CPU Scheduling Algorithms (FCFS and SJF)

Department of Computer Engineering
Yarmouk University

Spring 2017

## 1 Purpose

In this lab, you will simulate the following CPU scheduling algorithms:

- **First Come First Served (FCFS):** In FCFS scheduling, each process will be CPU scheduled based on its arrival time in the ready queue.
- **Non-preemptive Shortest Job First (SJF):** The SJF scheduling algorithm scheduled processes for running on CPU based on their CPU burst time; the process with the shortest CPU burst time is scheduled first. If two processes with the same CPU burst time exists in the ready queue, then FCFS scheduling approach is applied.

## 2 Exercise

Write a C program tht simulates the FCFS and *non-preemptive* SJF CPU scheduling algorithms. You can use the following template and data structures to complete your program. Repeat this experiment with different sets of processes and trace the waiting and turn-around times for each process.

```
#include <stdio.h>
#include <stdlib.h>
struct process{
    int pid;
    int burstTime;
    int waitingTime;
    int turnaroundTime;
}
void fcfsScheduling(struct process processes[], int processCount);
void sjfScheduling(struct process processes[], int processCount);
void main(){
    // 1- Prompt the user to enter the number of processes and allocate
    // an array of struct process to accomadate the number of user processes

    // 2- Prompt the user to enter the burstTime for each process

    // 3- Prompt the user to select the scheduling algorithm: FCFS or SJF
}
```

```
void fcfsScheduling(struct process p[], int n){
    // Write the code for FCFS scheduling in this function
}
void sjfScheduling(struct process p[], int n){
    // Write the code for SJF scheduling in this function
}
void displayGanttChart(struct process p[], int n){
    int i;
    printf("----------------------------");
    printf("\n\nGANTTChart\n-");
    for(i=0; i<(p[n-1].turnaroundTime + 2*n);i++){ printf("-"); }
    printf("\n|");
    for(i=0; i<n; i++){
        k = p[i].burstTime/2;
        for(j=0; j<k;j++){ printf(" "); }
        printf("P%d",p[i].pid);
        for(j=k+1; j<p[i].burstTime; j++){ printf(" "); }
        printf("|");
    }
    printf("\n-");
    for(i=0; i<(p[n-1].turnaroundTime + 2*n);i++){ printf("-"); }
    printf("\n0");
    for(i=0; i<n; i++){
        for(j=0; j<p[i].burstTime;j++){ printf(" "); }
        printf("%2d",p[i].turnaroundTime);
    }
}
```

Here is a sample of how your program output will look like:

```
Enter number of processes: 4
Enter CPU burst time for P1 (ms): 10
Enter CPU burst time for P2 (ms): 4
Enter CPU burst time for P3 (ms): 11
Enter CPU burst time for P4 (ms): 6
Enter [1] for FCFS scheduling and [2] for Non-preemptive SJF scheduling: 1
---------------------------------------
Process    B-Time    T-Time    W-Time
---------------------------------------
P1         10        10         0
P2          4        14        10
P3         11        25        14
P4          6        31        25
---------------------------------------
Average Waiting Time: 12.25 ms
Average Turnaround Time: 20.00 ms
GANTT Chart
-------------------------------------------
|     P1     |  P2  |     P3      |   P4   |
-------------------------------------------
0           10      14           25        31
```